

Draft from January 17, 2023



# Contents

## Preamble

## Acknowledgements

### 1. Introduction

1-1. What is a computable function? . . . . .	2
1-2. What are the non-computable functions? . . . . .	3
1-3. Motivations . . . . .	4
1-4. Overview of Computability Theory . . . . .	5

### 2. Cantor's infinity

2-1. Equipotence and subpotence . . . . .	13
2-2. Cantor–Bernstein's theorem . . . . .	14
2-3. Countable sets . . . . .	16
2-4. Cantor's diagonal argument . . . . .	19
2-5. Non-computable reals . . . . .	21
2-6. Cantor space . . . . .	22

## I Classical Computability Theory 27

### 3. Foundations of computability

3-1. Computable functions . . . . .	29
3-2. Computable sets . . . . .	34
3-3. Universal program . . . . .	35
3-4. SMT theorem . . . . .	36
3-5. Padding lemma . . . . .	38
3-6. Kleene's fixed point theorem . . . . .	38
3-7. Computably enumerable sets . . . . .	42

<b>4. Turing degrees</b>	
4-1. Finite strings . . . . .	49
4-2. Computation with oracle . . . . .	50
4-3. Relativization of proofs . . . . .	52
4-4. Use property . . . . .	54
4-5. Turing degrees . . . . .	55
4-6. Turing jump . . . . .	58
4-7. Limit computability . . . . .	59
4-8. Finite extensions method . . . . .	64
4-9. Low degrees . . . . .	69
4-10. High degrees . . . . .	71
<b>5. Arithmetic hierarchy</b>	
5-1. Elementary properties . . . . .	79
5-2. Arithmetic hierarchy and computability . . . . .	84
5-3. Relativization to an oracle . . . . .	85
5-4. Many-one degrees . . . . .	87
5-5. Post's theorem . . . . .	88
5-6. Rice's theorem . . . . .	91
5-7. Arithmetic codes . . . . .	92
<b>6. Church-Turing thesis</b>	
6-1. The Entscheidungsproblem and the quest for the Grail . . . . .	97
6-2. Church-Turing thesis . . . . .	102
6-3. Detailed study of recursive functions . . . . .	104
<b>7. Immunity and function growth</b>	
7-1. Immune sets . . . . .	128
7-2. DNC functions . . . . .	129
7-3. Arslanov completeness criterion . . . . .	134
7-4. Hyperimmune functions . . . . .	135
7-5. Computably dominated degrees . . . . .	137
7-6. Martin's domination theorem . . . . .	144
7-7. High or DNC degrees . . . . .	148
<b>8. <math>\Pi_1^0</math> classes and PA degrees</b>	
8-1. Binary trees . . . . .	152
8-2. Topology on Cantor space . . . . .	155
8-3. $\Pi_1^0$ classes . . . . .	163
8-4. Basis theorems . . . . .	166
8-5. Basis for perfect $\Pi_1^0$ classes . . . . .	169
8-6. PA degrees . . . . .	171
8-7. Finitely-branching trees . . . . .	176

<b>9. Formal interlude</b>	
9-1. A little history: the crisis of foundations . . . . .	185
9-2. First-order logic . . . . .	191
9-3. Incompleteness theorems of Gödel . . . . .	208
9-4. ZFC system . . . . .	218
<b>10. Cohen forcing</b>	
10-1. Formulas of second-order arithmetic . . . . .	226
10-2. $\Sigma_1^0/\Pi_1^0$ forcing . . . . .	228
10-3. Effective genericity . . . . .	236
10-4. $\Sigma_n^0/\Pi_n^0$ forcing . . . . .	250
10-5. Arbitrarily generic sets . . . . .	258
<b>11. Effective forcing</b>	
11-1. Fundamentals of forcing . . . . .	265
11-2. Forcing relation . . . . .	268
11-3. Forcing with trees . . . . .	271
11-4. Computational complexity and forcing question . . . . .	275
<b>12. Quest for natural degrees</b>	
12-1. Three emblematic undecidable problems . . . . .	290
12-2. Natural Turing degrees . . . . .	293
12-3. Mass problems . . . . .	296
<b>13. Priority method and c.e. degrees</b>	
13-1. C.e. degrees . . . . .	300
13-2. Permitting method . . . . .	300
13-3. $\Sigma_1^0$ priority method (finite injury) . . . . .	301
13-4. $\Sigma_2^0$ priority method . . . . .	310
13-5. $\Pi_2^0$ priority method (infinite injury) . . . . .	313
<b>14. Structure of the Turing degrees</b>	
14-1. Minimal degrees . . . . .	324
14-2. Nature of $\mathcal{D}$ . . . . .	330
14-3. Universality of $\mathcal{D}$ . . . . .	334
14-4. First-order theory of $\mathcal{D}$ . . . . .	339
14-5. Structure of the c.e. degrees . . . . .	347

## II Algorithmic Randomness

349

### 15. Introduction

<b>16. Kolmogorov complexity and random numbers</b>	
16-1. Kolmogorov complexity . . . . .	356
16-2. Random numbers in the sense of Chaitin/Levin . . . . .	364
16-3. Characterization of prefix-free complexity . . . . .	371
16-4. K-trivial sets . . . . .	374
<b>17. Borel classes, measure and computability</b>	
17-1. A little history . . . . .	379
17-2. First intuitions about Algorithmic Randomness . . . . .	383
17-3. Borel classes . . . . .	386
17-4. Lebesgue measure . . . . .	391
<b>18. Martin-Löf Randomness</b>	
18-1. Intuitions and definitions . . . . .	399
18-2. The Martin-Löf and Chaitin/Levin randomnesss coincide . . . . .	402
18-3. Randomness and Turing degrees . . . . .	403
18-4. Randomness and DNC degree . . . . .	406
<b>19. Other randomness notions</b>	
19-1. Weak-2 randomness . . . . .	411
19-2. Relativization of randomness . . . . .	416
19-3. 2-randomness . . . . .	419
19-4. Incomplete random . . . . .	423
<b>20. The K-trivials</b>	
20-1. Lowness and bases for randomness . . . . .	429
20-2. Golden run . . . . .	434
20-3. Characterization of the c.e. K-trivials . . . . .	446
20-4. New proof that K-trivial implies low-for-K . . . . .	453

## III Reverse Mathematics 457

<b>21. Introduction</b>	
21-1. Quest for optimal axioms . . . . .	459
21-2. Comparison of theorems . . . . .	462
<b>22. Second-order arithmetic</b>	
22-1. $Z_2$ language . . . . .	466
22-2. $Z_2$ theory . . . . .	467
22-3. Semantics of second-order arithmetic . . . . .	470
22-4. Formalizing Analysis in $Z_2$ . . . . .	475
22-5. $RCA_0$ and computable mathematics . . . . .	478
22-6. $ACA_0$ and the arithmetic hierarchy . . . . .	485

22-7. $WKL_0$ and the compactness argument . . . . .	488
22-8. More powerful systems . . . . .	494
<b>23. Induction and conservation</b>	
23-1. $RCA_0$ -provably computable functions . . . . .	502
23-2. Weak PA subsystems . . . . .	504
23-3. Induction hierarchies . . . . .	509
23-4. Primitive recursive functions and $RCA_0$ . . . . .	517
23-5. Bounded comprehension scheme . . . . .	520
23-6. Conservation theorems . . . . .	523
23-7. Hilbert program . . . . .	534
<b>24. Computable reductions</b>	
24-1. $\omega$ -reduction . . . . .	540
24-2. Computational reduction . . . . .	545
24-3. Weihrauch reduction . . . . .	547
24-4. Reduction games . . . . .	551
24-5. Strong reductions . . . . .	553
<b>25. Ramsey's theorem</b>	
25-1. Overview . . . . .	558
25-2. Ramsey's theorem in the arithmetic hierarchy . . . . .	561
25-3. Infinite pigeonhole principle . . . . .	573
25-4. Ramsey's theorem for pairs . . . . .	591
 <b>IV Higher Computability Theory</b>	 <b>603</b>
<b>26. Introduction</b>	
26-1. Motivations . . . . .	606
26-2. Panorama of higher computability . . . . .	608
26-3. Correspondence with Classical Computability Theory . . . . .	609
<b>27. Transfinite numbers</b>	
27-1. Motivation: computable iterations of the jump . . . . .	611
27-2. Ordinals . . . . .	614
27-3. Induction and transfinite recurrence . . . . .	622
27-4. Countable and uncountable ordinals . . . . .	629
27-5. Effective ordinals . . . . .	632
27-6. Relativization . . . . .	639

<b>28. Hyperarithmetical sets</b>	
28-1. Kleene hierarchy . . . . .	641
28-2. $\Pi_2^0$ singletons . . . . .	649
28-3. Relativization . . . . .	651
28-4. Effective Borel hierarchy . . . . .	652
<b>29. Beyond hyperarithmetical</b>	
29-1. A little history: the Moscow school . . . . .	657
29-2. Second-order quantifications . . . . .	661
29-3. The $\Pi_1^1$ sets and the well-orders . . . . .	667
29-4. Analogies between $\Pi_1^1$ sets and c.e. sets . . . . .	670
29-5. Kleene/Souslin equivalence theorem . . . . .	674
29-6. Other boundedness theorems . . . . .	681
29-7. Hyperarithmetical reduction . . . . .	682
<b>30. <math>\Sigma_1^1</math> and <math>\Pi_1^1</math> classes</b>	
30-1. Canonical representation of $\Sigma_1^1$ classes . . . . .	685
30-2. Basis theorems for $\Sigma_1^1$ classes . . . . .	687
30-3. The continuum hypothesis for $\Sigma_1^1$ classes . . . . .	691
30-4. Some emblematic $\Pi_1^1$ classes . . . . .	694
30-5. Study of a very special $\Pi_1^1$ class . . . . .	696
30-6. $\Pi_1^1$ singletons . . . . .	701
<b>31. The systems <math>\text{ATR}_0</math> and <math>\Pi_1^1\text{-CA}_0</math></b>	
31-1. Definitions . . . . .	707
31-2. $\text{ATR}_0$ and $\Pi_1^1\text{-CA}_0$ in higher computability . . . . .	709
31-3. HYP is not a model of $\text{ATR}_0$ . . . . .	711
31-4. Non-standard ordinal codes . . . . .	715
31-5. Separation between $\text{ATR}_0$ and $\Pi_1^1\text{-CA}_0$ . . . . .	721
<b>A. Exercise solutions</b>	
<b>Bibliography</b>	<b>769</b>
<b>Notations</b>	
<b>Notations</b>	<b>785</b>
<b>Index</b>	
<b>Index</b>	<b>791</b>

# Preamble

This book is an introduction to Computability Theory as well as to three of its main ramifications, namely, Algorithmic Randomness, Reverse Mathematics and Higher Computability Theory. It is mainly intended for research master students and teachers in Computer Science and Mathematics, as well as for researchers wishing to acquire a solid knowledge of Computability Theory.

## Reason for existence of the book

This book was initially written in French, aiming to make up for the lack of reference book on classical Computability Theory in the French literature. However, while writing this book, the authors realized this project was also for interest outside of the French community for the follows reasons:

There are many reference books in English on Computability Theory (*Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers* by Piergiorgio Odifreddi [168], *Computability Theory* by Barry Cooper [40] or *Turing computability : Theory and Applications* by Robert Soare [207]). Concerning Algorithmic Randomness Theory, we will cite *Computability and Randomness* by André Nies [166], and *Algorithmic Randomness and Complexity* by Rodney Downey and Denis Hirschfeldt [49]. In Reverse Mathematics, the historical reference is *Subsystems of Second Order Arithmetic* by Stephen Simpson [202]. We will mention the more recent work by Denis Hirschfeldt, *Slicing the Truth* [86], and *Reverse Mathematics: Problems, Reductions, and Proofs* by Damir Dzhafarov and Carl Mummert. Finally, in Higher Computability Theory, the two references are *Higher Recursion Theory* by Gerald Sacks [191] and *Recursion Theory: Computational Aspects of Definability* by Chi Tat Chong and Liang

Yu [34]. Each of these works presents the state of the art of research for a specific sub-domain of Computability Theory, but there is no single book providing a consistent presentation of these different aspects.

## Organization of the book

This book is structured in four main parts, namely Classical Computability Theory, Algorithmic Randomness, Reverse Mathematics and Higher Computability Theory.

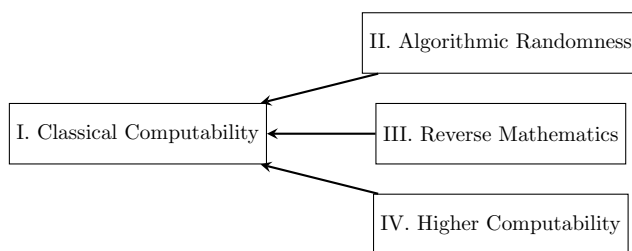
### General plan

- *Classical Computability Theory* is the study of Turing degrees, in other words, the computational power of sets of natural integers. It constitutes the historical heart of Computability Theory, and the epistemic base on which the following three parts are based.
- *Algorithmic Randomness* uses Classical Computability Theory to give an effective framework to Measure Theory, which makes it possible to study individually the sequences of bits known as “random”. The hierarchies induced by Classical Computability Theory make it possible to define various levels of randomness, in the light of which one can, for example, re-examine the meaning of such or such probabilistic theorem stating that a property is true almost everywhere.
- *Reverse Mathematics* form a program on the foundations of mathematics, which aims to identify the axioms necessary to prove mathematical theorems of everyday life<sup>1</sup>. They are based on a basic theory,  $\text{RCA}_0$ , whose axioms capture “computable” mathematics, thanks to a correspondence between computability and definability by logical formulas.
- *Higher Computability Theory* extends the notion of computability to a more general framework joining Set Theory. Just as elementary arithmetic operations extend to ordinals, Turing machines can extend their computing time from integers to ordinals, and thus handle larger classes of reals. This is the approach by “models of computation” of hypercomputing, which corresponds, as for Classical Computability Theory, to certain logical classes.

The last three parts all rely heavily on Classical Computability Theory, but are relatively independent from each other, and can be mostly read in any order:

---

<sup>1</sup>From the everyday life of the mathematician.



Dependencies between the four main parts of the book

Let us note that the notion of “Borel class” introduced in Part II is fundamental for the understanding of Part IV.

### **Classical Computability Theory**

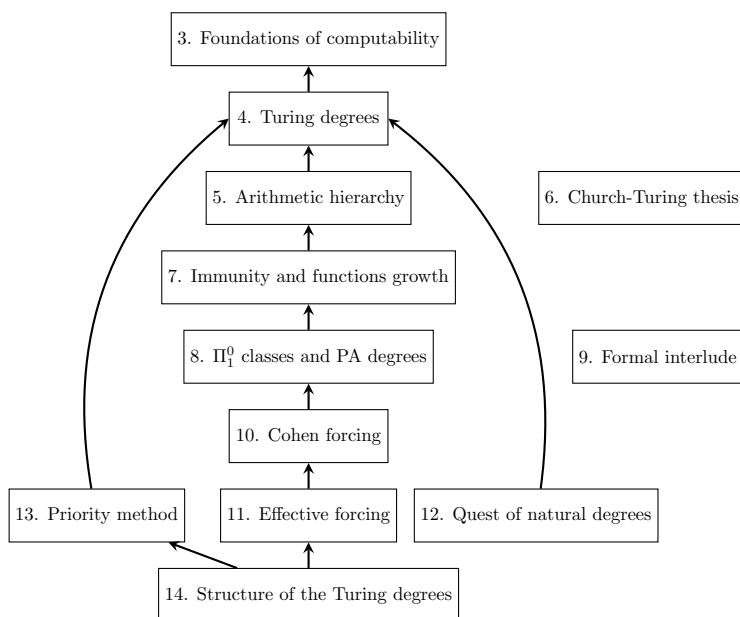
Classical Computability Theory has a preponderant role, in that it fixes a formal framework and a series of tools which will be used to develop the following parts. It is therefore advisable to linger on the first part and to detail the dependencies of its chapters. The fundamental chapters are mainly to be read linearly, with the following exceptions: chapters 6 and 9 can be read independently of the others, but will nevertheless be useful to serenely approach Part III of the book, on Reverse Mathematics. Chapters 12 and 14 will not be absolutely necessary for the understanding of the following parts, and aim to take a step back from our work. Chapter 12 — less technical — will do this through the examination of a specific question, on the borderline of philosophy; and Chapter 14 through a more abstract study of the structure of Turing degrees, and the presentation of some of the major open questions in the field.

## **How to read this book**

### **For teachers**

This book can be used as a support for an introductory course on Computability Theory at master’s level, as well as for more advanced thematic courses, talking about Algorithmic Randomness, Reverse Mathematics and Higher Computability Theory. The topics covered go well beyond the knowledge of Computability Theory that one would expect from a master’s student. We are therefore going to propose a lesson plan containing the essential concepts.

The equivalence between the computational models forms a robust basis for the development of Computability Theory. However, the proofs can seem



Dependencies between the chapters of Classical Computability Theory

quite long and tedious. Nowadays, with the popularization of computers, one can expect that students will have a certain intuition of what an algorithm is, and it seems better to start from this intuition to make the first developments in order to avoid a relatively heavy formalism. We recommend to approach the equivalence of computational models, in particular between general recursive functions and Turing machines, through a tutorial session, where students will have the opportunity to manipulate the formalisms by defining more and more complicated functions to convince oneself that these definitions allow to capture all algorithms.

We invite our readers to follow the developments of the various chapters 3, 4, 5, 7, 8, 10, 13 in that order. Chapter 3 establishes the first fundamental theorems of Computability Theory on the basis of the intuition that we have of algorithms. Chapter 4 defines the notions of oracle machine and of Turing degree. We find there the most central definitions of Computability Theory, such as the Turing jump, and the finite extension method, which is a very powerful technique to prove the existence of some Turing degrees. Chapter 5 on the arithmetic hierarchy establishes an essential link between the computational power of sets of integers and their definability by arithmetic formulas, through Post's theorem.

With Chapter 7, we begin the study of various fundamental computational properties, such as the notion of hyperimmune degree, and its links with the existence of fast-growing functions. This study is continued in Chapter 8 on  $\Pi_1^0$  classes, where we define the notion of PA degree which is a central notion in Computability Theory. It is found throughout this book, particularly in Algorithmic Randomness and Reverse Mathematics.

Chapter 10 introduces a fundamental technique of Computability Theory, namely forcing, presented here as an elaboration of the finite extension method of Chapter 4. This chapter can also serve as the first step of an incremental understanding of the general technique of forcing in Set Theory.

Chapter 13 finally introduces the priority methods, another fundamental technique of Computability Theory, which makes it possible in particular to show the existence of some computably enumerable degrees.

### **For students**

The skills required to understand the concepts presented in this work are those of a first year of a bachelor's degree in Computer Science or Mathematics. It is essential to understand the usual mathematical language (variables, quantifications, etc.), to have some elementary notions of logic (proof by contraposition, proof by the absurd, etc.), and to understand the elements of the basic mathematical corpus (understanding what a bijection is, what an intersection between two sets is, the power and logarithm functions, etc.). In addition to that, at least a basic experience in programming, or an understanding of what an algorithm is, is also necessary to serenely approach the reading of this book.

The mathematics that we will use and which are not taught in the first year of the license will be introduced and explained as and when required (basic notions of Topology or Measure Theory for example). Having established this, let us note all the same that the degree of elaboration of the proofs, as well as the technicality of certain concepts, will undoubtedly make this work difficult to approach without a certain mathematical maturity.

The techniques developed in Computability Theory are quite different from those learned through a standard mathematical course. This particularity of Computability Theory is a strength and makes this discipline more accessible since it is not very sensitive to the gaps that one may have developed during his course (or his lack of course). On the other hand, this difference can also destabilize the student because it requires creating a conceptual universe. It goes without saying that in the absence of a teacher, it is all the more essential to do the exercises suggested in the book to properly integrate the concepts. The solutions are available at the end of the book.

The size of this book can be overwhelming for a student wanting to take his first steps in Computability Theory. We remind you that this book covers knowledge going far beyond what is expected of a master's student. We therefore recommend that autodidacts follow the lesson suggestion in the previous section, intended for teachers.

### **For researchers**

This book is an introduction to Computability Theory and several of its main branches. However, we should not stop at the introductory aspect of this work, because most of the results presented correspond to the state of the art of research. This book is therefore aimed at researchers in related fields, wishing to acquire solid knowledge in Computability Theory, as well as Ph.D. students and researchers intending to do some research in Computability Theory. Indeed, the techniques and concepts of this book make the research articles in the field directly accessible.

## **Exercises**

The chapters are interspersed with exercises of varying difficulty, the correction of which is given at the end of the book. We cannot stress enough the importance of doing exercises to properly assimilate the concepts presented in the chapters. The intuition of concepts is created by manipulating them in all their forms. The difficulty of the exercises is indicated using a star system (★) ranging from 0 to 3: an exercise with no star is a direct application of the definitions, while a two-star exercise requires a deep mastery of concepts to be solved. There are also some three-star exercises, which are “research” level.

Through this work, we will present many computational properties on sets of integers or on other more complex structures. In addition to the given exercises, it is important to show an intellectual curiosity consisting in systematically seeking how these properties combine, knowing whether one can construct objects satisfying several of them simultaneously, and so on. Likewise, when the theorems have hypotheses, it is useful to look for counter-examples without these hypotheses, in order to better understand their necessity as well as their use in the proof.

## **Errata**

You can't write a book of this size without letting a number of typos slip through. This book will probably not deviate from this rule. We will

maintain a list of typos on the authors web page. You can report errors to one of the following addresses:

`benoit.monin@computability.fr` or  
`ludovic.patey@computability.fr`



# Acknowledgements

Our very first thanks go to the National Research Agency, which largely funded the collaboration between the two authors, especially during the writing of the book, within the framework of the project “Aspects Calculatoires des Théorèmes Combinatoires – ACTC”

<https://anr.fr/Projet-ANR-19-CE48-0012>

Several results presented in this book, in particular Theorem 25-3.23 on the pigeonhole principle or the simplified proof of Liu’s theorem 25-3.24 come from articles funded by the same project.

We would also like to thank our teams and related organizations, which provided us with a breeding ground for intellectual emulation, as well as moral and financial support. During the writing of this work, Monin was lecturer in the Algorithmics, Complexity and Logic Laboratory of the University of Paris-Est Créteil and Patey researcher at the CNRS, in the team Algèbre, Géométrie, Logique at the Camille Jordan Institute in Villeurbanne.

Many colleagues have generously helped us to improve the quality of this work, by giving a critical look at its scientific and educational content, drawing on their respective expertise, or by pointing out the typographical errors that inevitably crept into the book. We would therefore like to thank Paul-Elliot Anglès d’Auriac, Sébastien Tavenas, Pascal Vanier, Mathieu Hoyrup, Benjamin Hellouin, Laurent Bienvenu, Denis Kuperberg, Julien Cervelle, Damir Dzhafarov, Loïc Gassmann, William Gaudelier, Denis Hirschfeldt, Quentin Le Houerou, Alexander Shen, Keita Yokoyama, Adrien Deloro, Pascal Monin and Shahin Amini.

The scientific content of the book is above all the work of the computability-theoretic community. The authors forged their intuitions by reading the

works of their predecessors and adding their contribution to this magnificent intellectual edifice. We would like to thank our colleagues at the international level for the collaborations and mutual visits which have improved our understanding of the subject.

We would also like to thank Laurent Bienvenu, who through his work and through the direction of our respective theses, has been able to transmit his passion to us, and has largely contributed to the introduction of Computability Theory in France.

Writing a book of this magnitude takes a lot of time and energy, and could not have happened without the moral support of our families and friends.

# Chapter 1

## Introduction

The scientist does not study nature because it is useful to do so. He studies it because he takes pleasure in it, and he takes pleasure in it because it is beautiful. If nature were not beautiful it would not be worth knowing, and life would not be worth living. I am not speaking, of course, of the beauty which strikes the senses, of the beauty of qualities and appearances. I am far from despising this, but it has nothing to do with science. What I mean is that more intimate beauty which comes from the harmonious order of its parts, and which a pure intelligence can grasp.

Science and Method, Henri Poincaré

**What is Computability Theory?** Computability Theory is classically considered to be one of the four pillars of logic, alongside Set Theory, Model Theory and Proof Theory. The field was initially forged on the question of what characterizes the functions  $f : \mathbb{N} \rightarrow \mathbb{N}$  whose values can be obtained by a process purely *mecanizable* or *algorithmic*, in a finite time, although arbitrarily large. We will say that such functions are *effectively computable*. Long before the appearance of the first computers, Computability Theory based its theoretical basis on an observation — or rather a miracle — namely, the existence of a robust definition, consensual and independent of any formalism, of the epistemological notion of effectively computable function.

The initial question, that is, “What is a computable function?” having obtained a satisfactory answer, the study naturally turned to the question of knowing, among the natural functions, which are computable and which

are not. Subsequently, the field has undergone considerable development thanks to the notion of relative computability, the question no longer being to determine whether a function is computable or not, but to identify the computational power intrinsic to this function, through questions like “If this function were computable, which other functions could we compute?”

More recently, the subject of study has extended to very many mathematical objects — for instance algebraic structures, or subsets of  $\mathbb{R}$  — and has given many ramifications. We will see in this book in particular that Computability Theory serves as a robust foundation for Algorithmic Randomness, and for Reverse Mathematics, the objects of study of which are the mathematical theorems themselves.

Nowadays, the appellation “Computability Theory” for a field which studies arbitrary mathematical objects, most of which are not computable, may seem surprising, even a residue of its historical subject of study. In reality, this name is still valid, but its meaning has changed: the term *computability* no longer relates to the subject of the study, but to the angle from which the subject is approached. A modern one-sentence definition of Computability Theory might be: **Computability Theory is the study of mathematics under the prism of their computational complexity.**

## 1. What is a computable function?

The main difficulty of this question lies in obtaining a class of functions sufficiently robust, not to depend on the computer model, the choice of programming language, technological progress, or the advance of knowledge in such a general way.

With the advent of computers, the notion of algorithm has gradually taken root in scientific culture. Anyone who has already had a first contact with programming will have formed a good idea of what an automatable task is. Based on our knowledge of computer engineering, the following definition would come naturally: “A function is effectively computable if it has an algorithm, in other words if it can be programmed in a sufficiently expressive programming language, and executed by a sufficiently powerful computer.”

This definition, if it has the advantage of being in adequacy with our intuition, does not provide a sufficiently formal framework for reasoning about the class of computable functions. A second approach would consist in fixing a computer and a standard programming language, and defining a function as computable if it is programmable in this language, and executable by this computer in finite time, using sufficient memory. If one does not worry about the speed of execution, nor the necessary memory space, it quickly appears that this definition coincides with the preceding one.

In fact, the power and memory of computers increase with technological progress, and therefore allow programs to be executed more quickly, but do not necessarily increase the class of computable functions. Even computers based on new computing paradigms, like quantum or biological computers, are simulable — with the cost of an exponential time and space overhead — by classical computers, and therefore do not change the class of computable functions. As for programming languages, the existence of operating systems and interpreters make it easy to convince oneself that the main ones such as C++, Java or Python, allow programming — more or less elegantly — the same mathematical functions. This therefore empirically shows a certain robustness in the definition of the class of programmable functions.

A problem remains: what is the guarantee that today's computers represent the limit of what is automatable, or computable by a human being? Who tells us that with the progress of science, we will not discover a new paradigm of computation or a new way of reasoning allowing to consider as computable a larger class of functions? This is the subject of a long foundational quest started in the 20th century, and culminating in the famous Church-Turing thesis in 1936, which we will present in Chapter 6.

## 2. What are the non-computable functions?

With regard to our previous definition, for the moment very informal, most — if not the entirety — of the mathematical functions used on a daily basis are computable: addition, multiplication, the function  $(n, m) \mapsto n^m$ , the function which at  $n$  associates the  $n$ -th prime number, or even the one which computes the greatest common divisor of two natural integers, are all computable. We can add to this list less trivial examples: the function which takes a computer program written in C++ and determines if the program is syntactically correct — this is what does a C++ compiler, among others — or the one which returns the  $n$ -th decimal of  $\pi$ ,  $\sqrt{2}$  or the golden ratio — each of these numbers is the sum of a computable sequence of rationals with a sufficiently fast convergence — or to finish the one which to  $n$  associates the number of possible games that we can play in Go on a board — also known as *goban* — of size  $n \times n$ . This last example illustrates in particular the following fact: one does not deal in Computability Theory with the time that a computation takes. Only the existence of an algorithm matters to us. In the case of the number of games in Go, the algorithm in question is based on a simple idea; it “suffices” to list all the possible games and to count them. However, there are execution time of such an algorithm is so large that it makes it impossible to use in practice for  $n > 2$  <sup>(1)</sup>. For  $n = 19$ , which is the size of a standard goban, this number ranges

---

<sup>1</sup>There are already 386 356 909 593 possible games on a goban of size  $2 \times 2$  [224]!

from  $10^{10^{48}}$  to  $10^{10^{171}}$  [224], which is clearly too many games to count, even if all the world's computers harnessed it for a billion years ...

Although the function of multiplying by 2 is, in a sense, much more accessible to us than the one that counts the number of games of go, there is an algorithm that computes each of them. These two functions are therefore not different from one another from the point of view of Computability Theory: they are both computable, and we will mainly be interested in the functions which *are not*, that is to say functions whose values *cannot* be obtained by a purely mechanizable or algorithmic process. The mere existence of such functions is not self-evident, and one of the first tasks we will tackle will be to demonstrate their existence. This will be done in the next chapter via Cantor's diagonal argument. We will then give many examples of such functions throughout the book, the best known of which is undoubtedly the *halting problem*, defined as the function which takes a program as input, and determines whether its execution will halt, necessarily in a finite amount of time. We will see that the halting problem cannot be computed; and it is important to understand that it is indeed a question here of a theoretical and fundamental impossibility, which does not depend on the power or speed of computation of the computers. The non-computability of the halting problem is not due to an ignorance of its algorithm which could one day be discovered, but indeed to an absolute impossibility, because the existence of such an algorithm would lead to a paradox.

### 3. Motivations

Computability Theory relates mainly to the study of non-computable functions —or more general mathematical objects—. It is legitimate to wonder if such a study is really reasonable. If even some computable functions are inaccessible to us —like the number of possible plays at Go— then why bother to think about functions *even more inaccessible*?

A first motivation for our study is exploratory. There are inaccessible objects, let's try to explore the universe. Simply because it is there, and out of curiosity about the mysteries it contains. Our efforts will be rewarded with a series of theorems of great depth. Anyone who immerses himself seriously in the developments of this book, once perhaps past some difficulties of adaptation inherent in any scientific discipline, will see a world of astounding richness come to life in his mind, with its flora and fauna, its rules and mechanisms. Computability Theory is characterized by the highly dynamic nature of its proofs, each of which provides insight into the detailed workings of a fragment of the titanic machinery that animates this universe.

A second motivation arises quite simply out of necessity. The Pythagoreans found themselves constrained and forced to admit the existence of irrational measures, such as the diagonal of a side square 1, which went against their understanding of the world, which they believed to be explicable only on the basis of the relationships between integers. But if we admit the existence of integers, and the existence of the square, we are forced to admit that of *incommensurable* quantities, which we call today irrational, like  $\sqrt{2}$ . In the same way, we will see that if we admit the existence of computable objects, we are also forced to admit the existence of objects which are not, and which nevertheless appear naturally in a whole series of situations.

A final motivation is of a practical nature. Computability Theory, through its understanding of non-computable objects, has achieved major success in providing a formal framework for the study of questions at the borders between science and philosophy. We will see two of them: the search for the definition of random objects with Part II, and the understanding of what *strength* means of a theorem, in particular with respect to another, with Part III. Part IV of this book will bring Computability Theory to the frontier that it shares with Set Theory.

## 4. Overview of Computability Theory

Computability Theory can be broken down into several subdomains, which are all based on the same robust notion of effectively computable function.

### 4.1. Areas covered by this book

This book is broken down into four parts, each of them covering an offshoot of Computability Theory: Classical Computability Theory, Algorithmic Randomness, Reverse Mathematics, and Higher Computability Theory.

#### Classical Computability Theory

As we have mentioned, Computability Theory relates above all to objects which cannot be computed. Classical Computability Theory focuses on functions  $f : \mathbb{N} \rightarrow \mathbb{N}$  as well as on sets of integers  $E \subseteq \mathbb{N}$ . Note that such a set can also be represented by its characteristic function  $\chi_E : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $\chi_E(x) = 1$  if  $x \in E$ , and  $\chi_E(x) = 0$  otherwise.

The developments of Classical Computability Theory revolve around a fundamental tool which will allow us to compare or even measure the *degree of non-computability* of a function, also called *degree of unsolvability* or *Turing degree*, with reference to the mathematician Alan Turing who introduced the notion. Let us fix a non-computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$ . It is natural to ask “If I were able to compute  $g$ , which other functions could

I compute?” We say that a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is *computable relative to  $g$*  (or  *$g$ -computable*) if there exists an algorithm allowing to compute  $f$  in an extended programming language, where we would have added the function  $g$  as primitive: a special instruction allows us to call the function  $g$  on a parameter  $n$  in our program, as if it really existed, and to retrieve the result. If  $f$  is  $g$ -computable, nothing tells us how to compute  $g$ , but if we had a “oracle” allowing us to compute the values of  $g$ , it would be possible to compute the values of  $f$ .

This notion of relative computability allows us to define a partial pre-order between the functions, noting  $f \leq_T g$  if the function  $f$  is  $g$ -computable. This is the *Turing reduction*. Different functions can carry the same computational power, in the sense that they are mutually computable. We therefore define the *Turing degree* of a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  as the set  $\deg_T f$  of all the functions  $g$  such that  $f \leq_T g$  and  $g \leq_T f$ . The notion of Turing degree represents a computational power, in the sense that two functions of the same Turing degree are indistinguishable from the point of view of computability. The partial pre-order on the functions induces a partial order on the Turing degrees.

Classical Computability Theory relates mainly to the study of Turing degrees together with the partial order relation defined above. Are there an infinite number of computational powers? Are they linearly ordered? More generally, what are the properties of this partial order? It turns out that this structure is extremely rich and complex, as we will have the opportunity to see.

## Algorithmic Randomness

The classical Probability Theory studies probabilistic phenomena, successfully modelled via the notion of measure which is used to formally define the laws of probability. On the other hand, this theory does not have the necessary tools —and it is not its goal— to speak of random objects *individually*. It is this precise point that Algorithmic Randomness proposes to clarify, by relying on Computability Theory. Let’s go through an example.

Let us represent a real number  $R \in [0, 1)$  by its binary expansion, of the form  $R = 0.b_0b_1b_2b_3 \dots$  where  $(b_n)_{n \in \mathbb{N}}$  is a sequence of bits. Suppose that the real  $R$  is obtained by drawing its bits *randomly* by a tossing sequence. We assume of course that each draw is *equiprobable*: we have a 50% chance of getting heads and a 50% chance of getting tails. Intuition tells us that the real  $R$  thus obtained is *random*. What does this mean exactly? We do not expect, for example, to obtain only “heads” on the first hundred thousand throws: if each draw is equally likely, this cannot happen, or in any case with such a low probability that we can consider it negligible. We do not expect to obtain twice as much of “heads” as of “tails” either.

Again, the probability of this happening in 100,000 draws is so low that one will assume the draws to be biased rather than witnessing such an unlikely event. We can in fact identify a first property that we are entitled to expect from a sequence of equiprobable draws: the sequence obtained should respect the law of large numbers, that is to say that the number of draws “heads” and “tails” should roughly be the same.

However, is this sufficient? Suppose now for example that on each number  $n$ , if  $n$  is a prime number, we systematically obtain equally a “heads”. In the hypothesis where a certain obsession with prime numbers would lead us to notice this curious phenomenon, we will again be faced with a — slightly absurd — enigma and we will be led to think that in one way or another, something abnormal is happening. But let’s take a further step back. Basically, and regardless of the sequence of bits obtained, we can identify numbers  $n_1 < n_2 < n_3 < \dots$  such that the draw numbers  $n_1, n_2, n_3, \dots$  are all draws “heads”. In the case where our sequence  $n_1, n_2, n_3, \dots$  contains prime numbers, this seems to us to be a “probabilistic bug”, but why should it be acceptable if  $n_1, n_2, n_3, \dots$  are any integers? This is where Computability Theory comes into play, and will allow us to precisely formalize the properties that a sequence of random bits should have — according to our human intuition —.

### Reverse Mathematics

The notion of *theorem* relates to a system of axioms. When one omits to mention the system of reference, it is commonly accepted that one refers to the system of Zermelo Fraenkel (ZF), which represents a set of consensual axioms which serve as the foundation of all mathematics. The ZF system is, however, very powerful, and we have no guarantee that it will not be inconsistent.

Reverse Mathematics aims to find the axioms necessary and sufficient to prove the theorems of everyday mathematics. It is therefore a question of studying existing theorems, to find more elementary proofs, or on the contrary to show the optimality of their proof. Better understanding the hypotheses of theorems allows to better control their “fragility” in the face of a potential contradiction of the proof system. It is therefore a meta-mathematical approach aimed at answering the question “Which confidence can we have in our mathematics?”

At first glance, this approach is not linked to Computability Theory. However, Reverse Mathematics relies to a basic theory,  $\text{RCA}_0$ , capturing the *computable mathematics*, and which represents a basis of confidence more in connection with the concrete world, because its objects being computable, they can be represented by an algorithm, therefore they have a finite description. Reverse Mathematics therefore consists, given a theorem  $T$ , in

searching for axioms  $A$  such that  $\text{RCA}_0$  proves the equivalence between  $A$  and  $T$ . By the choice of the basic theory  $\text{RCA}_0$ , equivalences are computational processes calling on the tools of Computability Theory.

## Higher Computability Theory

One of the reasons for the success of Computability Theory as a tool for analyzing mathematics lies in the existence of a strong intuition of the notion of computation, thus making it possible to guide the manipulation of concepts and to prove theorems without being embarrassed by a heavy formalism. Higher Computability Theory aims to extend the reach of these tools to more powerful computational models, which can be seen as machines having the possibility of continuing their execution for an infinite computation time (formally in ordinal computation time). Just as notions of Classical Computability Theory can be captured by logical formulas, so can Higher Computability Theory. For example, where the sets of integers which can be enumerated (out of order) by a computer program are those which can be described by a  $\Sigma_1^0$  formula of arithmetic, those which are enumerable by a hypercomputer programs are those which can be defined by a  $\Pi_1^1$  formula of arithmetic.

We will see that this aspect of things brings Higher Computability Theory closer to Descriptive Set Theory, a branch of Set Theory which classifies sets according to the degree of difficulty in describing them. Higher Computability Theory can be seen as a bridge between Descriptive Set Theory and Classical Computability Theory.

## 4.2. Other branches of Computability

In order to allow this work to keep a reasonable size, we have chosen to ignore two important branches of Computability Theory, namely Computable Structure Theory and Degrees of Enumeration.

### Computable Structure Theory

It is a branch of Computability Theory that studies the extent to which the algebraic properties of a mathematical structure affect their descriptive complexity. By structure, we mean sets equipped with operations, such as groups, rings and fields, but also any structure within the meaning of Model Theory. This branch borrows its techniques from both Model Theory and Classical Computability Theory to answer this question.

Concretely, this theory studies countable structures and asks questions of the form “Given a computable structure  $\mathcal{A}$ , what are the possible Turing degrees of structures isomorphic to  $\mathcal{A}$  ?” or “Given two isomorphic computable structures, what is the computational complexity of their isomorphism?” For example, some structures like the dense linear orders without

endpoints are computably isomorphic to all their computable copies. They are called *computably categorical*.

### Degrees of Enumeration

Classical Computability Theory places “computable sets” as the reference computational power. But some problems are expressed naturally in the form of non-computable sets, the elements of which can however be enumerated out of order by a computable process. We call these sets *computably enumerable* (c.e.). In particular, if  $E$  is a set of integers c.e. and if  $n \in E$ , it is possible to realize it in finite time, by launching the enumeration procedure and waiting for  $n$  to appear. On the other hand, if  $n \notin E$ , then it will not be possible in general to know it in a finite time. For example, the set of Diophantine equations (equations with integer coefficients, of type  $3x^3 - 2y^2 + x - 2 = 0$ ) which admit integer solutions is computably enumerable, because it suffices to search exhaustively for solutions, and to enumerate the equation if a such a solution exists.

Degrees of Enumeration place “computable enumeration” as the reference power. We can define an *enumeration reducibility*  $A \leq_e B$  iff any enumeration of the elements of  $B$  computes an enumeration of the elements of  $A$ . This reduction is a partial pre-order, which induces a notion of *enumeration degree*: the degree of enumeration of  $A$  is the set  $\deg_e(A)$  of all sets  $B$  such that  $A \leq_e B$  and  $B \leq_e A$ . The study of the degrees of enumeration equipped with the partial order  $\leq_e$  constitutes an active branch of research in Computability Theory.



## Chapter 2

# Cantor's infinity

If we had to give a date to the birth of modern logic, we would place it without hesitation in 1872, the date on which Georg Cantor exposes his first proof of Theorem 4.1 to come of the uncountability of real numbers. Cantor isolates later the quintessence of this first proof through is famous *diagonal argument*, which will have a central place in Computability Theory.

Cantor's work then marked the beginning of a “complex” Set Theory which will play a big role in the foundational quest of mathematics in the early 20th century, which we will talk about in detail in Chapter 9. This crisis will lead to the development of Mathematical Logic as we know it today, with modern Set Theory, known as ZFC, but also with the development of the first theories of calculus, used by Gödel to show his famous incompleteness theorem, which can be seen as a sophisticated variation of Cantor's diagonal argument.

What is it exactly? Cantor shows that infinite sets do not all have the same “size”. There are strictly more real numbers than integers, in a sense that we will define precisely in a few lines. Cantor will use this discovery to develop a mathematical study of infinity. In particular, he will create the



Georg Cantor, 1845–1918

transfinite numbers, which will constitute the backbone of mathematical definitions, and which we will discuss in Chapter 27.

Cantor, however, was not the first to notice that infinity does not obey the same rules as the finite. In particular, a surprising characteristic of infinite sets is that the whole is not necessarily greater than its parts. Galileo makes a luminous exhibition of it in his work “Dialogues Concerning Two New Sciences” [70] through a tasty dialogue between two characters, Salviati and Simplicio:

- Salviati. *This is one of the difficulties which arise when we attempt, with our finite minds, to discuss the infinite, assigning to it those properties which we give to the finite and limited.; but this I think is wrong, for we cannot speak of infinite quantities as being the one greater or less than or equal to another.[...] I take it for granted that you know which of the numbers are squares and which are not.*
- Simplicio. *I am quite aware that a squared number is one which results from the multiplication of another number by itself; thus 4, 9, etc., are squared numbers which come from multiplying 2, 3, etc., by themselves.*
- Salviati. *Very well; and you also know that just as the products are called squares so the factors are called sides or roots; while on the other hand those numbers which do not consist of two equal factors are not squares. Therefore if I assert that all numbers, including both squares and non-squares, are more than the squares alone, I shall speak the truth, shall I not?*
- Simplicio. *Most certainly.*
- Salviati. *If I should ask further how many squares there are one might reply truly that there are as many as the corresponding number of roots, since every square has its own root and every root its own square, while no square has more than one root and no root more than one square.*
- Simplicio. *Precisely so.*
- Salviati. *But if I inquire how many roots there are, it cannot be denied that there are as many as there are numbers because every number is a root of some square. This being granted we must say that there are as many squares as there are numbers because they are just as numerous as their roots, and all the numbers are roots.*

On reading Galileo's dialogue, we observe the paradox trap closing in on Simplicio. Galileo uses for that a concept which will be taken up by Cantor: two sets  $A$  and  $B$  “have as many elements” if one can make exactly correspond the elements of  $A$  and the elements of  $B$ , in other words if there is a bijection between the two sets.

## 1. Equipotence and subpotence

Recall that a function  $f : E \rightarrow F$  is *injective* if

$$\forall x, y \quad x \neq y \Rightarrow f(x) \neq f(y),$$

*surjective* if its image is the whole set  $F$ , and *bijective* if it is both injective and surjective.

**Definition 1.1.** Two sets  $E$  and  $F$  are *equipotent* if there is a bijection between them. We will then write  $|E| = |F|$ . ◇

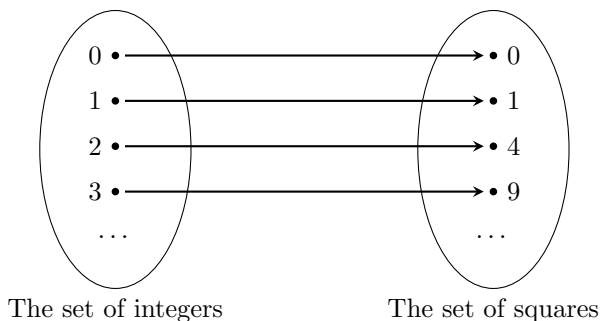


Figure 1.2: Galileo's argument to say there are “as many” integers as square numbers

According to our definition, the integers and the squares of integers therefore have the same cardinality: there are as many elements in the two sets. What seems paradoxical is that the squares of integers form a strict part of the set of integers. The paradox is solved in the simplest possible way: the intuition that we have on finite sets, which wants a strict subset of a set to contain fewer elements is simply no longer true for infinite sets.

### Remark

The notation  $|E| = |F|$  seems to suggest equality between two objects  $|E|$  and  $|F|$ , which we call respectively *cardinality* of  $E$  and *cardinality* of  $F$ . It is possible to give a precise definition of  $|E|$ , as a mathematical object. For the moment, we will be satisfied with defining cardinality as the informal notion of the size of a set, and if the object  $|E|$  is not clearly defined, the statement  $|E| = |F|$  is, and is sufficient to our treatment of infinity.

Note that Galileo uses his idea to explain precisely why there is no sense in comparing the size of infinite sets. This is where all the genius of Can-

tor intervenes, who will discover that contrary to Galileo's intuition, it is possible to give a formal notion of size to infinite sets: there are infinities "larger" than others.

Intuitively, a set  $F$  is at least as big as a set  $E$  if we can match the elements of  $E$  to distinct elements of  $F$ , in other words if we can associate each element of  $E$  with its own "representative" in  $F$ .

**Definition 1.3.** A set  $E$  is *subpotent* to a set  $F$  if there is an injection of  $E$  into  $F$ . We then write  $|E| \leq |F|$ . If  $E$  is not subpotent to  $F$ , we write  $|E| \not\leq |F|$ . ◇

It is easy to verify that this relation is *transitive*, ie, if  $|E| \leq |F|$  and  $|F| \leq |G|$ , then  $|E| \leq |G|$ . Indeed, if the functions  $f : E \rightarrow F$  and  $g : F \rightarrow G$  are two injections, then their composition  $g \circ f : E \rightarrow G$  is an injection witnessing the relation  $|E| \leq |G|$ . It is however much less clear that if  $|E| \leq |F|$  and  $|F| \leq |E|$ , then  $|E| = |F|$ . Unrolling the definitions, the question comes back to knowing if, when there is an injection of  $E$  in  $F$  and another of  $F$  in  $E$ , there is a bijection between  $E$  and  $F$ . Let us see straight away that this is indeed the case: it is the Cantor–Bernstein theorem.

## 2. Cantor–Bernstein's theorem

The heart of the proof of the Cantor–Bernstein theorem lies in the following lemma.

**Lemma 2.1.** If  $B \subseteq A$  are sets, and  $f : A \rightarrow B$  is an injective function, then there is a bijection  $h : A \rightarrow B$ . ★

PROOF. The reader can use Figure 2.2.

Let  $C_0, C_1, C_2, \dots$  be the sequence defined by induction as follows:

$$C_0 = A \setminus B \text{ and } C_{n+1} = f(C_n).$$

Let  $C = \bigcup_n C_n$ . A simple reasoning by induction on  $n$  allows to show, using the injectivity of  $f$ , that the sets  $C_n$  for  $n \in \mathbb{N}$  are pairwise disjoint, although this is not necessary in the proof. Let the function  $h : A \rightarrow B$  be defined by:

$$h(x) = \begin{cases} f(x) & \text{if } x \in C \\ x & \text{if } x \notin C. \end{cases}$$

Let us show that  $h$  is injective. The function  $f$  being injective, the function  $h$  restricted to  $C$  is injective. The function  $h$  restricted to  $A \setminus C$  is the identity function, and is therefore also injective. Finally, the image

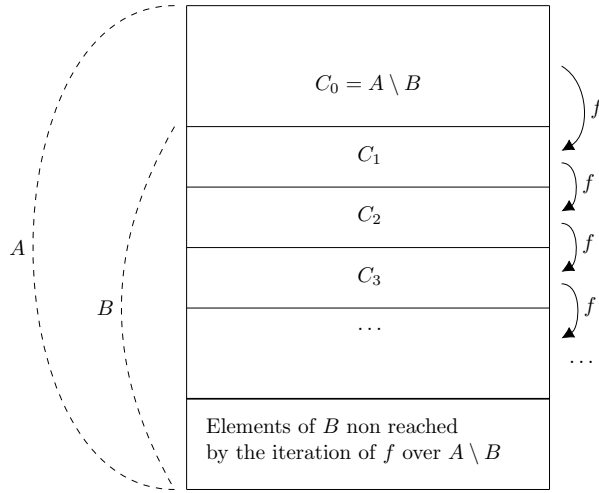


Figure 2.2: Illustration of the proof of Lemma 2.1

of  $C = \bigcup_n C_n$  by  $h$  is

$$\bigcup_n f(C_n) = \bigcup_n C_{n+1} \subseteq C,$$

and the image of  $A \setminus C$  by  $h$  is  $A \setminus C$ . The function  $h$  is the union of two injective functions having disjoint images, and is therefore injective.

Finally, let us show that  $h$  is surjective. Let  $y \in B$ . If  $y \notin C$ , then  $h(y) = y$  and therefore  $y$  has a predecessor by  $h$ . If  $y \in C$ , as  $y \notin C_0$ , there exists an  $n \in \mathbb{N}$  such that  $y \in C_{n+1}$ . By definition of  $C_{n+1} = f(C_n)$ , there exists an  $x \in C_n$  such that  $h(x) = f(x) = y$ . ■

We can now show the announced theorem.

**Theorem 2.3 (Cantor–Bernstein)**

*If  $A$  and  $B$  are sets, and  $f : A \rightarrow B$  and  $g : B \rightarrow A$  are injective functions, then there is a bijection between  $A$  and  $B$ .*

PROOF. Let  $A' = g(B)$ . The application  $g \circ f$  is an injection of  $A$  into  $A'$ . By Lemma 2.1, there is therefore a bijection  $h : A \rightarrow A'$ . The function  $g$  being a bijection between  $B$  and  $A'$ , the function  $g^{-1} \circ h : A \rightarrow B$  is a bijection. ■

**Corollary 2.4**

*Two mutually subpotent sets are equipotent.*

The question of knowing if there are infinities of distinct sizes, and in particular with one of them strictly larger than the other, therefore comes down to knowing if we can find two sets  $A, B$  for which  $|A| \leq |B|$  but  $|B| \not\leq |A|$ . We will see that this is indeed the case.

### 3. Countable sets

The infinite set par excellence is of course  $\mathbb{N}$ , the set of integers. In this case, a specific vocabulary is used.

**Definition 3.1.** A set  $A$  is said to be *countably infinite* if there is a bijection  $f : \mathbb{N} \rightarrow A$ , in other words if  $A$  and  $\mathbb{N}$  are equipotent. A set  $A$  is *countable* if it is finite or countably infinite. Otherwise,  $A$  is said to be *uncountable*. ◇

**Remark**

Some authors use the term “denumerable” to denote countably infinite sets.

Intuitively, the infinity of natural numbers is the smallest infinity, in the sense that it is subpotent to any infinite set.

**Proposition 3.2.** The set  $\mathbb{N}$  is subpotent to any infinite set. ★

PROOF. Let  $A$  be an infinite set. We are going to define an injective function  $f : \mathbb{N} \rightarrow A$  by induction on  $\mathbb{N}$ . Let  $f(0)$  be any element of  $A$ . Suppose the values  $f(0), \dots, f(n)$  are defined. In particular,  $B = \{f(0), \dots, f(n)\} \subseteq A$  is a finite set while  $A$  is infinite. There is therefore necessarily an element in  $A \setminus B$ . Let  $f(n+1)$  be this element. By construction,  $f$  is injective. ■

Note that in its full generality, the previous proof uses the *axiom of choice*, which we will talk about again in Section 9-4, and which is necessary in order to choose at each step an element of  $A \setminus B$ . In most cases, however (as in the following corollary), this axiom is not absolutely necessary.

**Corollary 3.3**

*Any subset of a countable set is countable.*

PROOF. Let  $A$  be a countable set, and let  $B \subseteq A$  be a subset. If  $B$  is finite, then it is countable. Suppose  $B$  is infinite. In particular,  $A$  is also

infinite, so  $A$  is equipotent to  $\mathbb{N}$ . The set  $A$  being equipotent to  $\mathbb{N}$ , it is therefore subpotent to  $\mathbb{N}$ . As  $B \subseteq A$ , it is subpotent to  $A$ , therefore to  $\mathbb{N}$ . Furthermore, by Proposition 3.2,  $\mathbb{N}$  is subpotent to  $B$ . By the Cantor-Bernstein theorem (Theorem 2.3),  $B$  and  $\mathbb{N}$  are equipotent. ■

**Exercise 3.4. (★)** Let  $A$  be a countably infinite set, and let  $f : \mathbb{N} \rightarrow A$  be a bijection. Finally, let  $B \subseteq A$  be an infinite subset. Directly construct a bijection from  $\mathbb{N}$  to  $B$  which is not based on the axiom of choice. By this, we mean that the definition of the function must be based on an explicit algorithm, and not on an abstract procedure making it possible to choose an element in a non-empty set, without knowing which element it is. ◇

Let us now introduce a bijection that we will use regularly in this book, which is the one commonly used to witness the countability of the product  $\mathbb{N} \times \mathbb{N}$ .

**Proposition 3.5.** The set  $\mathbb{N} \times \mathbb{N}$  is countably infinite. ★

PROOF. Let  $\alpha : \mathbb{N} \rightarrow \mathbb{N}^2$  be the function such that the values

$$\alpha(0) = (0, 0), \quad \alpha(1) = (1, 0), \quad \alpha(2) = (0, 1), \dots$$

enumerate the pairs of integers by successive diagonals, as in Figure 3.6.

The function is injective by construction, and any pair will appear at one stage of the enumeration. Thus,  $\alpha$  is a bijection from  $\mathbb{N}$  to  $\mathbb{N} \times \mathbb{N}$  witnessing the equipotence between the two sets. ■

It is possible to give an analytical definition of the reciprocal function of  $\alpha$  defined in the previous proposition. It will therefore be a bijection from  $\mathbb{N}^2$  to  $\mathbb{N}$ , which we will call  $\alpha_2$  and which the reader will be able to discover through the exercise below.

**Exercise 3.7. (★)** Let  $\alpha_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$  defined by:

$$\begin{aligned} \alpha_2(x, y) &= y + \sum_{i=0}^{x+y} i \\ &= y + \frac{(x+y+1)(x+y)}{2}. \end{aligned}$$

1. Show that  $\alpha_2$  is bijective.
2. Show that  $\alpha_2(a, b) \geq a$  and  $\alpha_2(a, b) \geq b$ . ◇

The bijection  $\alpha_2$  of the previous exercise will be very often used in the developments of this book, via the following notation.

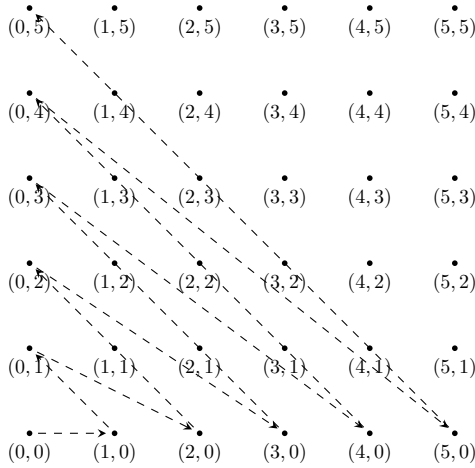


Figure 3.6: Illustration of the proof of Proposition 3.5

### Notation

We denote by  $\langle n, m \rangle$  the integer to which the pair  $(n, m)$  is sent via the bijection  $\alpha_2$ .

Note that if we have a bijection  $\alpha_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$ , we can define a bijection  $\alpha_3 : \mathbb{N}^3 \rightarrow \mathbb{N}$  by simply taking  $\alpha_3(x, y, z) = \alpha_2(x, \alpha_2(y, z))$ . We can continue in this way to define bijections from  $\mathbb{N}^n$  to  $\mathbb{N}$  for any  $n \in \mathbb{N}^*$ , which leads to the following notation.

### Notation

We denote by  $\langle x_1, \dots, x_k \rangle$  the integer on which the  $k$ -tuple  $(x_1, \dots, x_k)$  is sent via the bijection from  $\mathbb{N}^k$  to  $\mathbb{N}$  described above.

Let us take advantage of our freshly introduced bijections to deduce the following corollary.

### Corollary 3.8

*The Cartesian product of two countably infinite sets is countably infinite.*

PROOF. Let  $A$  and  $B$  be countable sets and let  $f : A \rightarrow \mathbb{N}$  and  $g : B \rightarrow \mathbb{N}$  be bijections witnessing it. The function  $h : A \times B \rightarrow \mathbb{N}$  defined by  $h(x, y) = \langle f(x), g(y) \rangle$  is a bijection. ■

Let us end this section with three exercises, allowing to manipulate the concepts seen so far. In particular, we draw attention to the first of them,

which will be used regularly in future developments.

**Exercise 3.9. (★)** Show that  $\mathbb{Z}$  is a countably infinite set. Deduce that  $\mathbb{Z} \times \mathbb{Z}$  is a countably infinite set and finally that the set  $\mathbb{Q}$  of rational numbers — which can be written under the form  $p/q$  with  $p, q \in \mathbb{Z}$  with  $q \neq 0$  — is also countably infinite.  $\diamond$

**Exercise 3.10. (★)** Let  $f : \mathbb{N} \rightarrow A$  be a surjective function towards an infinite set  $A$ . Show that  $A$  is countably infinite.  $\diamond$

**Exercise 3.11. (★)** Let  $(B_n)_{n \in \mathbb{N}}$  be a sequence of countable sets with their respective bijections  $f_n$  with  $\mathbb{N}$ . Show that  $B = \bigcup_n B_n$  is a countable set.

Warning: if we do not have the bijections  $(f_n)_{n \in \mathbb{N}}$ , we can always show that  $B$  is countable, but we must use the *axiom of choice*, in order to choose uniformly for each  $B_n$  one of its bijections with  $\mathbb{N}$ . We will talk about it again in Section 9-4.  $\diamond$

## 4. Cantor's diagonal argument

Let us now tackle the theorem announced at the beginning of this chapter: there exist infinities larger than others, and in particular an infinity larger than that of integers. The following theorem uses Cantor's famous diagonal argument, which will be taken up on numerous occasions and in various forms throughout this book.

### Theorem 4.1 (Cantor)

*The set of real numbers is uncountable.*

PROOF. The reader can use Figure 4.2, which illustrates the argument of the proof. We reason through the absurd. Let us suppose on the contrary that there exists a bijection  $f : \mathbb{N} \rightarrow \mathbb{R}$ . We are going to construct a real number  $R \in \mathbb{R}$  which is not in the image of  $f$ . We simply define  $R$  as follows: the integer part of  $R$  is 0, and for any  $n$ , if the  $n$ -th decimal of  $f(n)$  is different from 0, then the  $n$ -th decimal place of  $R$  is equal to 0. Conversely, if the  $n$ -th decimal of  $f(n)$  is equal to 0, then the  $n$ -th decimal of  $R$  is equal to 1.

It is clear that for any integer  $n$ , our real number  $R$  cannot be equal to  $f(n)$ , because the  $n$ -th decimal of  $R$  is different from the  $n$ -th decimal place of  $f(n)$ .  $\blacksquare$

$R$	$=$	$0,$	$9 - x_{00}$	$9 - x_{11}$	$9 - x_{22}$	$9 - x_{33}$	$9 - x_{44}$	$9 - x_{55}$	$9 - x_{66}$
$f(0)$	$=$	$N_0,$	<b><math>x_{00}</math></b>	$x_{01}$	$x_{02}$	$x_{03}$	$x_{04}$	$x_{05}$	$x_{06}$
$f(1)$	$=$	$N_1,$	$x_{10}$	<b><math>x_{11}</math></b>	$x_{12}$	$x_{13}$	$x_{14}$	$x_{15}$	$x_{16}$
$f(2)$	$=$	$N_2,$	$x_{20}$	$x_{21}$	<b><math>x_{22}</math></b>	$x_{23}$	$x_{24}$	$x_{25}$	$x_{16}$
$f(3)$	$=$	$N_3,$	$x_{30}$	$x_{31}$	$x_{32}$	<b><math>x_{33}</math></b>	$x_{34}$	$x_{35}$	$x_{36}$
$f(4)$	$=$	$N_4,$	$x_{40}$	$x_{41}$	$x_{42}$	$x_{43}$	<b><math>x_{44}</math></b>	$x_{45}$	$x_{46}$
$f(5)$	$=$	$N_5,$	$x_{50}$	$x_{51}$	$x_{52}$	$x_{53}$	$x_{54}$	<b><math>x_{55}</math></b>	$x_{56}$
$f(6)$	$=$	$N_6,$	$x_{60}$	$x_{61}$	$x_{62}$	$x_{63}$	$x_{64}$	$x_{65}$	<b><math>x_{66}</math></b>
$\dots$									

Figure 4.2: Illustration of Cantor's diagonal argument. We construct our real  $R$  using the diagonal of the table, the decimal  $9 - x_{ii}$  being different from  $x_{ii}$ . Note that the real  $R$  described here is not exactly the one described by the proof, but the result is the same: the element  $R$  does not belong to the image of  $f$ .

What does the previous theorem tell us? That there are “more” real numbers than integers. These sets of numbers are both infinite, but the infinity of reals is “larger” than that of integers. Indeed, when we try to match an integer to each real number, we see that there are always “exceeding” reals, which do not correspond to any integer. We therefore have  $|\mathbb{R}| \not\leq |\mathbb{N}|$ . Note that we have on the other hand  $|\mathbb{N}| \leq |\mathbb{R}|$ , via identity injection. In this case, we can write  $|\mathbb{N}| < |\mathbb{R}|$ , meaning that there is an injection of  $\mathbb{N}$  in  $\mathbb{R}$ , but no injection of  $\mathbb{R}$  in  $\mathbb{N}$ .

Is there an infinity greater than that of the reals? Cantor also answered this question, with a similar argument: for any set  $A$ , there exists a set  $B$  such that  $|A| < |B|$ .

**Theorem 4.3 (Cantor)**

*For any set  $A$ , the set  $\mathcal{P}(A)$  of the subsets of  $A$  is such that  $|A| < |\mathcal{P}(A)|$ .*

PROOF. The set  $A$  is subpotent to  $\mathcal{P}(A)$ , considering the injection which to any element  $x \in A$  associates the singleton  $\{x\}$ . Thus,  $|A| \leq |\mathcal{P}(A)|$ . Let us now assume absurdly that  $|\mathcal{P}(A)| = |A|$ , i.e., suppose there is a bijection  $f : A \rightarrow \mathcal{P}(A)$ . Consider the set

$$B = \{x \in A : x \notin f(x)\},$$

and let  $y$  be such that  $f(y) = B$ . Then,  $y \in B$  iff  $y \notin f(y)$ , in other words if  $y \notin B$ , which is a contradiction. ■

One can legitimately wonder if the cardinality of the sets is always comparable: given two sets  $A, B$ , is there always an injection of one into the other? The answer to this question again depends on the *axiom of choice*. We will talk about it again in Section 9-4 as well as in Section 27-4.1.

## 5. Non-computable reals

Cantor's theorem will provide us our first argument for the existence of non-computable real numbers. We use for the following proposition and corollary the — for the moment informal — notions of “computer program” and “computable real”. They will both be precisely defined later in this book; consider for the moment that a real is computable if there is a computer program which enumerates in order the infinite list of its decimals.

**Proposition 5.1.** The set of computer programs is countable. ★

PROOF. A computer program (written in any programming language whatsoever) takes the form of a finite sequence of characters, where each character belongs to a finite alphabet (say the set of characters in the ASCII table). Let us show that there exists a bijection from  $\mathbb{N}$  to the set of finite sequences of ASCII characters. For that, we define an infinite list of all the finite sequences of ASCII characters: we list first all the sequences comprising exactly one ASCII character, then all the sequences comprising two ASCII characters, then all those comprising three, etc. It is clear that any finite sequence of ASCII characters appears somewhere in our infinite list.

It is now sufficient to remove from our list the finite sequences which do not correspond to a valid program. We then define the element  $f(n)$  as being the  $n$ -th element of the list resulting from this operation. It is clear that  $f$  is a bijection. In particular, the set of computer programs can be counted. ■

### Corollary 5.2

*There are real numbers that cannot be computed.*

PROOF. Let  $P$  be the set of computer programs. Suppose that every real is computed by an element of  $P$ . Let  $p_1$  be the first computer program computing a real number. Here “first” means first according to the order obtained via the bijection between the computer programs and  $\mathbb{N}$ . Suppose that  $p_1, \dots, p_n$  are defined and all compute a different real. Let  $p_{n+1}$  be the first computer program computing a real number different from

those computed by  $p_1, \dots, p_n$ . We define by induction in this way the sequence  $(p_n)_{n \in \mathbb{N}}$ .

We then obtain a bijection between  $\mathbb{R}$  and an infinite subset of  $P$ . As  $P$  is countable, this infinite subset is also countable, which gives a bijection between  $\mathbb{N}$  and  $\mathbb{R}$ , and hence a contradiction. ■

The proof of the corollary 5.2 is non-constructive: we show the existence of non-computable numbers without giving a precise example. This will obviously be done over and over again in the rest of this work, through a detailed study of different types of non-computable numbers. Before we get down to it, let's mention two or three important notions concerning Cantor's study of infinity following its discovery.

## 6. Cantor space

Cantor has shown that there is no greatest infinity. Among all the possible infinities, the smallest is “countably infinite”. Another infinity is of great interest, namely, that of real numbers:

**Definition 6.1.** A set  $A$  is said to have *the power of the continuum* if  $|A| = |\mathbb{R}|$ . ◇

The power of the continuum therefore characterizes the infinity of reals. Cantor conjectured that there was no infinity strictly between  $|\mathbb{N}|$  and  $|\mathbb{R}|$ , but without succeeding in proving it. His conjecture known as the *continuum hypothesis* will be considered for nearly a century as one of the most important mathematical questions. Gödel will show in 1938 [73] that it is not possible to demonstrate that the continuum hypothesis is false, and Cohen will put an end to the question in 1963 [36] by showing that it is not possible nor to show that the continuum hypothesis is true: it is a question independent of the rest of mathematics, which can be considered true or false without introducing any contradiction. We will discuss this in more detail in Section 9-4.

Among the sets having the power of the continuum, we will pay particular attention to the set of infinite sequences of 0 and 1, which will constitute the main part of our playing field throughout this work. By “infinite sequences of 0 and 1”, we mean sequences indexed by integers, that is to say sequences of the form  $x_0x_1x_2x_3\dots$  where each  $x_i \in \{0, 1\}$ .

**Definition 6.2.** We call *Cantor space* the set of infinite sequences of 0

and 1, we denote it  $2^{\mathbb{N}}$ , and its elements will be denoted by uppercase letters, in general  $A, B, C, X, Y, Z$ .  $\diamond$

Cantor space has the power of the continuum.

**Proposition 6.3.** We have:  $|2^{\mathbb{N}}| = |[0, 1]| = |\mathbb{R}|$ .  $\star$

PROOF. Let us first show  $|[0, 1]| = |\mathbb{R}|$ . The identity function is an injection of  $[0, 1]$  into  $\mathbb{R}$ . We can easily verify that the function

$$f(x) = \begin{cases} 1/2 + 1/(x+2) & \text{if } x \geq 0 \\ 1/2 - 1/(|x|+2) & \text{if } x < 0 \end{cases}$$

is an injection of  $\mathbb{R}$  into  $[0, 1]$ . By the Cantor-Bernstein theorem (see Theorem 2.3), we therefore have  $|[0, 1]| = |\mathbb{R}|$ .

Let us now show  $|2^{\mathbb{N}}| = |[0, 1]|$ . To define an injection of  $2^{\mathbb{N}}$  into  $[0, 1]$ , one should be careful: some real numbers have two possible binary expansions, thus  $1.00000\ldots = 0.11111\ldots$ , where  $0.11111\ldots$  is the decimal number  $0.99999\ldots$  written in binary. It is the same for any real number whose binary expansion ends with an infinity of consecutive 0: this one then has an equivalent binary expansion which ends with an infinity of consecutive 1. We will therefore define the injection  $f : 2^{\mathbb{N}} \rightarrow \mathbb{R}$  which to  $X$  associates the real number of  $[0, 1]$  whose *ternary* expansion, that is to say in base 3, consists of the bits of  $X$ . The use of the ternary representation makes it possible to circumvent these problems of equality and thus to make the function  $f$  injective.

The injection  $g : [0, 1] \rightarrow 2^{\mathbb{N}}$  is defined by associating to any real  $r \in [0, 1]$  its binary expansion  $R$ . When there are several representations of the same real number, we will choose by convention the one ending with an infinity of 0. By the Cantor-Bernstein theorem, we therefore have  $|2^{\mathbb{N}}| = |[0, 1]|$ . ■

### Notation

Given  $X \in 2^{\mathbb{N}}$  and  $n \in \mathbb{N}$ , we denote by  $X(n)$  the  $n$ -th element of the sequence  $X$  which we will also call the  $n$ -th *bit* of  $X$ . Thus, if the sequence  $X$  is written  $x_0x_1x_2\ldots$ , then  $X(0) = x_0$ ,  $X(1) = x_1$ ,  $X(2) = x_2$ , ...

The fact that some real numbers have two possible binary representations make  $|2^{\mathbb{N}}|$  and  $|[0, 1]|$  topologically different spaces. For example, for any  $x, y \in \mathbb{R}$  with  $x < y$ , there is a real  $z$  strictly between the two. On the other hand, if we provide  $2^{\mathbb{N}}$  with the lexicographic order  $<_{lex}$  defined

by

$$X <_{lex} Y \text{ if } X(n) < Y(n),$$

where  $n$  is the smallest integer such that  $X(n) \neq Y(n)$ , then the infinite sequences

$$X = 0111111\dots \text{ and } Y = 100000\dots$$

have no element strictly between them for this order.

This difference is anecdotal from the point of view of the computational complexity of the elements, and we will sometimes speak of real or of infinite binary sequences indistinctly. The difference is however important for the development of Computability Theory in topological spaces other than  $2^{\mathbb{N}}$ . We will discuss this briefly in Section 22-4.2.

Let us see now that there exists, on the other hand, a very natural bijection between the set  $\mathcal{P}(\mathbb{N})$  —the power set of  $\mathbb{N}$ — and Cantor space. Thus,  $\mathcal{P}(\mathbb{N})$  has the power of continuum.

**Proposition 6.4.** We have:  $|2^{\mathbb{N}}| = |\mathcal{P}(\mathbb{N})|$ . ★

PROOF. Let  $f : 2^{\mathbb{N}} \rightarrow \mathcal{P}(\mathbb{N})$  be the function which to  $X \in 2^{\mathbb{N}}$  associates the set  $Y = \{n \in \mathbb{N} : X(n) = 1\}$ . The function  $f$  is clearly a bijection. ■

The bijection between the set  $\mathcal{P}(\mathbb{N})$  of the parts of  $\mathbb{N}$  and Cantor space  $2^{\mathbb{N}}$  is so elementary that we can consider  $\mathcal{P}(\mathbb{N})$  and  $2^{\mathbb{N}}$  as two representations of the same mathematical concept. In the following, we will speak without distinction of a subset of  $\mathbb{N}$  or of an infinite sequence of 0 and of 1, and we will use the same notation  $2^{\mathbb{N}}$  to denote the set of these elements. Thus, given  $X \in 2^{\mathbb{N}}$  seen as a sequence, we will have  $X(n) = 1$  iff  $n$  belongs to  $X$  seen as a set.

### Digression

The name “Cantor space” undoubtedly comes from the eponymous concept *Cantor's triadic set*. We define  $A_0 = [0, 1]$ , then  $A_1$  is  $A_0$  minus its central third:

$$A_1 = [0, 1/3] \cup [2/3, 1].$$

Then  $A_2$  is  $A_1$  minus the middle third of each of its intervals:

$$A_2 = [0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1].$$

In general, to go from  $A_n$  to  $A_{n+1}$ , we remove the middle third of each of the intervals of  $A_n$ . The triadic set of Cantor is ultimately the result of the application of this operation, that is to say the set  $\bigcap_{n \in \mathbb{N}} A_n$ .

Cantor space  $2^{\mathbb{N}}$  defined above, is topologically equivalent to the triadic

Cantor set. In particular, each point of  $\bigcap_{n \in \mathbb{N}} A_n$  can be described as a sequence of 0 and 1 as follows: the  $n$ -th bit of a point corresponds to determining whether it is to the right or to the left of the  $n$ -th third which is subtracted from the current interval. We can also see Cantor space as the set of reals of  $[0, 1]$  whose ternary representation avoids the number 1.



Part I

**Classical Computability  
Theory**



# Chapter 3

## Foundations of computability

Our goal is to conduct a mathematical study of computability. To do this, it is customary to define mathematically what is meant by *computable fonction*. This quest for a formal definition capturing this epistemological concept was the genesis of Computability Theory, and resulted in what is nowadays called the *Church-Turing thesis*. This thesis states that any computable process can be executed with a *Turing machine*, a computational model imagined by Alan Turing in 1936 and which can be considered as a precursor of modern computers. Other approaches than that of Turing machines allow us to capture the notion of computable function, among which we shall cite the general recursive functions and  $\lambda$ -calculus. These different models will be presented in more detail in the interlude on the Church-Turing thesis (see Chapter 6), and we will adopt for this chapter a less formal approach.

### 1. Computable functions

In Computability Theory, the formalism of Turing machines tends to serve as a reference model, not only for historical reasons — Turing was the first to convince the community that his model captured all the computable processes — but also because this model highlights the notion of atomic stage of computation, which opens the door to Complexity Theory. It is customary to begin the study of Computability Theory by that of its computational models, and to prove their equivalence, in order to be convinced of the robustness of the notion of computable function and of the validity of the definitions. This is a fairly long and tedious development. So it is

easy to be put off by this step at the end of which we believe too easily — wrongly — that Computability Theory comes down to complex and boring coding techniques.

We therefore made the choice to break with tradition, and defer the definition and the mathematical study of computational models to Chapter 6, in order to facilitate the first contact with Computability Theory, and to access its fundamental concepts more directly. The advent of computers and the democratization of programming education have firmly anchored the notion of algorithms in scientific culture. We will therefore adopt the following informal definition.

**Definition 1.1.** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is *computable* if it can be defined by an algorithm, or in other words programmed in a modern programming language.  $\diamond$

It follows from our educational choice that the proofs of our first theorems will largely appeal to the intuition of the properties expected of a computable function. These are however theorems in their own right, in the sense that it is possible to prove them from the formal definitions of Chapter 6.

### Integers in Computer Science

We are mainly interested in the functions from  $\mathbb{N}$  to  $\mathbb{N}$ . In most standard programming languages, integers are bounded. For our theoretical definition, it is important to take into account *all the integers*.

**Algorithms.** In order to agree on what is meant by a modern programming language or algorithm, let us list its main aspects, which will be formally taken up in Section 6-3 for our definition of structured programs on the model of register machines.

- (1) The language must be able to handle integers, using constants, integer variables, and the usual arithmetic operations, namely addition, subtraction, multiplication, and integer division. The reader will be able to add other types to it, such as strings or floating point numbers, without this changing the computational power.
- (2) The language must be able to handle Boolean expressions, and perform comparison operations on integers.
- (3) The language must contain the usual control structures, namely conditional instructions of type “if ... then ... else ...” and loops “for ” and “while”, which repeat as long as a certain condition is true.

- (4) We will assume that the machine's memory is unbounded, and that it can be use as much as necessary (especially to read its input, which can be an arbitrarily large integer).

### Data types

Point (1) emphasizes that adding data types other than the integer type is unnecessary. This is true on the condition of course of considering that our integers can be arbitrarily large (for example to encode large strings of characters), which will always be the case by convention. The reader will be able to find an example of encoding of arrays by integers in Proposition 6-3.26.

### Unbounded memory

One might be surprised by point (4) above, which allows unbounded memory. Let us insist on the fact that one does not allow oneself an infinite memory for all that: a computation which ends only carried out a finite number of operations and therefore could only use a finite amount of memory. Simply, we do not deal in Computability Theory with the spatial complexity of the algorithms.

**Examples.** Before starting the formal study of computable functions and their properties, let's list some examples of computable functions, in order to begin to form an intuition.

- (1) The usual arithmetic operations can be computed. In particular, addition, multiplication, subtraction and integer division are computable.
- (2) The function which associates the  $n$ -th prime number with  $n$  is computable, as is the decomposition of a number into its prime factors.
- (3) The function which, taking as parameter a list of city positions, returns one of the shortest paths passing through all these cities only once, is computable<sup>1</sup>.

Conversely, there are, as we will see, many non-computable functions. However, while it suffices to give an algorithm to show that a function is computable, showing that a function is not computable often requires more elaborate reasoning, because it is not enough that the function does not have any known algorithm; it must be shown that it is theoretically impossible to program it. Each of the following statements is therefore a theorem in its own right.

---

<sup>1</sup>This is the famous *travelling salesman problem*.

- (1) The function which takes as input a computer program (coded by an integer or a string), and decides if its execution will one day halt, is not computable (see Theorem 7.8).
- (2) The function which takes as input a formula in the language of arithmetics (for example, “ $\forall x \forall y \forall z x^3 + y^3 \neq z^3$ ”), and returns 1 if there is a mathematical proof of this formula in the axiomatic system of arithmetic, and 0 otherwise, is not computable (see Theorem 9-3.9).
- (3) The function which takes as input a Diophantine equation, ie, an equation with integer coefficients — par exemple  $3x^2 + 2xy + 4y^2 = 0$  — and decides if this equation admits an integer solution, is not computable (see Theorem 12-1.2).

**Partial functions.** As explained in the frame “Integers in Computer Sciences” above, we will generally restrict ourselves to programs taking an integer as a parameter, and returning another integer if the computation ends. It is essential to note that the functions generated by the computer programs are *partial*, in the sense that the computation can never halt on some of its inputs. This bias is due to control structures of type “while” whose termination condition may never be satisfied, as shown in the following example, which computes — in the worse possible manners — the square root of an integer:

```
function Root (n){
  r = 0;
  while (r * r ≠ n){
    r = r + 1;
  }
  return r;
}
```

If  $n$  is not the square of an integer, the loop **while** will execute ad infinitum, and the program will never return a value. When the program does not halt on an input  $n$ , it is considered that the function from  $\mathbb{N}$  to  $\mathbb{N}$  which is associated with it is not defined on  $n$ . The domain of definition of the partial function associated with a program is therefore the set of inputs on which it halts. Functions defined by programs are called *partial computable functions*. When the program halts on all its inputs, the generated function is then called *total computable function*, or quite simply *computable fonction*.

### Notation

If  $f$  and  $g$  are two partial functions, we will denote by  $f(x) = g(x)$  to signify that either  $f$  and  $g$  are both defined and return the same value on  $x$ , or neither  $f$  nor  $g$  are defined on  $x$ .

**Codes.** In our mathematical study, we will represent computer programs by integers, assuming a fixed coding function. By *computer programs*, we must understand here a finite sequence of characters — supposed to have meaning in a programming language chosen beforehand. Concretely, we will say that an integer  $e$  *codes* for a program  $P$  if  $e$  is the integer encoded by the binary representation of the string corresponding to the program  $P$ . In particular, this coding is injective and intuitively computable. Note that one could imagine many other possible encodings. We will see another in detail in the proof of Theorem 6-3.27. In the meantime, this one will be suitable.

### Notation

We denote by  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  the partial function defined by the computer program of code  $e$ .

We will assume that any integer  $e$  codes for a valid program. In practice, in all programming languages, there are strings corresponding to illformed programs. Some integers  $e$  encode unintelligible strings of characters. If this is the case, we can then realize it (this corresponds to having a syntax error when we try to compile a program) and consider that  $e$  then corresponds to a program which never halts. Then  $\Phi_e$  is the nowhere-defined function.

### Notation

Let  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  be a partial computable function and  $x \in \mathbb{N}$  an integer. We will write  $\Phi_e(x) \downarrow$  if the program encoded by  $e$  halts on the input  $x$  (necessarily after a finite number of computation steps) and returns an integer result. In the opposite case, we will write  $\Phi_e(x) \uparrow$ .

The domain of definition of the partial computable function  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  is therefore  $\{x \in \mathbb{N} : \Phi_e(x) \downarrow\}$ . If  $\Phi_e(x) \downarrow$ , we will sometimes write  $\Phi_e(x) \downarrow = y$  to mean that the code program  $e$  halts on the input  $x$  and returns the integer  $y$ . Conversely, we will sometimes write  $\Phi_e(x) \uparrow \neq y$  to mean the opposite, ie  $\Phi_e(x) \uparrow \vee \Phi_e(x) \downarrow \neq y$ .

**Computation time.** Any programming language comes with a notion of execution step and computation time. An execution step is an atomic operation of the language, indecomposable into sub-steps. It corresponds to an elementary instruction of the language. The notion of execution

step induces that of computation time, which is defined by the number of execution steps carried out since the launching of computation. When a program halts on an input, its computation time is finite.

### Notation

Let  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  be a partial computable function and  $x, t \in \mathbb{N}$  two integers. We will write  $\Phi_e(x)[t] \downarrow$  if the program encoded by  $e$  halts *before  $t$  computation steps*. In the opposite case, we will write  $\Phi_e(x)[t] \uparrow$ .

Note that  $\Phi_e(x) \downarrow$  iff there exists a computation time  $t$  such that  $\Phi_e(x)[t] \downarrow$ . Furthermore, if  $\Phi_e(x)[t] \downarrow$ , then  $\Phi_e(x)[s] \downarrow$  for all  $s \geq t$ .

### Remark

We will have to handle computable functions with several parameters. For a fixed integer  $n \in \mathbb{N}^*$ , we will denote as above by  $\Phi_e : \mathbb{N}^n \rightarrow \mathbb{N}$  the partial function with  $n$  integer parameters encoded by  $e$  and we will write  $\Phi_e(x_1, \dots, x_n) \downarrow = y$  if the code program  $e$  halts on the inputs  $x_1, \dots, x_n$  and returns the integer  $y$ .

## 2. Computable sets

The adjective “computable” naturally applies to functions; indeed as described above, any integer  $e$  codes for a program to which a partial computable function corresponds. Most often, however, when we speak of computable functions, we will consider that they are total functions.

**Definition 2.1.** Let  $n \in \mathbb{N}^*$ . A function  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  is *computable* if there exists a program code  $e$  such that, for all  $x_1, \dots, x_n$ , we have

$$\Phi_e(x_1, \dots, x_n) \downarrow = f(x_1, \dots, x_n)$$

Computability can also be used to measure the descriptive complexity of countable mathematical objects, and in particular that of sets of integers. Intuitively, a set of integers  $E \subseteq \mathbb{N}$  is computable if it can be described by a computable process. A set being totally specified by its elements, it is computable if its characteristic function is computable.

**Definition 2.2.** Let  $n \in \mathbb{N}^*$ . A set  $A \subseteq \mathbb{N}^n$  is *computable* if there exists a program code  $e$  such that, for any  $x_1, \dots, x_n$ , we have

- $\Phi_e(x_1, \dots, x_n) \downarrow = 1$  iff  $(x_1, \dots, x_n) \in A$ .

- $\Phi_e(x_1, \dots, x_n) \downarrow = 0$  iff  $(x_1, \dots, x_n) \notin A$ .

◇

We will sometimes call *predicates* the subsets of  $\mathbb{N}^n$ , then using the notation  $A(x_1, \dots, x_n)$  to mean  $(x_1, \dots, x_n) \in A$ . The term predicate comes from the view of  $A \subseteq \mathbb{N}^n$  not as a set, but as a property of  $n$ -tuples of integers. Thus,  $A(x_1, \dots, x_n)$  means that the  $n$ -tuple  $(x_1, \dots, x_n)$  has the property  $A$ . Conversely,  $\neg A(x_1, \dots, x_n)$  means that the  $n$ -tuple  $(x_1, \dots, x_n)$  does not have the  $A$  property.

**Exercise 2.3.** Show that the pairing bijection defined in Proposition 2-3.5 and Exercise 2-3.7, which to  $(a, b) \in \mathbb{N}^2$  associates  $\langle a, b \rangle \in \mathbb{N}$ , is computable. Show that the inverse functions  $\pi_0, \pi_1$  such that  $a = \pi_0(\langle a, b \rangle)$  and  $b = \pi_1(\langle a, b \rangle)$  are also computable. ◇

**Exercise 2.4.** Let  $A \subseteq \mathbb{N}^2$  be a computable predicate. Show that the sets:

- (1)  $\{(x, y) \in \mathbb{N}^2 : \forall z < y (x, z) \in A\}$
- (2)  $\{(x, y) \in \mathbb{N}^2 : \exists z < y (x, z) \in A\}$

are also computable. ◇

#### Recursive set

Historically, computable sets were called *recursive* due to the computation paradigm of general recursive functions in which definitions by induction play a dominant role (see Chapter 6). Gradually, with the improvement of our understanding of the concept of computation, the terminology of the field has evolved. One can however regularly see the terms of *Recursion Theory* and of *recursive set* to speak about Computability Theory and computable set.

### 3. Universal program

The computational model devised by Turing in 1936 consists in defining a machine for each computable function. If we compare a Turing machine to a physical device, it consists, for each task that we want to accomplish, to create a robot performing that specific task.

We are now going to state a first fundamental theorem of Computability Theory and proved by Turing in his original article: the existence of a *universal* Turing machine, capable of simulating all other Turing machines.

This work is sometimes considered a precursor of von Neumann's architecture<sup>2</sup>, one aspect of which is the storage of programs in memory. In modern Computer Science, this theorem can be seen as stating the existence of a *universal* computer program, allowing all other computer programs to be simulated.

**Theorem 3.1**

*Let  $n \in \mathbb{N}^*$ . There exists a computer program code  $e$  for which  $\Phi_e : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  is such, that for all  $a, x_1, \dots, x_n$ , we have*

- $\Phi_e(a, x_1, \dots, x_n) \uparrow$  iff  $\Phi_a(x_1, \dots, x_n) \uparrow$ ;
- $\Phi_e(a, x_1, \dots, x_n) \downarrow = y$  iff  $\Phi_a(x_1, \dots, x_n) \downarrow = y$ .

In practice, a universal program exists very concretely in many languages. For example, in Java, it is simply the virtual machine that compiles and runs any program written in Java. For languages that do not use a virtual machine, a universal program will simply be an interpreter which decomposes the program which it receives into a sequence of instructions, and executes it step by step. While such a program is of course complex to design, there should be no doubt for the programmer that this is something very possible: it is simply a virtual machine. The reader can consult the proof of Theorem 6-3.27, which demonstrates the existence of a universal program for the specific computational model of register machines.

The existence of such a program allows us to define functions which perform manipulations on codes before executing them, or dynamically execute codes passed as parameters. For example,

$$f(x, y) = \Phi_{x+1}(y + 2)$$

is a valid definition, because it is equivalent to  $f(x, y) = \Phi_e(x + 1, y + 2)$ , where  $\Phi_e$  is the universal program.

## 4. SMN theorem

The SMN theorem is our first theorem on the manipulation of computer program codes. It is not conceptually difficult. Let  $\Phi_e : \mathbb{N}^{m+n} \rightarrow \mathbb{N}$  be the function of code  $e$ . Then, given  $x_1, \dots, x_m$ , we can computably transform our code  $e$  into a code  $a$  such that the computation  $\Phi_a(y_1, \dots, y_n)$  gives the same result as the computation

$$\Phi_e(x_1, \dots, x_m, y_1, \dots, y_n)$$

---

<sup>2</sup>Which is almost always the one in use today.

The important point is that the transformation of  $e$  into  $a$  is computable as a function of  $e$  and  $x_1, \dots, x_m$ .

**Theorem 4.1 (SMN theorem)**

For all  $n, m \in \mathbb{N}^*$ , there exists a total computable function

$$S_n^m : \mathbb{N}^{m+1} \rightarrow \mathbb{N},$$

such that for all  $e, x_1, \dots, x_m, y_1, \dots, y_n$ ,

$$\Phi_{S_n^m(e, x_1, \dots, x_m)}(y_1, \dots, y_n) = \Phi_e(x_1, \dots, x_m, y_1, \dots, y_n).$$

PROOF. Let us describe the function  $S_n^m$ . Given  $e, x_1, \dots, x_m$ , decode  $e$  to get the program  $P_e$ . Computably modify the program  $P_e$  to add “hard-coded” instructions assigning the values  $x_1, \dots, x_m$  to the corresponding variables, then compute the code of the new program. For example, if the program  $\Phi_e : \mathbb{N}^4 \rightarrow \mathbb{N}$  is

```
function MyProgram (x1, x2, x3, x4){
    // CODE
}
```

The program  $\Phi_{S_2^2(e, 5, 3)}$  matches the string

```
function MyProgram2 (x3, x4){
    x1 = 5;
    x2 = 3;
    // CODE
}
```

■

The SMN theorem will be used from now on without calling on it explicitly, with sentences like “for every  $x$ , let  $e_x$  be the code of the program which takes  $y$  as input, and does ... [something which depends on  $x$  and  $y$ ]”, it being understood then that the process to obtain  $e$  from  $x$  is computable. We also say in this case that the process is *uniform* in  $x$ .

**Acceptable coding**

The universal program existence theorem and the SMN theorem are not valid for all coding functions. Let us recall the one we use here: each program  $P$  is encoded by the integer corresponding to the binary representation of the string which contains  $P$ .

Rogers [182] proved that any coding function satisfying the Universal Program Theorem and the SMN Theorem was the result of a computable

permutation of this canonical coding. However, there are other codings of partial computable functions which do not satisfy these theorems. In particular, Friedberg [63] defined a computable encoding of all partial functions computable without repetition. This is of course not the case for our coding, for which the same partial function has an infinity of different codes. This is what we are about to do with Lemma 5.1.

## 5. Padding lemma

The padding lemma is useful from time to time and indicates that for any computer program  $e$ , there exists an infinite number of equivalent programs, the code of which can also be computed from  $e$ : it suffices to add instructions that are useless.

**Lemma 5.1 (Padding lemma).** There exists a total computable function  $h : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that for all  $e, n \in \mathbb{N}$ , we have  $h(e, n) \geq n$ , and  $\Phi_{h(e,n)} = \Phi_e$ . ★

PROOF. Given the code  $e$  and an integer  $n$ , decode  $e$  to obtain the program  $P_e$ . Add  $n$  unnecessary instructions to  $P_e$ , then encode the new program to an integer  $i$ .

For example, if the language has an instruction `skip` which does nothing, then it suffices to add to the program a sequence of instructions `skip` as follows:

```
function MyProgram (x){
    // CODE
    skip;
    skip;
    ...
}
```

■

Note that  $\Phi_{h(e,n)}$  and  $\Phi_e$  are equal as mathematical functions, but have different computer codes.

## 6. Kleene's fixed point theorem

Kleene's fixed point theorem, also known as *recursion theorem*, is much more subtle than the SMN theorem. Stephen Cole Kleene is considered

with Kurt Gödel, Alan Turing, Emil Post and Alonzo Church, his teacher, as one of the founders of Computability Theory. He formalizes with Post the notion of *degree of unsolvability*, which we will call later *Turing degree* and which will be precisely defined in Chapter 4.

Among his most remarkable works, figure the definition and the study of hyperarithmetic sets and computable ordinals [113], which we will see in Part IV and for which we will need to create computer programs which can *access their own code*. The notion may seem doubtful: how can we use in the definition of an object  $A$  the object  $A$  itself? Self-references often lead to paradoxes. However, we will see that in the case of computer programs, access to its own code is quite valid, and even very useful in some cases. We will see a remarkable example of the use of the fixed point theorem in the proof of Theorem 19-1.7. Let's see more precisely what it is. The theorem states that for any function that modifies programs, there exists a program whose behavior is not modified by the function.



Stephen Cole Kleene, 1909–1994

### Theorem 6.2

For any total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , there exists  $e \in \mathbb{N}$  such that for any  $n$ ,

$$\Phi_{f(e)}(n) = \Phi_e(n).$$

Before moving on to the proof, let's see how this mysterious claim allows you to create programs with access to their own code. Suppose that a program  $M$  uses a variable `var` that was initialized to a certain value at the start of its execution. We can then easily define a total computable function  $f$  which takes an integer  $n$  as a parameter, and returns the code of the program  $M$ , which begins its execution with `var` initialized to  $n$ . According to the fixed point theorem, there is a value  $e$  such that the programs of code  $e$  and  $f(e)$  have the same behavior. So  $e$  is a program code equivalent to that of the  $M$  program running with the variable `var` initialized to  $e$ : there is a version of  $M$  that can access its own code. Here is, more formally, what we have just stated.

**Corollary 6.3**

For any partial computable function  $g : \mathbb{N}^2 \rightarrow \mathbb{N}$ , there exists  $e \in \mathbb{N}$  such that for any  $n$ , we have

$$\Phi_e(n) = g(e, n)$$

PROOF. Let  $i$  be such that  $\Phi_i(x, n) = g(x, n)$  for all  $x, n$ . By the SMN theorem (see Theorem 4.1), for all  $x, n$ , we have  $\Phi_{S_2^1(i, x)}(n) = \Phi_i(x, n)$ . Let  $f$  be the function defined by  $f(x) = S_2^1(i, x)$ . By the fixed point theorem (see Theorem 6.2), there exists  $e$  such that, for all  $n$ ,  $\Phi_{f(e)}(n) = \Phi_e(n)$ . In particular,  $\Phi_e(n) = \Phi_{f(e)}(n) = \Phi_i(e, n) = g(e, n)$ . ■

The proof of Theorem 6.2, although concise, is somewhat obscure; this is why we provide beforehand a piece of code whose objective is to give the programmer an intuition of how to write a program with access to its own code. The example is given here in the JavaScript language.

In the following, the two ellipses must each contain the same code, no matter which one. In JavaScript, “backquotes” delimit a string over several lines. Function `replace` will replace the first occurrence of ‘#’ with the content of variable `v`.

```
function fct (){
  let v = `
function fct (){
  let v = '#'
  v = v.replace ('#', v)
  ... // my code
} `
  v = v.replace ('#', v)
  ... // my code
}
```

The execution of this program will be done with the variable `v` having for content the program itself, except that the “backquotes” are then transformed into simple “quotes”. It is of course possible to make the variable `v` contain *exactly* the program, but that would make the example much less understandable. Aiming to give an intuition and not a proof, we have kept it that way. Let us now proceed to the formal proof of our theorem, within the framework of the notations and principles that we have defined so far.

PROOF OF THEOREM 6.2. Let  $a$  be the code of a one-parameter machine, which on input  $n$  returns the code of a one-parameter machine  $m$ , which performs the following operations.

- (1) It starts the computation of  $f(\Phi_n(n))$ .
- (2) If we have  $f(\Phi_n(n)) \downarrow$ , then it returns the result of the computation of the code machine  $f(\Phi_n(n))$  on the input  $m$  (and otherwise does not halt).

Formally,  $a$  is such that, for all  $n, m \in \mathbb{N}$ ,

$$\Phi_{\Phi_a(n)}(m) = \Phi_{f(\Phi_n(n))}(m).$$

There is a slight abuse of notation here: if  $\Phi_n(n) \uparrow$ , then  $m \mapsto \Phi_{f(\Phi_n(n))}(m)$  denotes the nowhere defined function. Note that  $\Phi_a$  is a total function: for any  $n$ , in the computation of  $\Phi_a(n)$ , we do not try to do the steps (1) and (2), but only to compute the code of a machine that makes them

The proof that such a code  $a$  exists is given by the SMN theorem, here is how. The function  $(n, m) \mapsto \Phi_{f(\Phi_n(n))}(m)$  is computable (possibly partial), and there is therefore a code  $b$  such that

$$\Phi_b(n, m) = \Phi_{f(\Phi_n(n))}(m).$$

According to the SMN theorem, there is a total computable function  $s$  such that  $\Phi_{s(b,n)}(m) = \Phi_{f(\Phi_n(n))}(m)$ . As  $s$  is total computable, there is a code  $a$  such that  $\Phi_a(n) = s(b, n)$ .

The fixed point will then be  $\Phi_a(a)$ . Indeed, we have

$$\forall m, \quad \Phi_{\Phi_a(a)}(m) = \Phi_{f(\Phi_a(a))}(m).$$

This concludes the proof. ■

### Fixed point theorem and paradox

As explained above, Kleene's fixed point theorem makes it possible to design *self-referential* programs, i.e., programs which can read their code during execution, and adapt their behavior accordingly. The ability to self-refer is often a source of paradoxes.

The Barber's Paradox, for example, tells the story of a philanthropic barber who decided to shave all the people who did not shave themselves. Should he shave himself? Paradox ... In mathematics, Russel's paradox follows the same pattern: let  $E$  be the set of all sets which do not belong to themselves. For example,  $\mathbb{N}$  is not an integer, so  $\mathbb{N} \notin \mathbb{N}$ , therefore  $\mathbb{N} \in E$ . The problematic question is then "Does  $E \in E$ ?"

Why does Kleene's fixed point theorem not generate a paradox? If we try to follow the same scheme as the two previous paradoxes, we will define a function  $\Phi_e$  which knows its code  $e$ , and therefore can decide, for any input  $n$ , to execute  $\Phi_e(n)$  and return a different value than the one returned by  $\Phi_e(n)$ . We would therefore have  $\Phi_e(n) \neq \Phi_e(n)$ .

The solution comes from the partiality of the functions: indeed,  $\Phi_e$  will simply not be defined in  $n$  and will run indefinitely.

The reader wishing to explore the possibilities of the fixed point theorem can tackle the following exercise, the object of which is to prove Rice's theorem, which will be discussed in more detail in Section 5-6.

**Exercise 6.4. ( $\star$ )** Let  $A \subseteq \mathbb{N}$  be such that  $A \neq \mathbb{N}$ ,  $A \neq \emptyset$  and such that  $A$  is computable.

1. Show that there exists a computable function  $f$  such that we have  $x \in A$  iff  $f(x) \notin A$ .
2. Using the fixed point theorem, deduce that there are  $i, j$  such that  $\Phi_i = \Phi_j$ , with  $i \in A$  and  $j \notin A$ .
3. Deduce that there are no computable predicates  $A$  such that  $e \in A$  iff  $e$  is the code of a program which multiplies by 2.
4. Generalize the previous question to show *Rice's theorem*: for all “non-trivial behavior”, it is not possible to compute the set of program codes having this behavior. Formally: let a predicate  $P \subseteq \mathbb{N}$  with  $P \neq \mathbb{N}$  and  $P \neq \emptyset$  such that, for all  $e_1, e_2$  for which  $\Phi_{e_1} = \Phi_{e_2}$ , we have  $e_1 \in P \leftrightarrow e_2 \in P$ . Then,  $P$  is not computable.  $\diamond$

## 7. Computably enumerable sets

Computability places “computable” as the reference computational power. This is the weakest computational notion that this paradigm allows to identify. As we have seen, a set  $E$  is computable if there is a procedure which, given an element, indicates whether this element belongs to  $E$  or not. The procedure should always halt and give a correct answer.

There are, however, a certain number of mathematical problems which can be expressed naturally in the form of sets whose elements are enumerable by a computable procedure, but out of order. These sets are called *computably enumerable*.

For example, let  $E$  be the set of mathematical theorems. There is no procedure which, given a mathematical formula, returns true or false depending on whether this formula is provable or not. However, it is possible to enumerate the theorems, listing all possible strings, testing whether this is a valid proof, and if so, listing the conclusion of the proof (we will discuss this in more detail in Chapter 9).

We are now going to formally define the notion of computably enumerable set in a form which may seem far from the informal definition that we have just given. We will see through Proposition 7.2 that these definitions coincide.

**Definition 7.1.** A set  $A \subseteq \mathbb{N}$  is *computably enumerable* (c.e.) If there exists a program code  $e$  such that  $n \in A \leftrightarrow \Phi_e(n) \downarrow$ , for all  $n \in \mathbb{N}$ .  $\diamond$

Note that computably enumerable sets were historically called *recursively enumerable*. It is still common to see articles using the old terminology, and in particular the abbreviation r. e. instead of c.e.

We can see the machine code  $e$  as a process which enumerates  $A$ : for each integer  $n$ , we look for an integer  $t$  such that  $\Phi_e(n)$  halts in  $t$  computation steps. If such a  $t$  is found, then we enumerate  $n$  in our set. Knowing that there is an infinity of integers, it is necessary to fix an order of execution to carry out only a finite number of computations at each step. The idea is to break it down step by step as follows.

1. Test if  $\Phi(0)[0] \downarrow$ .
2. Test whether  $\Phi(0)[1] \downarrow$  or  $\Phi(1)[1] \downarrow$ .
3. Test whether  $\Phi(0)[2] \downarrow$  or  $\Phi(1)[2] \downarrow$  or  $\Phi(2)[2] \downarrow$ .
4. ...

The first time that we find a step  $t$  such that  $\Phi(n)[t] \downarrow$  for a certain integer  $n$ , we enumerate the latter. Such an enumeration can be seen as a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  where  $f(n)$  is the  $n$ -th element listed in  $A$ . This idea is repeated in the proof of the following proposition.

**Proposition 7.2.** An infinite set  $A$  is computably enumerable iff there exists an injective total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that

$$f(\mathbb{N}) = A.$$

PROOF. Let  $A$  be an infinite computably enumerable set, and let  $e$  be such that  $\Phi_e(n) \downarrow$  iff  $n \in A$ . We define the computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  as follows.

- To compute  $f(0)$ , we look for the smallest  $t$  such that  $\Phi_e(s)[t] \downarrow$  for some  $s \leq t$ , and we then return the smallest such integer  $s \leq t$ .
- To compute  $f(n+1)$ , we first run the computation of  $f(i)$  for all  $i \leq n$ , then we look for the smallest  $t$  such that  $\Phi_e(s)[t] \downarrow$  for some  $s \leq t$  such that  $s$  is different from each  $f(i)$  for  $i \leq n$ , and we then return the smallest such integer  $s \leq t$ .

The process for determining  $f(n)$  is indeed computable, and since our computably enumerable set is infinite, the function  $f$  will halt on all its values. It is then clear that  $f(\mathbb{N}) = A$ .

Suppose now that  $f(\mathbb{N}) = A$  for a total computable function  $f$ . We define the function  $g$  which on  $n$  searches for the smallest  $t$  such that  $f(t) = n$ , and then halts (and otherwise searches indefinitely without ever halting). We have  $g(n) \downarrow$  iff  $\exists t f(t) = n$ . ■

It should be clear to the reader that a computable set is computably enumerable.

**Exercise 7.3.** Show that any computable set is computably enumerable. ◇

We will see that the reverse is not necessarily true: there are computably enumerable sets which are not computable. The following proposition gives more precisely the connection between these two concepts.

**Proposition 7.4.** A set  $A$  is computable iff  $A$  and  $\mathbb{N} \setminus A$  are both computably enumerable. ★

PROOF. Let  $A$  be a computable set. According to Exercise 7.3, it is computably enumerable. Moreover, the set  $\mathbb{N} \setminus A$  is also computable, and therefore computably enumerable.

Suppose now that we have two codes  $e_1, e_2$  such that  $\Phi_{e_1}(n) \downarrow$  iff  $n \in A$  and  $\Phi_{e_2}(n) \downarrow$  iff  $n \in \mathbb{N} \setminus A$ . We define the computable function  $f(n)$  which searches for the smallest  $t$  such that  $\Phi_{e_1}(n)[t] \downarrow$  or such that  $\Phi_{e_2}(n)[t] \downarrow$ , then returns 1 in the first case and 0 in the second. It is clear that the function  $f$  computes the set  $A$ . ■

If a computably enumerable set is not in general computable, it can be approximated by an increasing sequence of uniformly computable sets, in the following sense. We say that a sequence of sets  $A_0, A_1, \dots$  is *uniformly computable* if there exists a computable function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  such that, for all  $x, n$ , we have  $f(x, n) = 1$  iff  $x \in A_n$ .

**Definition 7.5.** A *c.e. approximation* of a set  $A$  is a uniformly computable sequence of sets  $A_0, A_1, \dots$  such that  $A_n \subseteq A_{n+1}$  for all  $n$ , and  $\bigcup_n A_n = A$ . ◇

**Proposition 7.6.** A set  $A$  is computably enumerable if, and only if, it has a c.e. approximation ★

PROOF. Suppose that  $A$  is c.e. By definition, there is a program code  $e$  such that  $x \in A \leftrightarrow \Phi_e(x) \downarrow$ , for all  $x \in \mathbb{N}$ . Let  $A_0, A_1, \dots$  be the uniformly computable sequence of sets defined by  $A_n = \{x : \Phi_e(x)[n] \downarrow\}$ . It is clear that  $A_n \subseteq A_{n+1}$ , because if the machine  $\Phi_e$  halts on an input  $x$  before  $n$  steps, it will halt before  $n+1$  steps.

Suppose now that the set  $A$  has a c.e. approximation, namely,  $A_0, A_1, \dots$ . Let  $\Phi_e$  be the program which, for an input  $x$ , computes  $A_0(x)$ , then  $A_1(x)$ , then  $A_2(x)$ , and so on, until  $A_n(x) = 1$  for some  $n$  and halts there. If  $A_n(x) = 0$  for all  $n$ , then  $\Phi_e(x) \uparrow$ , otherwise  $\Phi_e(x) \downarrow$ . By construction,  $\{x : \Phi_e(x) \downarrow\} = \bigcup_n A_n = A$ . ■

### Notation

Given a c.e. set  $A$ , we denote by  $A[0], A[1], \dots$  a fixed c.e. approximation of  $A$ . In particular,  $A[s]$  is the *approximation of  $A$  at step  $s$* . By convention,  $A[s]$  is a finite set with  $\max A[s] < s$  if  $A[s] \neq \emptyset$ . We will sometimes also use the notation  $A_s$  instead of  $A[s]$ .

As we have said, there are computably enumerable sets which are not computable. The canonical example is known as the “halting problem”.

**Definition 7.7.** We call *halting problem*, which we denote by  $\emptyset'$ , the set:

$$\emptyset' = \{n \in \mathbb{N} : \Phi_n(n) \downarrow\}.$$

◇

Alan Turing demonstrates in 1936 that the halting problem is not computable, using a diagonal argument, like the one introduced by Cantor to demonstrate the non-countability of real numbers. Before giving a mathematical proof, we give an intuition via a little code using Java syntax.

Let us absurdly assume that we have at our disposal a function `boolean Halt(String p, String v)` which returns `true` if the function in the string `p` halts when it is executed with the parameter `v`, and returns `false` otherwise. As usual, if `p` contains a string that does not match a valid function or does not match a function taking a parameter of type `String`, then `Halt` returns `false`. Consider the following program.

```
function Diagonal (x){
  if (Halt (x, x)){
    while (true); //infinite loop
  } else{
    return; //end
  }
}
```

What does the execution of program **Diagonal** return with as parameter a string  $x$  corresponding to the program **Diagonal** itself? We see that we come up with paradox:

- if  $\text{Halt}(x, x)$  returns **true**, then **Diagonal** does not halt on **Diagonal** as a parameter;
- in the opposite case, **Diagonal** halts on **Diagonal** as parameter.

Thus, the function **Halt** does not keep its promises.

**Theorem 7.8**

*The halting set is a computably enumerable set which is not computable.*

PROOF. The partial function  $f : \mathbb{N} \mapsto \Phi_n(n)$  is clearly computable, and we have  $f(n) \downarrow$  iff  $\Phi_n(n) \downarrow$ . By definition,  $\emptyset' = \{n : f(n) \downarrow\}$ . Therefore,  $\emptyset'$  is computably enumerable.

Let us now assume absurdly that  $\emptyset'$  is a computable set. In particular, according to Proposition 7.4, the set  $\mathbb{N} \setminus \emptyset'$  is computably enumerable, and there exists a code  $e$  such that  $n \in \mathbb{N} \setminus \emptyset'$  iff  $\Phi_e(n) \downarrow$ . Then, for all  $n$ ,

$$\Phi_e(n) \downarrow \leftrightarrow n \in \mathbb{N} \setminus \emptyset' \leftrightarrow \Phi_n(n) \uparrow.$$

In particular, for  $n = e$ ,

$$\Phi_e(e) \downarrow \leftrightarrow e \in \mathbb{N} \setminus \emptyset' \leftrightarrow \Phi_e(e) \uparrow,$$

which is a contradiction. The sentence is therefore not computable, and in particular the negation of the sentence is not computably enumerable. ■

We invite the reader to consider the following two exercises, which should not pose any difficulties and which allow us to reflect a little on computably enumerable sets.

**Exercise 7.9. (★)** An infinite set is *computably enumerable in the order* if there exists a total function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(\mathbb{N}) = A$  and such that  $f(n) < f(n+1)$ . Show that if an infinite set  $A$  is computably enumerable in order, then  $A$  is computable. ◇

**Exercise 7.10. (★)** Show that any infinite computable enumerable set contains an infinite computable subset. ◇

**Exercise 7.11. (★★)** Two sets  $A$  and  $B$  are *computably inseparable* if  $A \cap B = \emptyset$  and if no computable set  $C$  allows to separate  $A$  and  $B$ ; in other words, no computable set  $C$  is such that  $A \subseteq C$  and  $C \cap B = \emptyset$ . Show that there are two computably inseparable c.e. sets. ◇

**Exercise 7.12. (★★)** Show that given  $A, B \subseteq \mathbb{N}$  two computable sets, the set  $D_{A,B} = \{x - y : x \geq y \text{ and } x \in A \text{ and } y \in B\}$  is not necessarily computable.

Indication .– Show that for any c.e. set  $C \subseteq \mathbb{N}$ , there exist two computable sets  $A, B \subseteq \mathbb{N}$  such that  $x \in C$  iff  $2^x \in D_{A,B}$ .  $\diamond$

Some people — a small minority — will perhaps have integrated with great ease everything that has been seen so far, to the point no doubt of getting bored a little. This is what the following exercise is aimed at, which should occupy them for a little while ...

**Exercise 7.13. (★★★)** (*Friedberg [63]*). A c.e. set  $X$  is *maximal* if  $\mathbb{N} \setminus X$  is infinite and if any c.e. set  $Y \supseteq X$  is such that  $Y \setminus X$  is finite or such that  $\mathbb{N} \setminus Y$  is finite. Show that there exists a maximal c.e. set.

Indication .– We denote by  $W_e$  the c.e. set given by  $\{n \in \mathbb{N} : \Phi_e(n) \downarrow\}$ . We can start by finding a uniform process which, on each  $e \in \mathbb{N}$ , associates a code  $d$  such that  $\mathbb{N} \setminus W_d$  is infinite, and such that if  $W_d \subseteq W_e$ , then either  $W_e$  is finite, or  $\mathbb{N} \setminus W_e$  is finite.  $\diamond$



# Chapter 4

## Turing degrees

A non-computable set can be seen as an insoluble problem: there is no algorithm making it possible to decide whether or not an integer belongs to this set. One of the goals of Computability Theory is to study and understand the universe of insoluble problems, through different comparisons and classifications. Various tools are introduced for this. The most important of them is the subject of this chapter and finds its genesis in “Systems of logic based on ordinals” [225], the famous article by Alan Turing presenting his thesis work, and will be formalized and studied later by Post [180] then Post and Kleene [117]: given a non-computable set  $X$ , we imagine being able to use it as “oracle” in order to increase the computational power of our machines. We will then say that two sets are in the same *degree of insolubility* or in the same *Turing degree*, if each can be computed with an algorithm using the other as “oracle”.

### 1. Finite strings

Before getting to the heart of the matter, we need to introduce some vocabulary and notation about binary strings. Formally, a binary string is a partial function from  $\mathbb{N}$  to  $\{0, 1\}$  whose domain of definition is an initial segment of  $\mathbb{N}$ . More informally, it is a finite sequence of 0 and 1.

**Definition 1.1.** We denote by  $2^{<\mathbb{N}}$  the set of finite sequences of 0 and 1. The variables  $\sigma, \tau, \rho$  will normally be used to denote elements of  $2^{<\mathbb{N}}$ , which will generally be called *strings*. These sequences will be handled via the following symbols:

- $\epsilon$ : the empty string, of length 0
- $i^n$  for  $i \in \{0, 1\}$ : a sequence of  $n$  bit repetitions  $i$
- $\sigma\tau$  or  $\sigma \hat{\ } \tau$ : the concatenation of  $\sigma$  and  $\tau$
- $\sigma \preceq \tau$ : the string  $\sigma$  is a prefix of  $\tau$ , that is  $\exists \rho$  tel que  $\sigma\rho = \tau$
- $\sigma \prec \tau$ : the string  $\sigma$  is a strict prefix of  $\tau$ , ie  $\exists \rho \neq \epsilon$  such that  $\sigma\rho = \tau$
- $|\sigma|$ : the length of  $\sigma$
- $\sigma(n)$  for  $n < |\sigma|$ : the value of the  $n$ -th bit of  $\sigma$ , starting at 0
- We will say that two strings  $\sigma, \tau$  are *incompatible* if we have neither  $\sigma \preceq \tau$  nor  $\tau \preceq \sigma$  (we will also write  $\sigma \not\preceq \tau$  and  $\tau \not\preceq \sigma$ ). If, on the other hand,  $\sigma \preceq \tau$  or  $\tau \preceq \sigma$ , the two strings  $\sigma$  and  $\tau$  are compatible.

◇

We will sometimes identify a bit  $i \in \{0, 1\}$  with the string of length 1 whose only bit is  $i$ . Thus, we will denote by  $\sigma i$  or  $\sigma \hat{\ } i$  the concatenation of a string  $\sigma$  and a bit  $i$ . According to the previous definitions,  $|\epsilon| = 0$ , and for any non-empty string  $\sigma$ , the first and last bit are respectively  $\sigma(0)$  and  $\sigma(|\sigma| - 1)$ . Chains and infinite sequences can be combined.

**Definition 1.2.** We adopt the following notations for  $\sigma \in 2^{<\mathbb{N}}$  and  $X \in 2^{\mathbb{N}}$ :

- $\sigma X$ : the concatenation of  $\sigma$  and  $X$
- $\sigma \prec X$ : the string  $\sigma$  is a prefix of  $X$ , that is  $\exists Y \in 2^{\mathbb{N}} \ \sigma Y = X$
- $X \upharpoonright_n$  for  $n \geq 0$ : the prefix of  $X$  of size  $n$ .

◇

## 2. Computation with oracle

Intuitively, a computation with an oracle  $X \subseteq \mathbb{N}$  is easy to understand: it is a computation which can at any time use an instruction of the form “does  $n$  belong to  $X$ ?”. This instruction can be compared to a function call, which always returns the correct answer.

If the oracle used is not computable, it is therefore possible to write computer programs which computes, using this oracle, objects which would not be computable otherwise. In particular,  $X$  itself with computable with oracle  $X$ . Our goal now is to study sets of integers in terms of the computational power they provide when used as oracles.

**Definition 2.1.** A *Turing functional* or simply a *functional* is a function computable by an algorithm in a programming language enriched with instructions of the form “does  $n$  belong to the oracle?”.  $\diamond$

A functional therefore has two kinds of parameters: the usual integer parameters, as for computable functions, and one oracle parameter, which is an infinite binary sequence, or equivalently a set of integers, representing the oracle. We will also call *first-order parameters* the integer parameters and *second-order parameter* the oracle parameter. A functional can be seen as a scheme of partial functions, parameterized by the oracle: the same functional “fed in” with a different oracle will yield a different function.

#### Notation

We note  $\Phi_e(X, n)$  or  $\Phi_e^X(n)$  for the result of the computation of the functional  $\Phi_e$  with the oracle  $X$  and on the input  $n$ . We will write in the same way  $\Phi_e(X, n) \downarrow$ ,  $\Phi_e(X, n) \uparrow$ ,  $\Phi_e(X, n)[t] \downarrow$ ,  $\Phi_e(X, n)[t] \uparrow$  to signify that the functional  $\Phi_e$  respectively halts, does not halt, halts in time lower than  $t$ , does not halt in time lower than  $t$ , with the oracle  $X$  and on the input  $n$ .

In order to have a uniform vision, we now suppose that we only work with functionals. It is easy to see that the computable functions are exactly those which are computable by functions using the empty set as oracle (or any other computable set).

**Definition 2.2.** A set  $A$  is said to be  *$X$ -computable* or computable relative to  $X$  if it is computable by a Turing functional using  $X$  as an oracle. A set  $A$  is said to be *computably enumerable relative to  $X$*  if it is the domain of definition of a Turing functional using  $X$  as an oracle.  $\diamond$

The notion of oracle is generalized to functions. With an oracle  $f : \mathbb{N} \rightarrow \mathbb{N}$ , a computation consists in adding the function  $f$  to the primitives of the programming language. Thus, the program can at any time query  $f$  on inputs to know the results. We can therefore speak of an  *$f$ -computable* set if it is computable by a Turing functional using  $f$  as an oracle. Equivalently, a set is  *$f$ -computable* if it is  $G_f$ -computable, where  $G_f \in 2^{\mathbb{N}}$  is an encoding of the graph of the function  $f$  by an element of  $2^{\mathbb{N}}$ , for example with  $\langle n, m \rangle \in G_f$  iff  $f(n) = m$ .

#### Notation

We will also write  *$X$ -c.e.* to mean computably enumerable relative to  $X$ .

**Example 2.3.** Suppose for example that we have an oracle  $X \subseteq \mathbb{N}$  such that the function  $f$  which to  $n$  associates the  $n$ -th element of  $X$  increases fast enough to dominate the halting time of computer programs. Formally:  $\Phi_e(e) \downarrow$  implies  $\Phi_e(e)[f(e)] \downarrow$  for all  $e \in \mathbb{N}$ . It is then easy to create a Turing functional allowing to compute  $\emptyset'$  from  $X$ : to know if  $e \in \emptyset'$ , it suffices to browse  $X$  until you find its  $e$ -th element  $f(e)$ , then to compute  $\Phi_e(e)$  during  $f(e)$  stages of computation. If  $\Phi_e(e)[f(e)] \downarrow$ , then  $e \in \emptyset'$ . Otherwise,  $e \notin \emptyset'$ .

### 3. Relativization of proofs

Most of the computability-theoretic arguments apply to oracle machines by replacing “machine” by “machine with an oracle  $X$ ”. This purely syntactic operation, which we call *relativization* (to an oracle), therefore gives for free a scheme of similar results, parameterized by an oracle  $X$ . Let us take the example of the undecidability of the halting problem, by replacing the notion of computation by that of computation with oracle  $X$ .

#### Theorem 3.1

For any oracle  $\mathbf{X}$ , the set  $Y = \{n : \Phi_n^{\mathbf{X}}(n) \downarrow\}$  is not  $\mathbf{X}$ -computable.

PROOF. The partial function  $f : n \mapsto \Phi_n^{\mathbf{X}}(n)$  is clearly  $\mathbf{X}$ -computable, and we have  $f(n) \downarrow$  iff  $\Phi_n^{\mathbf{X}}(n) \downarrow$ . By definition,  $Y = \{n \in \mathbb{N} : f(n) \downarrow\}$ . Then,  $Y$  is  $\mathbf{X}$ -computably enumerable.

Let us now assume absurdly that the set  $Y$  is  $\mathbf{X}$ -computable. In particular, according to Proposition 3-7.4 relativized to  $\mathbf{X}$ , the set  $\mathbb{N} \setminus Y$  is  $\mathbf{X}$ -computably enumerable, and there exists a code  $e$  such that  $n \in \mathbb{N} \setminus Y$  iff  $\Phi_e^{\mathbf{X}}(n) \downarrow$ . We then have for all  $n$

$$\Phi_e^{\mathbf{X}}(n) \downarrow \leftrightarrow n \in \mathbb{N} \setminus Y \leftrightarrow \Phi_n^{\mathbf{X}}(n) \uparrow.$$

In particular, for  $n = e$ , we have

$$\Phi_e^{\mathbf{X}}(e) \downarrow \leftrightarrow e \in \mathbb{N} \setminus Y \leftrightarrow \Phi_e^{\mathbf{X}}(e) \uparrow,$$

which is a contradiction. Therefore,  $Y$  is not  $\mathbf{X}$ -computable, and in particular the complement of  $Y$  is not  $\mathbf{X}$ -computably enumerable. ■

However, one must be a little cautious when relativizing an argument. Indeed, some definitions mask calls to machines, and their definition must therefore also be relativized. For example, if we define the *halting problem* as the set  $\{n : \Phi_n(n) \downarrow\}$ , the relativization of the statement “The halting problem is not computable” is not “For all  $X$ , the halting problem is

not  $X$ -computable”, but “For all  $X$ , the halting problem relativized to  $X$  is not  $X$ -computable”, where the “halting problem relativized to  $X$ ” is in fact defined as the set  $\{n : \Phi_n^X(n) \downarrow\}$ .

In general, the relativization of a theorem is obtained by adding an oracle  $X$  to each machine, and by replacing *computable* by  *$X$ -computable*. This is for example the case for a naive relativization of the SMN theorem.

**Theorem 3.2 (Relativized SMN theorem - version 1)**

**For any oracle  $X$** , and for all non-zero integers  $n$  and  $m$ , there exists a total  **$X$ -computable** function  $S_n^m : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$  such that for all  $e$ ,

$$\Phi_{S_n^m(e, x_1, \dots, x_m)}^X(y_1, \dots, y_n) = \Phi_e^X(x_1, \dots, x_m, y_1, \dots, y_n).$$

An analysis of the proof of the SMN theorem reveals however that the function  $S_n^m$  does not depend on the oracle  $X$ , because it performs purely syntactic manipulations on the machine. It is therefore possible to formulate a stronger relativized version of the SMN Theorem, where the function  $S_n^m$  is computable, although the previous version is also valid.

**Theorem 3.3 (Relativized SMN theorem - version 2)**

**For any oracle  $X$** , for any  $n, m \in \mathbb{N}^*$  there exists a total **computable** function  $S_n^m : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$  such that for all  $e$ ,

$$\Phi_{S_n^m(e, x_1, \dots, x_m)}^X(y_1, \dots, y_n) = \Phi_e^X(x_1, \dots, x_m, y_1, \dots, y_n).$$

### Digression

The relativization of arguments is an empirical phenomenon which in a way reflects our partial understanding of the notion of calculus. However, and especially in Complexity Theory, proofs not necessarily relativize. The emblematic example is the question of the separation of the complexity classes P and NP. Each of these classes can be relativized to an oracle. Baker, Gill, and Solovay [9] have shown that there are oracles  $X$  for which  $P^X = NP^X$  and others for which  $P^X \neq NP^X$ . In order to solve the question P vs NP, it is then necessary to use an argument which cannot be relativized. This is in Computability Theory something quite unusual, but not necessarily in Complexity Theory, where we have other examples of solved problems that do not relativize. We know for example that the IP and PSPACE classes coincide [195], while being able to produce oracles  $X$  for which  $IP^X \neq PSPACE^X$  [59].

## 4. Use property

Given a computation  $\Phi_e(A, n)$  on an oracle  $A$ , when  $\Phi_e(A, n) \downarrow$ , then the functional  $\Phi_e$  has only used a finite part of the oracle  $A$  to return the result: this follows simply from the fact that a computation is always carried out in a finite number of steps. There is therefore an initial segment  $\sigma \prec A$  such that for any other oracle  $B$  having the same initial segment,  $\Phi_e(A, n)$  and  $\Phi_e(B, n)$  have exactly the same behavior. This leads us to define the notion of computation with finite oracles.

### Notation

Given a finite sequence  $\sigma \in 2^{<\mathbb{N}}$ , we write  $\Phi_e(\sigma, n) \downarrow$  or  $\Phi_e^\sigma(n) \downarrow$  to signify that the computation halts on the input  $n$  and with  $\sigma$  as a partial oracle, where the oracle has not been accessed outside of its domain.

In the previous notation, if the computation needs to access oracle values that exceed the size of  $\sigma$ , then we write  $\Phi_e(\sigma, n) \uparrow$  or  $\Phi_e^\sigma(n) \uparrow$ . The finite part of the oracle  $A$  queried until the computation  $\Phi_e(A, n)$  finishes is called the *use* of the computation.

**Proposition 4.1 (Use property).** Let  $\Phi_e$  be a Turing functional,  $X$  an oracle and  $n \in \mathbb{N}$  an input. Then,  $\Phi_e(X, n) \downarrow$  if, and only if, there exists a finite prefix  $\sigma \prec X$  such that  $\Phi_e(\sigma, n) \downarrow$ . ★

We will see in Section 8-2 that the use property corresponds to the concept of continuity on Cantor space. Despite its conceptual simplicity, the use property plays a primordial role in Computability Theory. For example, we will make a strong use of it (no pun intended) in Section 8 on the finite extension method.

**Definition 4.2.** Given a functional  $\Phi$  and an oracle  $X$ , we denote by  $\text{use}_\Phi^X : \mathbb{N} \rightarrow \mathbb{N}$  the partial  $X$ -computable function which on  $n$  returns the minimum length of the prefix of  $X$  used in the computation of  $\Phi(X, n)$ . ◇

### Remark

According to the use property, any Turing functional  $\Gamma$  can be represented by the c.e. set  $W$  of triples  $(\sigma, x, y)$  such that if  $(\sigma, x, y) \in W$ , then  $\Gamma^\sigma(x) \downarrow = y$ . Such an enumeration satisfies the following consistency property: for all  $(\sigma, x, y) \in W$  and  $(\tau, x, y') \in W$  such that  $\sigma \prec \tau$ , we have  $y = y'$ .

**Exercise 4.3.** Show that if  $Y$  is  $X$ -computable and  $Z$  is  $Y$ -computable, then  $Z$  is  $X$ -computable. ◇

## 5. Turing degrees

Oracle machines induce a notion of relative computability. Informally, a set  $Y$  is  $X$ -computable if  $X$  is at least as powerful as  $Y$ , in the sense that everything that is computable by  $Y$  is also computable by  $X$ . This gives rise to the *Turing reduction*.

**Definition 5.1 (Turing reduction).** Given two sets  $X, Y \subseteq \mathbb{N}$ , we say that  $X$  is *Turing reducible* to  $Y$  — written  $X \leq_T Y$  — if  $X$  is  $Y$ -computable. We write  $X <_T Y$  if  $X \leq_T Y$  but  $Y \not\leq_T X$ .  $\diamond$

The Turing reduction forms an *pre-order* on the sets of integers, i.e., this relation is reflexive and transitive. Indeed, if  $Y$  is  $X$ -computable and  $Z$  is  $Y$ -computable,  $Z$  is  $X$ -computable. However, it is not a partial order on the sets, because the Turing reduction is not anti-symmetric. For example, the sets of even numbers and odd numbers are trivially mutually computable since they are both computable, but they are not equal as sets of integers.

It is however possible to transform this pre-order into a partial order by identifying all the mutually computable sets. This gives the notion of *Turing degree* which represents a more robust computational power than the notion of set for the Turing reduction.

**Definition 5.2.** We write  $X \equiv_T Y$  if  $X \leq_T Y$  and  $Y \leq_T X$ . We will then say that  $X$  and  $Y$  are *Turing-equivalent*. We call *Turing degrees* the equivalence classes of the relation  $\equiv_T$ . The Turing degree of a set  $X$  is the collection  $\deg_T(X) = \{Y : Y \equiv_T X\}$ .  $\diamond$

By construction, if  $X \leq_T Y$ , then any element in the Turing degree of  $Y$  computes any element in the Turing degree of  $X$ . The Turing reduction therefore induces a partial order on the Turing degrees, which we will simply note  $\leq$ .

### Notation

We denote by  $(\mathcal{D}, \leq)$  the set of Turing degrees  $\mathcal{D}$  partially ordered by  $\leq$ . In general, the letters  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \dots$  will be used to designate its elements. We will sometimes write  $X \leq_T \mathbf{d}$  for  $\deg_T(X) \leq \mathbf{d}$ .

**Exercise 5.3.** Show that if  $X \equiv_T Y$  and  $A \equiv_T B$ , then we have  $X \leq_T A$  iff  $Y \leq_T B$ .  $\diamond$

Two sets are therefore Turing-equivalent if they have the same computational power, and the relation  $\leq$  makes it possible to compare no longer sets, but degrees of computational power. In particular, the Turing degrees are stable under finite variations, in the sense that if  $X \in \mathbf{d}$  and  $Y =^* X$ ,

then  $Y \in \mathbf{d}$ . Here,  $Y =^* X$  means that  $X$  and  $Y$  differ only over a finite number of bits.

**Exercise 5.4.** Show that Turing degrees are stable under finite variation.

◇

A large part of Classical Computability Theory consists in understanding the structure  $(\mathcal{D}, \leq)$ . Is the order  $\leq$  total over  $\mathcal{D}$ ? Is it well founded? If it is a partial order, what is the maximum size of an anti-chain? We will see that the structure of the Turing degrees is of great richness, but also of great complexity. Let's start with some immediate observations.

First of all, Turing degrees have a minimal element, namely the degree of computable sets. We will note it  $\mathbf{0}$ . We then have the following proposition.

**Proposition 5.5.** Any Turing degree is countably infinite. ★

PROOF. Let  $\mathbf{d}$  be a Turing degree and let  $X \in \mathbf{d}$ . In particular,

$$\mathbf{d} = \deg_T(X) = \{Y : X \equiv_T Y\} \subseteq \{\{n : \Phi_e^X(n) \downarrow = 1\} : e \in \mathbb{N}\},$$

therefore  $\mathbf{d}$  is countable. Moreover,  $\mathbf{d}$  is stable under finite variations, therefore is infinite. Thus,  $\mathbf{d}$  is countably infinite. ■

The collection of subsets of  $\mathbb{N}$  being uncountable, and each belonging to a Turing degree, it follows from the previous proposition that there are uncountably many Turing degrees. Suppose indeed the opposite. Then, according to Exercise 2-3.11, the union of all Turing degrees would be a countable set, as a countable union of countable sets is countable. As this union is equal to  $2^{\mathbb{N}}$ , there we obtain a contradiction. Note that Exercise 2-3.11 uses the axiom of choice (which we will discuss in detail in Section 9-4). The use of the axiom of choice is however not necessary: we will see several effective constructions of functions  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  (see Exercise 8-5.4 or Exercise 8-5.3) such that  $f(X)$  and  $f(Y)$  are in degrees Turing different for all  $X \neq Y$ , which makes  $f$  an injection of  $2^{\mathbb{N}}$  into Turing degrees and in particular shows  $|2^{\mathbb{N}}| \leq |\mathcal{D}|^1$ .

**Definition 5.6.** The *effective join* of two sets  $A$  and  $B$  is the set  $A \oplus B = \{2n : n \in A\} \cup \{2n+1 : n \in B\}$  (note that the intersection between the two sets is disjoint). ◇

The effective join of two sets is a way of encoding the information of each set so as to be able to decode it computably. There are of course many

---

<sup>1</sup>The inequality  $|\mathcal{D}| \leq |2^{\mathbb{N}}|$  seems obvious, but it uses the axiom of choice, in order to uniformly select an element in each Turing degree. We will discuss this briefly in Section 12-2.3.

ways to encode two sets into one, the effective join being the most direct and efficient, in the following sense.

**Proposition 5.7.** Let  $A$  and  $B$  be two sets. Then,  $\deg_T(A \oplus B)$  is the least upper bound of the degrees  $\deg_T(A)$  and  $\deg_T(B)$ , i.e., any degree above  $\deg_T(A)$  and  $\deg_T(B)$  is also above  $\deg_T(A \oplus B)$ . Thus, any pair of Turing degrees  $\mathbf{c}$  and  $\mathbf{d}$  admits an upper bound which we will denote by  $\mathbf{c} \cup \mathbf{d}$ . ★

PROOF. It is clear that the join  $A \oplus B$  allows to compute  $A$  and  $B$ . Thus,  $\deg_T(A \oplus B)$  is an upper bound of  $\deg_T(A)$  and  $\deg_T(B)$ . Let  $\deg_T(C)$  be an upper bound of  $\deg_T(A)$  and  $\deg_T(B)$ . In particular,  $C \geq_T A$  and  $C \geq_T B$ . Let  $i$  and  $j$  be codes such that  $\Phi_i(C, n) = A(n)$  and  $\Phi_j(C, n) = B(n)$  for all  $n$ .

The function  $f : \mathbb{N} \rightarrow \{0, 1\}$  defined by

$$f(n) = \begin{cases} \Phi_i(C, n/2) & \text{if } n \text{ is even} \\ \Phi_j(C, (n-1)/2) & \text{otherwise} \end{cases}$$

is  $C$ -computable, and we have  $f(n) = 1$  iff  $n \in A \oplus B$  for any integer  $n$ . Then,  $C \geq_T A \oplus B$ . ■

### Notation

In the proof of the previous proposition, we made use of the predicate “ $\Phi_i(C, n) \downarrow = A(n)$ ” for all  $n$ . Sometimes we will use the shorter notation  $\Phi_i(C) = A$ .

Beware, the least upper bound  $\mathbf{c} \cup \mathbf{d}$  of  $\mathbf{c}$  and  $\mathbf{d}$  has of course nothing to do with the set-theoretic union of the degrees  $\mathbf{c}$  and  $\mathbf{d}$ . In general, the letters in bold type  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \dots$  will be used to talk about degrees as abstract objects within a partial order, the detail of the sets of integers constituting each degree then not being relevant.

We will now be juggling between sets of integers and Turing degrees. Since each Turing degree can be represented by a set of integers (one of its members), an operation on the Turing degrees will normally be done via an operation on one of its representatives, so that the expected result is independent of the choice of such representative. The effective join constitutes a first example illustrating our point.

**Exercise 5.8.** Show that if we assume that  $X \leq_T Y$  and  $A \leq_T B$ , then

$$X \oplus A \leq_T Y \oplus B.$$

◇

The previous exercise shows that the effective join induces an operation on Turing degrees. We will study in the next section a new operation

on degrees playing an essential role in Computability Theory: the *Turing jump*.

## 6. Turing jump

The Turing jump is a fundamental operation in Computability Theory, and is defined as the relativization of the halting problem.

**Definition 6.1.** Given a set  $X$ , we define

$$X' = \{n : \Phi_n^X(n) \downarrow\}.$$

The *Turing jump* is the  $X \mapsto X'$  operator. ◇

For example, we can now define the halting problem relative to the halting problem: the set of computer program codes that halt on their own input, but using the halting problem as an oracle. We denote it  $\emptyset''$ . It is not very difficult to show that the Turing jump induces an operation on Turing degrees, and we leave the proof as an exercise.

**Exercise 6.2. (\*)** Show that if  $X \leq_T Y$ , then  $X' \leq_T Y'$ . ◇

We will see a reinforcement of the result of the previous exercise with Exercise 5-5.7. We will therefore denote by  $\mathbf{d}'$  the Turing jump of a Turing degree  $\mathbf{d}$ . In particular,  $\mathbf{0}'$  is the Turing degree of the halting problem.

**Proposition 6.3.** We have  $X <_T X'$  for all  $X \in 2^{\mathbb{N}}$ . Thus we have  $\mathbf{d} < \mathbf{d}'$  for any Turing degree  $\mathbf{d}$ . ★

PROOF. Let us first show that  $X'$  computes  $X$ . Let  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  be the partial  $X$ -computable function defined by

$$f(e, n) = \begin{cases} 1 & \text{if } e \in X \\ \uparrow & \text{otherwise.} \end{cases}$$

By the SMN theorem relativized to  $X$  (see Theorem 3.3), there exists a total computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that for all integers  $e$  and  $n$ , we have  $\Phi_{g(e)}^X(n) = f(e, n)$ . Thus, for all  $e$ :

- if  $e \in X$ , then  $\Phi_{g(e)}^X$  is the constant function 1, and therefore  $g(e) \in X'$ ;
- if  $e \notin X$ , then  $\Phi_{g(e)}^X$  is the nowhere defined function, and  $g(e) \notin X'$ .

In particular,  $X'$  can compute  $X$ : to know if  $n \in X$ , it suffices to look at whether  $g(n) \in X'$ .

The proof that  $X \not\leq_T X'$  is a relativization of the fact that  $\emptyset'$  is not computable, which was previously demonstrated with Theorem 3.1. ■

There is therefore a strictly increasing hierarchy of Turing degrees:

$$\mathbf{0} < \mathbf{0}' < \mathbf{0}'' < \dots$$

In the previous proof, note that the Turing functional used to compute  $X$  from  $X'$  is *the same* for any oracle  $X$ . This is something that will be used from time to time, for example in Chapter 26. We give the unconvinced reader the opportunity to reflect on this in the following exercise.

**Exercise 6.4.** Show that there exists a functional  $\Phi_e$  such that:

- $\Phi_e(X', n) = X(n)$  for all  $X \in 2^{\mathbb{N}}$  and for all  $n \in \mathbb{N}$ .
- $\Phi_e(Y, n) \downarrow$  for all  $Y \in 2^{\mathbb{N}}$  and for all  $n \in \mathbb{N}$ .  $\diamond$

Finally, note that the Turing jump is not an injective operator, as we will see later through the low and high sets. We will occasionally use the following notion of Turing-completeness.

**Definition 6.5.** A set  $A$  is said to be *Turing-complete* or (simply) *complete* if  $A \geq_T \emptyset'$ . A *complete* Turing degree is a degree  $\mathbf{d} \geq \mathbf{0}'$ . A set or degree which is not complete is *incomplete*.  $\diamond$

## 7. Limit computability

We are now going to study certain properties of sets computable by the halting problem. These sets admit in particular a very natural characterization in terms of approximations.

**Definition 7.1.** A function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  is *stable* if for all  $x \in \mathbb{N}$ ,  $\lim_y f(x, y)$  exists, that is, for every  $x \in \mathbb{N}$ , there is some threshold  $t \in \mathbb{N}$  such that for every  $y > t$ ,  $f(x, y) = f(x, t)$ . A set  $A$  is *limit-computable* if there exists a computable stable function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  such that for all  $x$

$$\lim_y f(x, y) = 1 \text{ iff } x \in A$$
 $\diamond$

We obtain our first characterization of the  $\emptyset'$ -computable sets.

**Lemma 7.2 (Shoenfield limit lemma).** A set  $A \subseteq \mathbb{N}$  is  $\emptyset'$ -computable if, and only if, it is limit-computable.  $\star$

PROOF. We can refer to figures 7.4 and 7.3 to help us in understanding the proof.

$\Rightarrow$ . Suppose that  $A \leq_T \emptyset'$  via a functional  $\Phi_e$ . Let  $\emptyset'_0 \subseteq \emptyset'_1 \subseteq \dots$  be a c.e. approximation of  $\emptyset'$ . Let  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  be the function which,

for an input  $(x, s)$ , checks if  $\Phi_e(\theta'_s, x)[s] \downarrow$ . If so,  $f(x, s) = \Phi_e(\theta'_s, x)[s]$ . Otherwise,  $f(x, s)$  is assigned an arbitrary value.

Let us show that  $f$  is stable and that its limit is  $A$ . Let  $x \in \mathbb{N}$ . By the use property, as  $\Phi_e(\theta'_s, x) \downarrow$ , this computation is done using the  $n$  first bits of the oracle for some integer  $n$ . Let then  $s$  be large enough so that  $\theta'_s \upharpoonright_n = \theta' \upharpoonright_n$  and so that  $\Phi_e(\theta'_s, x) \downarrow$  halts before  $s$  stages of computation (any sufficiently large  $s$  works). Then, for all  $t \geq s$ ,

$$\Phi_e(\theta'_t, x)[t] \downarrow = \Phi_e(\theta', x),$$

in which case  $\lim_t f(x, t) = \Phi_e(\theta', x) = A(x)$ .

$\Leftarrow$ . Let us now suppose that  $A$  is limit-computable, by a computable stable function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$ . Then,

$$A = \{x : \exists y \forall z \geq y f(x, z) = 1\} \quad \text{and} \quad \overline{A} = \{x : \exists y \forall z \geq y f(x, z) = 0\}.$$

We will define a  $\theta'$ -computable procedure to determine if  $x \in A$  or  $x \in \overline{A}$ . Let  $u, v : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  be two total computable functions such that for all  $x, y, n$ ,

$$\begin{aligned} \Phi_{u(x,y)}(n) &= \begin{cases} 1 & \text{if } \exists z \geq y f(x, z) \neq 1 \\ \uparrow & \text{otherwise,} \end{cases} \\ \Phi_{v(x,y)}(n) &= \begin{cases} 1 & \text{if } \exists z \geq y f(x, z) \neq 0 \\ \uparrow & \text{otherwise.} \end{cases} \end{aligned}$$

Specifically,

$$u(x, y) \notin \theta' \text{ iff } \forall z \geq y f(x, z) = 1 \quad \text{et} \quad v(x, y) \notin \theta' \text{ iff } \forall z \geq y f(x, z) = 0.$$

Thereby,

$$A = \{x : \exists y u(x, y) \notin \theta'\} \quad \text{and} \quad \overline{A} = \{x : \exists y v(x, y) \notin \theta'\}.$$

Given an integer  $x$ , to know if  $x \in A$  or  $x \in \overline{A}$ , it suffices to look for the smallest  $y$  such that  $u(x, y) \notin \theta'$  or  $v(x, y) \notin \theta'$ . We will necessarily end up finding such an integer  $y$ . If  $u(x, y) \notin \theta'$ , then  $x \in A$ . If  $v(x, y) \notin \theta'$ , then  $x \notin A$ . The procedure is  $\theta'$ -computable, and we thus have  $A \leq_T \theta'$ . ■

**Exercise 7.5.** Show that if  $Y$  is  $X$ -c.e. and  $Z$  is  $Y$ -c.e., then  $Z$  is not necessarily  $X$ -c.e. ◇

Any  $\theta'$ -computable set  $A$  is therefore associated with a stable function  $f$  whose limit is  $A$ . This function is often presented in the form of a succession of *uniformly computable* sets  $A_0, A_1, \dots$  defined by  $A_y = \{x : f(x, y) = 1\}$  for all  $y$ .

Approximation of the bit $x$	Machine codes
0 .....	► $u(x, 0)$ and $v(x, 0)$
1 .....	► $u(x, 1)$ and $v(x, 1)$
0 .....	► $u(x, 2)$ and $v(x, 2)$
0 .....	► $u(x, 3)$ and $v(x, 3)$
1 .....	► $u(x, 4)$ and $v(x, 4)$
0 .....	► $u(x, 5)$ and $v(x, 5)$
0 .....	► $u(x, 6)$ and $v(x, 6)$
...	

Figure 7.3: Illustration of the reciprocal ( $\Leftarrow$ ) of the proof of Shoenfield's lemma : to approximate the bit  $x$  at stage  $y$ , one create the code  $u(x, y)$  of the program which halts everywhere if a value different from 1 occurs for  $x$  at a stage later than  $y$ , and the code  $v(x, y)$  of the program which halts everywhere if a value different from 0 occurs for  $x$  at a stage later than  $y$ .

### Uniform computability

We have introduced the concept of *uniformly computable* sequence  $A_0, A_1, \dots$ : each element  $A_n$  of the sequence is computable by *the same function*, parameterized by an additional parameter  $n$  which indicates that the  $n$ -th element of the sequence is computed.

Let us insist on the fact that a sequence of computable sets  $(X_i)_{i \in \mathbb{N}}$  is not necessarily uniformly computable: there does not necessarily exist an algorithm allowing to compute  $X_i$  as a function of  $i$ . As a trivial example, each  $X_i$  can simply be an infinite sequence of 0, except for the bit in position  $i$  which is equal to the  $i$ -th bit of  $\emptyset'$ . Each  $X_i$  is a finite set and is therefore computable. On the other hand, an algorithm allowing to compute *uniformly*  $X_i$  as a function of  $i$  would make it possible to compute the halting set, and therefore cannot exist.

**Definition 7.6.** Let  $A \leq_T \emptyset'$  be a set. A  $\Delta_2^0$  *approximation* of  $A$  is a uniformly computable sequence of sets  $A_0, A_1, \dots$  such that for all  $x$ ,  $\lim_y A_y(x)$  exists and is equal to  $A(x)$ .  $\diamond$

The  $\Delta_2^0$  approximations are not canonical. It is always possible for example to “accelerate” a  $\Delta_2^0$  approximation  $A_0, A_1, \dots$  by considering the sequence  $A_{g(0)}, A_{g(1)}, \dots$  for a computable and strictly increasing function  $g : \mathbb{N} \rightarrow \mathbb{N}$ .

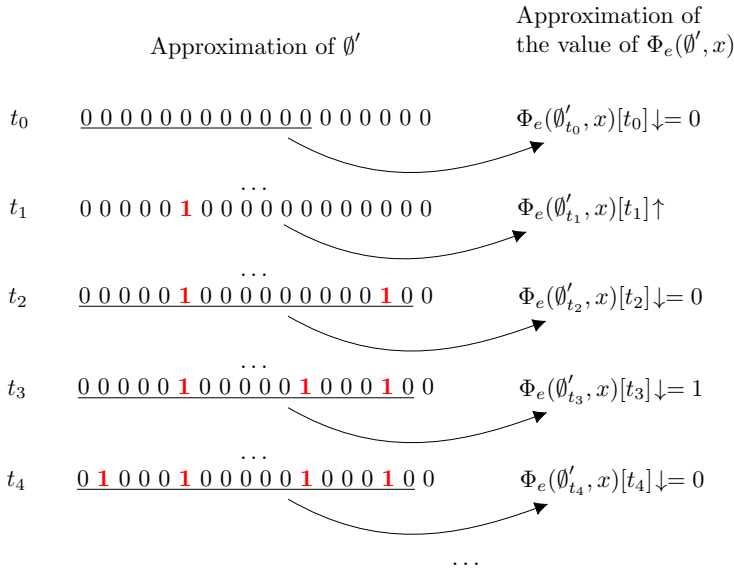


Figure 7.4: Illustration of the forward direction ( $\Rightarrow$ ) of the proof of Shoenfeld's lemma. The first column represents successive approximations of  $\emptyset'$ . The underlined part represents the use of the computation  $\Phi_e(\emptyset', x)[t_n]$  when it halts. While we enumerate the  $i$ th element in the halting set at stage  $t_i$ , one launches the computation of  $\Phi_e$  with the current approximation of the halting set as oracle, for  $t_i$  steps of computation. By the use property, the process converges necessarily.

### Remark

The name “ $\Delta_2^0$  approximation” will be justified in Chapter 5, where we will give a new characterization of  $\emptyset'$ -computable sets as those definable by a  $\Delta_2^0$  predicate.

The  $\Delta_2^0$  approximations make it possible to define two important functions, namely the modulus and the computation function. These functions express the computational complexity of the set  $A$  in the form of growth rate: any function increasing faster than the modulus of  $A$  or than its computation function allows to recompute  $A$ .

**Definition 7.7.** Let  $A_0, A_1, A_2, \dots$  be a  $\Delta_2^0$  approximation of a set  $A$ .

1. The *modulus* of the  $\Delta_2^0$  approximation is the function  $\mu_A : \mathbb{N} \rightarrow \mathbb{N}$  which to  $x$  associates the smallest integer  $n$  such that the se-

quence  $A_n \upharpoonright_x, A_{n+1} \upharpoonright_x, \dots$  is constant.

2. The *computation function* of the  $\Delta_2^0$  approximation is the function  $c_A : \mathbb{N} \rightarrow \mathbb{N}$  which to  $x$  associates the smallest integer  $n \geq x$  such that  $A_n \upharpoonright_x = A \upharpoonright_x$ .  $\diamond$

Note that any c.e. approximation is a degenerate  $\Delta_2^0$  approximation. In particular, every c.e. set is  $\emptyset'$ -computable. Unlike c.e. approximations which, when they cause an element to appear in  $A$ , never remove it again, a  $\Delta_2^0$  approximation of  $A$  has the right to “change its mind” an arbitrarily large (but finite) number of times, whether  $x$  belongs to  $A$  or not. In particular, the computation function generally grows slower than the modulus, because a set  $A_n$  can coincide with the set  $A$  on a long initial segment without having reached its threshold of stability on this segment.. The computation function, unlike modulus in general, is computable by the set  $A$ .

It is easy to verify that any function dominating the modulus of a  $\Delta_2^0$  approximation of a set computes this set.

**Exercise 7.8.** Let  $A_0, A_1, \dots$  be a  $\Delta_2^0$  approximation of a set  $A$ . Let  $\mu_A$  be its modulus. Show that any function dominating  $\mu_A$  computes  $A$ . A function  $f$  *dominates* a function  $g$  if  $f(n) \geq g(n)$  for all  $n \in \mathbb{N}$ .  $\diamond$

However, it is not clear that this is also the case with the computation function. This is however what the following proposition shows.

**Proposition 7.9 (Martin and Miller [156]).** Let  $A_0, A_1, \dots$  be a  $\Delta_2^0$  approximation of a set  $A$ . Let  $c_A$  be its computation function. Any function dominating  $c_A$  computes  $A$ .  $\star$

PROOF. Let  $f$  be a function dominating  $c_A$ . Let  $M(x)$  be the largest  $y \leq x$  such that for all  $x \leq t \leq f(x)$ ,  $A_t \upharpoonright_y = A_{f(x)} \upharpoonright_y$ . The function  $M$  is total  $f$ -computable. Moreover,  $M$  tends towards  $+\infty$ , because the approximation of  $A$  being  $\Delta_2^0$ , it will stabilize on increasingly larger initial segments. Finally, as  $x \leq c_A(x) \leq f(x)$ , then if  $M(x) = y$ ,  $A_x \upharpoonright_y = A_{c_A(x)} \upharpoonright_y = A \upharpoonright_y$ . Then, to decide if  $n \in A$ , it suffices to find an integer  $x$  such that  $M(x) > n$ , then test if  $n \in A_x$ . This procedure is  $f$ -computable.  $\blacksquare$

Note that it is important to ask that  $c_A(x) \geq x$  in the definition of the computation function. The preceding proposition becomes false when this precision is omitted.

#### Remark

The notions of modulus and computation function are not characteristic of a  $\emptyset'$ -computable set, but of a  $\Delta_2^0$  approximation of a set.

The same  $\emptyset'$ -computable set has an infinity of  $\Delta_2^0$  approximations, each having its modulus and its computation function.

## 8. Finite extensions method

We are now going to present a relatively simple and yet very powerful method, allowing to create sets satisfying “custom” computational properties. This is the *finite extension method*.

In Computability Theory, we call *weakness property* a property on sets, which is downward-closed under the Turing reduction, i.e., if  $X$  has a weakness property and if  $Y \leq_T X$ , then  $Y$  also has this weakness property. Conversely, a *strength property* is a property on sets, which upward-closed under the Turing reduction, that is, if  $X$  has a strength property and if  $X \leq_T Y$ , then  $Y$  also has this strength property.

The finite extension method is particularly suitable when we ask ourselves the question of the existence of sets simultaneously satisfying some strength and some weakness properties. It is then necessary to create a made-to-measure set, neither too strong nor too weak from a computational point of view. We will illustrate this method by proving two propositions.

**Proposition 8.1 (Kleene and Post, 1954).** There are two sets  $A$  and  $B$  which are incomparable by the Turing reduction. ★

**PROOF.** We are going to build simultaneously two sets  $A$  and  $B$ , seen as infinite binary sequences — remember the correspondence between the two.

The set  $A$  must satisfy a strength property ( $A$  is not computable by  $B$ ) and a weakness property ( $A$  does not compute  $B$ ). The set  $B$  must for its part satisfy the dual properties of strength and weakness.

**Requirements.** Computational properties, whether of strength or weakness, are in general schemes of properties, in the sense that they are declined in an infinity of more elementary properties and easier to satisfy independently. For example, the property “ $A \not\leq_T B$ ” corresponds to the collection of properties “ $\Phi_e^A \neq B$ ” for any functional code  $e$ , where “ $\Phi_e^A \neq B$ ” means that either  $\Phi_e^A$  is a partial function, or  $\Phi_e^A(x) \downarrow \neq B(x)$  for an  $x \in \mathbb{N}$ . These elementary properties are called *requirements*. The first step in a construction using the finite extension method consists in identifying the requirements. We therefore have two kinds  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\begin{aligned} \mathcal{R}_e &: \exists x \Phi_e^A(x) \uparrow \vee \exists x \Phi_e^A(x) \downarrow \neq B(x) \\ \mathcal{S}_e &: \exists x \Phi_e^B(x) \uparrow \vee \exists x \Phi_e^B(x) \downarrow \neq A(x). \end{aligned}$$

If all the  $\mathcal{R}$ -requirements are satisfied, then  $A \not\geq_T B$ , while if all the  $\mathcal{S}$ -requirements are satisfied,  $B \not\geq_T A$ . Since we are only interested in functionals computing elements of  $2^{\mathbb{N}}$ , we will consider — as often in this kind of case — that  $\Phi_e(X, n) \uparrow$  if ever  $\Phi_e(X, n) \downarrow \notin \{0, 1\}$ .

**Satisfaction of a requirement.** Suppose we want to satisfy only one requirement, say  $\mathcal{R}_e$ . Two cases arise.

- Case 1: there exists a set  $X$  such that  $\Phi_e^X(0) \downarrow = i$  for a given  $i \in \{0, 1\}$ . We can then fix  $A = X$  and  $B$  can be any set such that  $B(0) \neq i$ . We will then have satisfied the requirement  $\mathcal{R}_e$  by ensuring that  $\Phi_e^A(0) \downarrow \neq B(0)$ .
- Case 2: whatever the set  $X$ , we have  $\Phi_e^X(0) \uparrow$ . This case is even simpler,  $A$  and  $B$  can be any set.

The previous argument fully specified the sets  $A$  and  $B$  in order to satisfy a requirement  $\mathcal{R}_e$ , without leaving freedom for the other requirements to be satisfied. We will therefore try to be more economical to leave room for the satisfaction of other requirements. For that, we will make sure to specify only a finite initial segment of  $A$  and  $B$  to satisfy a given requirement.

**Construction.** The sets  $A$  and  $B$  will be built by finite approximations, in the form of two sequences of finite binary strings, representing increasingly long prefixes of  $A$  and  $B$ .

$$\sigma_0 \preceq \sigma_1 \preceq \dots \quad \text{and} \quad \tau_0 \preceq \tau_1 \preceq \dots$$

We do not necessarily ask that  $\sigma_n \prec \sigma_{n+1}$ : the sequence of strings may stagnate for a certain time. On the other hand, for all  $n$ , we ask that for  $m > n$  sufficiently large we have  $\sigma_n \prec \sigma_m$ . In this way, the strings  $\sigma_0 \preceq \sigma_1 \preceq \dots$  gradually converge towards a single infinite sequence  $A$  and the strings  $\tau_0 \preceq \tau_1 \preceq \dots$  gradually converge towards a single infinite sequence  $B$ . Formally, we define  $A$  and  $B$  via new notations: given a binary string  $\sigma$ , we will denote by  $[\sigma]$  the set of infinite binary sequences having  $\sigma$  as a prefix. We will therefore have  $A \in [\sigma_n]$  and  $B \in [\tau_n]$  for all  $n \in \mathbb{N}$ . By making sure that there are arbitrarily long strings, we make  $\bigcap_n [\sigma_n]$  and  $\bigcap_n [\tau_n]$  each contain exactly one element:  $A$  and  $B$  respectively.

The finite approximations of  $A$  and  $B$  will be defined in stages, so as to successively satisfy each requirement. As at a given step, only finite prefixes of  $A$  and  $B$  are known, it will therefore be necessary to satisfy a requirement whatever comes after these prefixes in the rest of the construction. A pair of strings  $\sigma_n$  and  $\tau_n$  *force* a requirement  $\mathcal{R}_e$  (or a requirement  $\mathcal{S}_e$ ) if the requirement property is satisfied for all  $A \succeq \sigma_n$  and  $B \succeq \tau_n$ . We must therefore ensure that the requirement is satisfied for all the elements of  $[\sigma_n]$

and  $[\tau_n]$ . Note in passing that if  $\sigma_n$  and  $\tau_n$  force a requirement, then for all  $\sigma \succeq \sigma_n$  and  $\tau \succeq \tau_n$  we have  $[\sigma] \subseteq [\sigma_n]$  and  $[\tau] \subseteq [\tau_n]$ , and therefore  $\sigma$  and  $\tau$  still force the requirement.

The different requirements are going to be intertwined so that each receives attention at a stage of the construction. During an even step  $n = 2e$ , we will define  $\sigma_{n+1}$  and  $\tau_{n+1}$  so as to force  $\mathcal{R}_e$ , while during an odd step,  $n = 2e+1$ , we will force  $\mathcal{S}_e$ . The requirements will therefore be satisfied in the order

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \dots$$

**Satisfaction of a requirement.** We are now going to satisfy the requirement  $\mathcal{R}_e$  again, this time specifying only a finite prefix of the oracles  $A$  and  $B$ . The case of  $\mathcal{S}_e$  requirements is symmetrical. Suppose that initial segments  $\sigma_n$  and  $\tau_n$  have already been specified for  $A$  and  $B$  (we start the construction with  $\sigma_0 = \tau_0 = \epsilon$  where  $\epsilon$  is the empty word). In other words, the final infinite binary sequences  $A$  and  $B$  must respect  $\sigma_n \preceq A$  and  $\tau_n \preceq B$ . Let  $x = |\tau_n|$ . In particular,  $x$  is the first position where  $B$  is not yet specified. All the values of  $B$  in the positions preceding  $x$  are already set by  $\tau_n$ . The following two cases arise.

- Case 1: there exists a set  $X \succeq \sigma_n$ , such that  $\Phi_e^X(x) \downarrow = i$  for a given  $i \in \{0, 1\}$ . In this case, by the use property, this computation calls upon a finite number of bits of the oracle, and there thus exists an initial segment  $\sigma_{n+1} \preceq X$  such that  $\Phi_e^{\sigma_{n+1}}(x) \downarrow = i$ , or in other words such that  $\Phi_e^Y(x) \downarrow = i$  for any set  $Y \succeq \sigma_{n+1}$ . We can choose the initial segment  $\sigma_{n+1}$  so that it is at least as long as  $\sigma_n$ , which ensures  $\sigma_{n+1} \succeq \sigma_n$ . Knowing that the set  $A$  will have  $\sigma_{n+1}$  for initial segment, we ensured that  $\Phi_e^A(x) \downarrow = i$ . Let  $\tau_{n+1}$  be the string obtained from  $\tau_n$  by adding the bit  $1 - i$  to it. In other words,  $|\tau_{n+1}| = |\tau_n| + 1$ ,  $\tau_{n+1} \succeq \tau_n$  and  $\tau_{n+1}(x) = 1 - i$ . We then ensured that for all  $B \succeq \tau_{n+1}$ ,  $B(x) = 1 - i$ . Thus, the strings  $\sigma_{n+1}$  and  $\tau_{n+1}$  extend  $\sigma_n$  and  $\tau_n$  respectively, and force the requirement  $\mathcal{R}_e$  by making sure that  $\Phi_e^A(x) \downarrow \neq B(x)$  for all  $A \succeq \sigma_n$  and  $B \succeq \tau_n$ .
- Case 2: for any set  $X \succeq \sigma_n$ , we have  $\Phi_e^X(x) \uparrow$ . In this case,  $\sigma_n$  and  $\tau_n$  already force the requirement  $\mathcal{R}_e$  by making sure  $\Phi_e^A(x) \uparrow$  for a certain  $x$ . It is therefore sufficient to take  $\sigma_{n+1} = \sigma_n$  and  $\tau_{n+1} = \tau_n$ .

We will ensure that the lengths of the elements of the sequences  $(\sigma_n)_{n \in \mathbb{N}}$  and  $(\tau_n)_{n \in \mathbb{N}}$  are increasingly large, by adding an arbitrary bit at the end of each string at the end of each step, before moving on to next step. As explained previously, forcing a requirement is closed under strings extension, so this will not alter the validity of the construction. The proof of Proposition 8.1 is complete.  $\blacksquare$

A new notation has been introduced in the previous proof; we oficialize its use below.

**Notation**

Given  $\sigma \in 2^{<\mathbb{N}}$ , we denote by  $[\sigma]$  the set of  $X \in 2^{\mathbb{N}}$  such that  $\sigma \prec X$ .

We will give another illustration of the finite extension method by proving a stronger proposition, implying Proposition 8.1. This time, we fix an arbitrary non-computable set  $A$ . We then construct a set  $B$  that  $A$  does not compute, and which does not compute  $A$ . This is a priori more difficult construction, because only one of the two sets is controlled. Note also that it will be necessary for this proof to use the fact that  $A$  is non-computable: indeed, if it were computable, it would be in particular computable from any set  $B$ .

**Proposition 8.2.** For any non-computable set  $A$ , there exists a set  $B$  such that  $B \not\leq_T A$  and  $A \not\leq_T B$ . ★

PROOF. Unlike Proposition 8.1, the set  $A$  is already fixed. We are going to build only the set  $B$  by the finite approximation method. The set  $B$  must still satisfy a strength property ( $B \not\leq_T A$ ) and a weakness property ( $A \not\leq_T B$ ). The requirements are therefore identical to those of Proposition 8.1, namely

$$\begin{aligned} \mathcal{R}_e & : \quad \exists x \Phi_e^A(x) \uparrow \vee \exists x \Phi_e^A(x) \downarrow \neq B(x) \\ \mathcal{S}_e & : \quad \exists x \Phi_e^B(x) \uparrow \vee \exists x \Phi_e^B(x) \downarrow \neq A(x). \end{aligned}$$

However, the sets  $A$  and  $B$  no longer playing a symmetrical role, the requirements  $\mathcal{R}_e$  and  $\mathcal{S}_e$  will each be satisfied in their own way.

**Construction.** The set  $B$  will be built by successive approximations

$$\tau_0 \preceq \tau_1 \preceq \tau_2 \preceq \dots$$

to define  $B$  as the only element of the set  $\bigcap_n [\tau_n]$ . A string  $\tau_n$  *forces* a requirement  $\mathcal{R}_e$  (or a requirement  $\mathcal{S}_e$ ) if the property is satisfied for all  $X \in [\tau_n]$ . At each stage of the construction, a requirement will be forced, by interlacing them as before:

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \dots$$

**Satisfaction of a requirement  $\mathcal{R}_e$ .** At step  $n$ , assume that the string  $\tau_n$  is defined. We want to find an extension  $\tau_{n+1} \succeq \tau_n$  forcing the requirement  $\mathcal{R}_e$ . The satisfaction of this type of requirement is very similar to that of Proposition 8.1. Let  $x = |\tau_n|$ . Two cases arise.

- Case 1:  $\Phi_e^A(x) \downarrow = i$  for an  $i \in \{0, 1\}$ . It is then enough to define  $\tau_{n+1}$  as the unique string of length  $|\tau_n| + 1$  extending  $\tau_n$  such that

$$\tau_{n+1}(x) = 1 - i.$$

As  $B \in [\tau_{n+1}]$ ,  $B(x) = \tau_{n+1}(x) = 1 - i$ , so  $\Phi_e^A(x) \downarrow \neq B(x)$ .

- Case 2:  $\Phi_e^A(x) \uparrow$ . In this case, the requirement  $\mathcal{R}_e$  is trivially satisfied, because  $\Phi_e^A$  is a partial function. It is therefore sufficient to take  $\tau_{n+1} = \tau_n$ .

Note that no assumptions were made on the set  $A$  to satisfy the requirements  $\mathcal{R}_e$ . The assumption according to which  $A$  is not computable will be exploited to satisfy the requirements  $\mathcal{S}_e$ .

**Satisfaction of a requirement  $\mathcal{S}_e$ .** The difficulty in satisfying a requirement such as  $\mathcal{S}_e$  comes from the fact that we have no control over the set  $A$ , which is fully specified. More precisely, during the satisfaction of a requirement  $\mathcal{R}_e$ , we fix an input  $x$  which is not yet specified for  $B$ , so as to choose its value in case 1 to make it different from the value of  $\Phi_e^A(x)$ . This is not possible in the case of the  $\mathcal{S}_e$  requirement, all the values of  $A$  being fixed. It will therefore be necessary to exploit the fact that the set  $A$  is not computable. Note on the other hand that the satisfaction of the requirements  $\mathcal{R}_e$  left a certain freedom, in particular in the choice of  $x$ . Indeed, only a finite number of entries is specified at a given step for  $B$ , and therefore almost all entries can be chosen for  $x$ . We will exploit this freedom of choice to satisfy the requirements  $\mathcal{S}_e$ . Three cases arise.

- Case 1: there is an input  $x$  and a set  $X \succeq \tau_n$  such that

$$\Phi_e^X(x) \downarrow \neq A(x).$$

By the use property, then there exists a finite string  $\tau_{n+1} \succeq \tau_n$  such that  $\Phi_e^X(x) \downarrow \neq A(x)$  for all  $X \in [\tau_n]$ . The string  $\tau_{n+1}$  therefore forces the requirement  $\mathcal{S}_e$ .

- Case 2: there is an input  $x$  such that for all sets  $X \succeq \tau_n$ , we have  $\Phi_e^X(x) \uparrow$ . In this case, the string  $\tau_n$  already forces the requirement  $\mathcal{S}_e$  ensuring that  $\Phi_e^B(x) \uparrow$ .
- Case 3: neither of the two previous cases occurs. We will then show that it is possible to compute the set  $A$ , and therefore to deduce a contradiction from it.

Here is the procedure to compute the value of  $A(x)$ : find a finite string  $\tau \succeq \tau_n$  such that  $\Phi_e^\tau(x) \downarrow$  and return the result of this computation. We claim the following two facts:

- (1) there is such a string, so the search will end,

(2) whatever  $\tau$  such that  $\Phi_e^\tau(x) \downarrow$ , then  $\Phi_e^\tau(x) \downarrow = A(x)$ .

To show (1), notice that the negation of case 2 means that for any  $x$  (and in particular for this  $x$  considered), there exists a set  $X \succeq \tau_n$  such that  $\Phi_e^X(x) \downarrow$ . By the use property, there then exists an initial segment  $\tau \succeq \tau_n$  of  $X$  such that  $\Phi_e^\tau(x) \downarrow$ .

Let us show (2). If  $\Phi_e^\tau(x) \downarrow \neq A(x)$ , then case 1 would be true taking any  $X \succeq \tau$ . It follows that  $\Phi_e^\tau(x) \downarrow$  implies  $\Phi_e^\tau(x) = A(x)$ . We have therefore described a computable procedure to determine the value of  $A(x)$  whatever  $x$ , contradicting the hypothesis according to which  $A$  is not computable.

This concludes the proof of Proposition 8.2. ■

Before concluding this section dedicated to the finite extension method, let us mention that in general, this method is not *effective*, in the sense that no computability constraint is imposed on the finite strings under construction. It is however possible to do a fine analysis of the argument to determine the computational power necessary to find a  $\tau_{n+1}$ , given  $\tau_n$ . We then obtain upper bounds on the complexity of the constructed set.

## 9. Low degrees

As we have seen, the Turing jump is invariant by Turing degree. Thus, for any computable set  $X$ ,  $X' \equiv_T \emptyset'$ . It is natural to wonder if only computable sets have a Turing jump equivalent to  $\emptyset'$ , and more generally if the Turing jump is an injective function on Turing degrees. The following proposition shows that this is not the case.

**Proposition 9.1.** There exists a non-computable set  $A$  such that

$$A' \equiv_T \emptyset'. \quad \star$$

**PROOF.** The set  $A$  will be built using an effective version of the finite extension method (see Section 8), making sure that the entire construction is computable in  $\emptyset'$ .

**Requirements.** The set  $A$  must satisfy a strength property ( $A$  non-computable) and a weakness property ( $A' \leq_T \emptyset'$ ).

The strength property comes in an infinity of requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : \exists x \Phi_e(x) \uparrow \vee \exists x \Phi_e(x) \downarrow \neq A(x).$$

The weakness property is of a new type. To satisfy it, we will make sure to “control” the Turing jump of  $A$  as the construction progresses, while

making sure that the whole construction itself is computable in  $\emptyset'$ . With the help of this fact, we will have a  $\emptyset'$ -computable function  $f$  for which we will have to satisfy an infinity of requirements  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{S}_e : \quad \Phi_e^A(e) \downarrow \rightarrow f(e) = 1 \quad \text{and} \quad \Phi_e^A(e) \uparrow \rightarrow f(e) = 0.$$

Informally, the requirement  $\mathcal{S}_e$  is satisfied if, at a finite moment of the construction, we know whether  $\Phi_e^A(e) \downarrow$  or  $\Phi_e^A(e) \uparrow$ , whatever the continuation of the construction.

**Construction.** The set  $A$  will be built by successive approximations

$$\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$$

to define  $A$  as the only element of  $\bigcap_n [\sigma_n]$ . A string  $\sigma_n$  *forces* a requirement  $\mathcal{R}_e$  or  $\mathcal{S}_e$  if the property is satisfied for all  $B \in [\sigma_n]$ . At each stage of the construction, a requirement will be forced, by interlacing them as before:

$$\mathcal{R}_0, \mathcal{S}_0, \mathcal{R}_1, \mathcal{S}_1, \mathcal{R}_2, \mathcal{S}_2, \dots$$

We must also make sure that the construction is computable in  $\emptyset'$ . Thus, to know the value of  $A'(e)$ , it will suffice to execute using  $\emptyset'$  the construction up to step  $2e$  satisfying  $\mathcal{S}_e$ , and return the result. This  $\emptyset'$ -computable procedure ensures that  $A' \leq_T \emptyset'$ . We will therefore show how to satisfy each type of requirement independently, while analyzing the computational complexity of each step to ensure that  $\sigma_{n+1}$  can be obtained from  $\sigma_n$  using the oracle  $\emptyset'$ .

**Satisfaction of a requirement  $\mathcal{R}_e$ .** Suppose  $\sigma_n$  is already defined. We want to find  $\sigma_{n+1} \succeq \sigma_n$  forcing the requirement  $\mathcal{R}_e$ . Let  $x = |\sigma_n|$ . In other words,  $x$  is the first value of  $A$  that is not yet specified. Two cases arise.

- Case 1:  $\Phi_e(x) \uparrow$  in which case  $\mathcal{R}_e$  is trivially satisfied, and we can define  $\sigma_{n+1} = \sigma_n$
- Case 2:  $\Phi_e(x) \downarrow$ , in which case the string  $\sigma_{n+1}$  obtained from  $\sigma_n$  by adding the bit  $1 - \Phi_e(x)$  force  $\mathcal{R}_e$ .

Depending on the case, we will therefore define a different  $\sigma_{n+1}$  extension. It must be ensured that this extension can be obtained computably using the oracle  $\emptyset'$ . The distinction between the two cases is not computable in itself, because it asks to decide if  $\Phi_e(x)$  halts or not. However, we can use  $\emptyset'$  to answer this question as follows: let  $\Phi_i$  be the partial computable function defined for all  $n$  by  $\Phi_i(n) = \Phi_e(x)$ . The code  $i$  can be computably constructed, because it is a simple machine manipulation. It follows that we are in the first case iff  $i \notin \emptyset'$ . In the first case,  $\sigma_{n+1} = \sigma_n$  is trivially

computable, while in the second case, it suffices to run  $\Phi_e(x)$  to retrieve the returned value, and thus obtain  $\sigma_{n+1}$ . We can therefore find an extension  $\sigma_{n+1}$  forcing the requirement  $\mathcal{R}_e$  using the oracle  $\emptyset'$ .

**Satisfaction of a requirement  $\mathcal{S}_e$ .** Suppose  $\sigma_n$  is already defined. We want to find  $\sigma_{n+1} \succeq \sigma_n$  such that the behavior of  $\Phi_e^A(e)$  is already defined by  $\sigma_{n+1}$ , in other words  $\Phi_e^X(e) \downarrow$  for all  $X \in [\sigma_{n+1}]$  or  $\Phi_e^X(e) \uparrow$  for all  $X \in [\sigma_{n+1}]$ . Two cases still arise.

- Case 1: there is a string  $\tau \succeq \sigma_n$  such that  $\Phi_e^\tau(e) \downarrow$ . In this case, taking  $\sigma_{n+1} = \tau$ , we ensure that  $\Phi_e^A(e) \downarrow$ , because  $A \in [\sigma_{n+1}]$ .
- Case 2: for any string  $\tau \succeq \sigma_n$ ,  $\Phi_e^\tau(e) \uparrow$ . In this case, by the use property, whatever the oracle  $A \in [\sigma_n]$ ,  $\Phi_e^A(e) \uparrow$ . By defining  $\sigma_{n+1} = \sigma_n$ , we therefore force  $\Phi_e^A(e) \uparrow$ .

Then, in each case, we have forced the behavior of  $\Phi_e^A(e)$  with a finite prefix of the oracle.

Here again, we have to find  $\sigma_{n+1}$  from  $\sigma_n$  computably using the oracle  $\emptyset'$ . As for the requirement  $\mathcal{R}_e$ , we define a partial computable function  $\Phi_i$  which, for each of its inputs, searches for a string  $\tau \succeq \sigma_n$  such that  $\Phi_e^\tau(e)[|\tau|] \downarrow$ , and halts if it finds any. Thus,  $i \in \emptyset'$  if, and only if, we are in the first case. Once the case has been determined, the string  $\sigma_{n+1}$  can be found computably. This concludes the proof of Proposition 9.1. ■

Among the first notions of weakness introduced and studied in Computability Theory by Cooper and Soare independently, is the hierarchy of  $\text{low}_n$  sets, of which we give here the first level.

**Definition 9.2.** A set  $A \subseteq \mathbb{N}$  is *low* if  $A' \leq_T \emptyset'$ . ◇

Informally, a set is low if it is indistinguishable from a computable set from the point of view of the Turing jump. We have seen that if  $X \leq_T Y$ , then  $X' \leq_T Y'$ .

Thus, as  $\emptyset \leq_T A$  for any set  $A$ ,  $\emptyset' \leq_T A'$ . It follows that a set is low iff  $A' \equiv_T \emptyset'$ . We will see further examples of non-computable low sets, in particular computably enumerable sets (see Chapter 13).

## 10. High degrees

If we consider a set  $A \leq_T \emptyset'$ , what are the extreme powers that can be taken by its Turing jump  $A'$ ?

Recall that if  $X \leq_T Y$ , then  $X' \leq_T Y'$ . In particular,  $A' \geq_T \emptyset'$ . On the other hand,  $A' \leq_T \emptyset''$  because  $A \leq_T \emptyset'$ . So we have

$$\emptyset' \leq_T A' \leq_T \emptyset'' \text{ if } A \leq_T \emptyset'.$$

Note that in the case where  $A \not\leq_T \emptyset'$ , the Turing jump of  $A$  can be arbitrarily complex.

The sets whose Turing jump is equal to the minimum bound, namely  $\emptyset'$ , are the low sets. We have seen that there are non-computable low sets. We are now going to look at the sets whose Turing jump computes the maximum bound  $\emptyset''$ .

**Definition 10.1.** A set  $A \subseteq \mathbb{N}$  is *high* if  $\emptyset'' \leq_T A'$ . ◇

**Remark**

The notion of high set has historically been defined only for sets  $A \leq_T \emptyset'$ . It has since been extended to all sets, but it's important to keep this historical difference in mind when reading the founding articles on Computability Theory.

Now let's think about the high sets. Unlike the low, their Turing jump has more computational power than expected: it allows the double jump to be computed. The halting problem itself is obviously a trivial example of a high set. As for low sets, the notion of high set is useful for non-trivial examples: high sets which do not compute  $\emptyset'$ . It is in fact possible to show that for any non-computable set  $C$  there exists a high set which does not compute  $C$ .

In the following proof, we use for the first time oracles which are not sets of integers, but functions  $f : \mathbb{N} \rightarrow \mathbb{N}$ . Such a function can be represented by the infinite binary sequence  $G_f$  such that  $G_f(\langle n, m \rangle) = 1$  iff  $f(n) = m$ . It is clear that  $G_f$  allows to compute  $f$ , and that any set  $Y$  allowing to compute  $f$  can also compute  $G_f$ : the set  $G_f$  is a minimum representation of the function  $f$  in Turing degrees. Other equivalent representations in terms of Turing degree are possible, for example with  $G_f = 1^{f(0)}01^{f(1)}01^{f(2)}0 \dots$  (We start with the  $f(0)$  first bits at 1, followed by a 0, then we continue with the next  $f(1)$  bits to 1, followed by a 0, etc.).

**Proposition 10.2.** For any non-computable set  $C$ , there exists a high set  $A$  such that  $A \not\geq_T C$ . ★

**PROOF.** The proof is quite similar to that of Proposition 8.2 with the finite extension method.

**Requirements.** The set  $A$  must satisfy a strength property ( $A' \geq_T \emptyset''$ ) and a weakness property ( $C \not\leq_T A$ ). Requirements for the weakness prop-

erty are standard, namely

$$\mathcal{S}_e : \exists x \Phi_e^A(x) \uparrow \vee \exists x \Phi_e^A(x) \downarrow \neq C(x).$$

The case of the strength property is different. First of all, it is not a question of controlling what the set  $A$  computes, but of controlling what its Turing jump computes. Then, this strength property is not expressed in a negative form (not to be computed by another set) but in a positive form (to compute a complicated object). Negative formulations are often proven by diagonalization, while positive formulations are more constructive. We are therefore not going to express the strength property in the form of requirements, but to impose it structurally in the nature of the object that we construct.

In previous uses of the finite extension method, we constructed a set by creating an infinite sequence of binary strings forming finite approximations of the set.

This time, we are going to build a stable function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  whose limit is  $\emptyset''$ . As explained in the paragraph preceding the proof, let us recall that this can be reduced to the construction of an element of  $2^{\mathbb{N}}$  by considering the set  $G_f$  defined by  $\langle n, m \rangle \in G_f$  iff  $f(n) = m$ . By the relativization of the Shoenfield limit lemma to  $f$  (see Lemma 7.2), a set  $B$  is limit- $f$ -computable iff  $B \leq_T f'$ . In particular, for  $B = \emptyset''$ , if  $\emptyset''$  is limit- $f$ -computable, then  $\emptyset'' \leq_T f'$ , in other words  $f$  is high. The set  $A$  is therefore any set in the Turing degree of  $f$ .

**Construction.** The function  $f$  will be built from successive finite approximations which are more and more precise. These approximations, instead of being binary strings, will be pairs  $(g, m)$ , where

- $g \subseteq \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  is a partial function with two parameters whose domain is finite, representing a piece of the function  $f$  that we are building.
- $m$  is an integer meaning that henceforth, when we extend the domain of  $g$  with a new input  $(x, y)$ , if  $x < m$  then  $g(x, y) = \emptyset''(x)$ .

In other words, the limit of the  $m$  first “columns” of the function  $f$  has already been reached and has the correct value. We will call these couples *conditions*, because they condition part of the behavior of the function  $f$ .

In the same way that the suffix relation  $\sigma \preceq \tau$  for binary strings means that  $\tau$  is a more precise approximation of the sequence that we construct, we will define an extension relation on the conditions  $(g, m) \preceq (h, n)$  to mean that the condition  $(h, n)$  is more precise, or more restrictive, than the condition  $(g, m)$ .

$y$				$m = 3$			
	$f(0, y)$	$f(1, y)$	$f(2, y)$	$f(3, y)$	$f(4, y)$	$f(5, y)$	$f(6, y)$
0	0	1	0	0	0	0	0
1	0	0	0	0	1	0	0
2	1	1	1	0	1	1	0
3	0	0	1	0	0	1	1
4	1	0	0	1	1	0	1
5	1	0	0	0	0	1	1
6	1	0	0	0	1	1	1
7	1	0	0	1	0	0	0
<hr/>							
				0	1	1	0
				=	=	=	=
				$\emptyset''(0)$	$\emptyset''(1)$	$\emptyset''(2)$	$\emptyset''(3)$
							$\emptyset''(4)$
							$\emptyset''(5)$
							$\emptyset''(6)$

Figure 10.3: The dark grey part represents a condition with  $m = 3$ . The light grey part represents an extension of this condition: each  $i$ th column for  $i < m$  is fixed for every extension to a value which has to correspond to the  $i$ th bit of  $\emptyset''$ .

Given a partial function  $h \subseteq \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$ , we denote by  $\text{dom } h$  its domain of definition. We therefore say that the condition  $(h, n)$  *extends*  $(g, m)$  (denoted  $(h, n) \succeq (g, m)$ ) if  $n \geq m$  and if, in addition,

(P1)  $g \subseteq h$ , i.e.  $\text{dom } g \subseteq \text{dom } h$  and for all  $(x, y) \in \text{dom } g$ ,

$$g(x, y) = h(x, y);$$

(P2) for all  $(x, y) \in \text{dom } h \setminus \text{dom } g$ , if  $x < m$ , then  $h(x, y) = \emptyset''(x)$ .

The property (P1) means that the finite functions must be compatible, and more precisely that the function  $h$  must extend the function  $g$ , while the property (P2) formalizes the idea according to which the second parameter of a condition fixes the columns of the function by stabilizing them. Figure 10.3 illustrates what has just been explained.

Let's stop for a moment to familiarize ourselves with this new mathematical object and learn how to manipulate it. First, for any condition  $(g, m)$  and any  $n$ ,  $(g, n)$  is also a condition. If in addition  $n \geq m$ , then  $(g, n)$  is an extension. The integer  $m$  does not impose any constraint in itself on the finite function  $g$  to form a condition  $(g, m)$ . On the other hand,  $m$  imposes restrictions on the extensions of  $(g, m)$ . More precisely, if  $n \geq m$ , then the set of extensions of  $(g, n)$  is a subset of the extensions of  $(g, m)$ . Finally,  $(\emptyset, 0)$  is a valid condition, where  $\emptyset$  is the function defined nowhere.

From the point of view of Computability Theory, the set of conditions is a computable set. On the other hand, the extension relation between two conditions cannot be computed because of the property (P2) which involves  $\emptyset''$ . However, if we fix  $m$ , then the extension relation between conditions having  $m$  for second component is computable, because this only involves a finite segment  $\emptyset'' \upharpoonright_m$  of the set  $\emptyset''$ . It suffices to “hardcode” this initial segment in the program. This observation will be exploited to satisfy the requirements  $\mathcal{S}_e$ .

In the same way that we denote by  $[\sigma]$  the set of infinite binary sequences having for initial segment  $\sigma$ , we will denote by  $[g, m]$  the set of total functions  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  such that  $g \subseteq f$  and such that for all  $(x, y) \notin \text{dom } g$  with  $x < m$  we have  $f(x, y) = \emptyset''(x)$ . Thus,  $[g, m]$  is the collection of the set of candidate functions that can be obtained by completing the partial approximation  $(g, m)$ . Note that  $[g, m]$  does not only contain stable functions.

We are therefore going to build  $f$  by successive approximations in the form of conditions

$$(g_0, m_0) \preceq (g_1, m_1) \preceq (g_2, m_2) \preceq \dots$$

to define  $f = \bigcup_t g_t$ . In other words,  $\text{dom } f = \bigcup_t \text{dom } g_t$ , and for any pair  $(x, y) \in \text{dom } f$ ,  $f(x, y) = g_t(x, y)$  for a  $t$  such that  $(x, y) \in \text{dom } g_t$ . The function  $f$  is well defined thanks to the compatibility property (P1). If we make sure that the integers  $m_t$  become arbitrarily large, it is easy to verify by the property (P2) that the resulting function  $f$  is stable, and has a limit of  $\emptyset''$ . Note in particular that  $f \in \bigcap_t [g_t, m_t]$ .

A condition  $(g, m)$  *forces* a requirement  $\mathcal{S}_e$  if the property is satisfied for all  $f \in [g, m]$ . Here, we have replaced the occurrences of  $A$  by  $f$  in the requirement  $\mathcal{S}_e$ . At each stage of construction, a requirement is going to be forced.

**Satisfaction of a requirement  $\mathcal{S}_e$ .** The argument is similar to that of Proposition 8.2, but handling conditions and not binary strings. Let  $(g_t, m_t)$  be a condition. The following three cases arise.

- Case 1: there is an input  $x$  and a function  $f$  in  $[g_t, m_t]$  such that

$$\Phi_e^f(x) \downarrow \neq C(x).$$

In this case, by the use property, there exists an extension  $(g_{t+1}, m_t)$  of  $(g_t, m_t)$  such that  $\Phi_e^f(x) \downarrow \neq C(x)$  for all  $f$  in  $[g_{t+1}, m_t]$ . Note that  $m_{t+1} = m_t$ . The condition  $(g_{t+1}, m_t)$  therefore forces the requirement  $\mathcal{S}_e$ .

- Case 2: there is an input  $x$  such that for all the functions  $f$  in  $[g_t, m_t]$ , we have  $\Phi_e^f(x) \uparrow$ . In this case, the condition  $(g_t, m_t)$  already forces the requirement  $\mathcal{S}_e$  ensuring that  $\Phi_e^f(x) \uparrow$ .
- Case 3: neither of the two previous cases occurs. We will then show that it is possible to compute the set  $C$ , and therefore to deduce a contradiction from it. Here is the procedure to compute the value of  $C(x)$ : look for a condition  $(h, m_t)$  extending  $(g_t, m_t)$  with the same second component  $m_t$ , such that  $\Phi_e^h(x) \downarrow$ . We claim the following two facts:
  - (1) there is such an extension, and therefore the search will end,
  - (2) whatever  $(h, m_t) \succeq (g_t, m_t)$ ,  $\Phi_e^h(x) \downarrow \rightarrow \Phi_e^h(x) = C(x)$ .

To show (1), notice that the negation of Case 2 means that for any  $x$  (and in particular for this  $x$  considered), there exists a function  $f \in [g_t, m_t]$  such that  $\Phi_e^f(x) \downarrow$ . By the use property, there then exists a condition  $(h, m_t) \succeq (g_t, m_t)$  such that  $\Phi_e^h(x) \downarrow$ .

Let us show (2). If  $\Phi_e^h(x) \downarrow \neq C(x)$ , then Case 1 would be true taking any  $f \in [h, m_t] \subseteq [g_t, m_t]$ . It follows that  $\Phi_e^h(x) \downarrow = C(x)$ . Finally, notice that we only considered conditions with the same  $m_t$ . As explained above, for  $m_t$  fixed, the extension relation is computable.

We have therefore described a computable procedure to determine the value of  $C(x)$  whatever  $x$ , contradicting the hypothesis according to which  $C$  is not computable.

It is then enough to satisfy each requirement  $\mathcal{S}_e$  by gradually extending our conditions, while making “artificially” their second components increase towards  $+\infty$  in order to ensure that the final solution is indeed a stable function whose limit is  $\emptyset''$ . This concludes the proof of Proposition 10.2. ■

#### Corollary 10.4

*There is a set  $A$  both high and Turing-incomplete. In other words,  $A' \geq_T \emptyset''$  and  $A \not\geq_T \emptyset'$ .*

PROOF. Immediate by Proposition 10.2, taking  $C = \emptyset'$ . ■

#### Remark

The name “condition” is a generic name borrowed from the theory of forcing, and which will be called upon to designate very different mathematical objects throughout this work, although all corresponding to the idea of an approximation of an object that we construct. This notion will be developed in Chapter 11.

Note that the argument of the previous proof does not depend specifically on  $\emptyset''$  and we could have constructed, for any  $B$ , a set  $A$  such that  $A' \geq_T B$  and  $A \not\geq_T C$ . We will see in Chapter 13 that there are non-computable low c.e. sets. Sacks [186] has also shown the existence of high c.e. sets of incomplete degrees.

There is a big difference between the low and high sets: the former are all computable in  $\emptyset'$ , and they are therefore in countable quantity. The second, on the other hand, can be arbitrarily complex, and it is easily shown that they are in uncountable quantity. However, we will see with Corollary 19-3.9 and Proposition 10-3.38 that there are “few” high sets, from the point of view of Measure Theory and the theory of categories of Baire.

Let's end this chapter with a few exercises.

**Exercise 10.5. (★)** Adapt the proof of Proposition 10.2 to show that for any set  $B$  and any non-computable set  $C$ , there exists a set  $A$  such that  $A' \geq_T B$  and  $A \not\geq_T C$ .  $\diamond$

**Exercise 10.6. (★★)** Adapt the proof of Proposition 10.2 to show that there exists an incomplete and  $\emptyset'$ -computable high set.  $\diamond$

**Exercise 10.7. (★★)** Show by the finite extension method that there exists a sequence of sets  $(A_n)_{n \in \mathbb{N}}$  pairwise incomparable for the Turing reduction, i.e., such that

$$A_n \not\leq_T A_m \text{ et } A_m \not\leq_T A_n,$$

for all  $n \neq m \in \mathbb{N}$ .  $\diamond$



# Chapter 5

## Arithmetic hierarchy

We introduce a complexity hierarchy on the sets of integers. The computably enumerable sets are called  $\Sigma_1^0$  and their complements are called  $\Pi_1^0$ . The effective countable intersections of  $\Sigma_1^0$  sets are said to be  $\Pi_2^0$  and the effective countable union of  $\Pi_1^0$  sets are said to be  $\Sigma_2^0$ , and so on.

We call this construction *arithmetic hierarchy*, because the sets it contains are exactly those which are definable by a first-order formula in the language of Peano arithmetic. We will talk about this equivalence again in Section 9-3. There is a direct correspondence between the definition of a set by meetings/intersections, and the fact of being able to define it by a formula comprising quantifiers  $\exists/\forall$ . Thus, a union corresponds to an existential quantifier and an intersection to a universal quantifier.

**Example 1.** The set of codes  $e$  for the total computable functions can be written

$$\{e \in \mathbb{N} : \forall n \exists t \Phi_e(n)[t] \downarrow\} = \bigcap_n \bigcup_t \{e \in \mathbb{N} : \Phi_e(n)[t] \downarrow\}.$$

It is a  $\Pi_2^0$  set, that is to say a countable intersection of  $\Sigma_1^0$  sets, each of them being moreover a countable union of computable sets.

### 1. Elementary properties

Let's start with a formal definition of the arithmetic hierarchy.

**Definition 1.1.** Let  $m, n \geq 1$ .

1. A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Sigma_n^0$  if there exists a computable set  $R$  included in  $\mathbb{N}^{n+m}$  such that

$$A = \{(y_1, \dots, y_m) : \overbrace{\exists x_1 \forall x_2 \dots Q x_n}^{n \text{ quantifiers}} (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

where  $Q$  is  $\exists$  if  $n$  is odd, and  $\forall$  if  $n$  is even.

2. A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Pi_n^0$  if there exists a computable set  $R$  included in  $\mathbb{N}^{n+m}$  such that

$$A = \{(y_1, \dots, y_m) : \overbrace{\forall x_1 \exists x_2 \dots Q x_n}^{n \text{ quantifiers}} (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

where  $Q$  is  $\forall$  if  $n$  is odd, and  $\exists$  if  $n$  is even. ◇

Beware, in the preceding definition, it is the alternation between the quantifiers  $\exists$  and  $\forall$  which counts.

Then, for example, the set

$$\{y : \exists x_1 \forall x_2 R(y, x_1, x_2)\}$$

for  $R$  computable is a  $\Sigma_2^0$  set, but the set

$$\{y : \exists x_1 \exists x_2 \exists x_3 R(y, x_1, x_2, x_3)\}$$

is, despite its three quantifiers, a  $\Sigma_1^0$  set: one can easily eliminate repetitions of quantifiers of the same type. Consider for example a formula of the form

$$\exists x_1 \exists x_2 \forall y_1 \forall y_2 R(x_1, x_2, y_1, y_2).$$

The latter can simply be rewritten as  $\exists x \forall y R'(x, y)$  where  $R'$  will be a modified version of  $R$ , which will consider  $x$  and  $y$  respectively as pairs  $\langle x_1, x_2 \rangle$  and  $\langle y_1, y_2 \rangle$ . The predicate  $R'$  will then use the projections  $\pi_1$  and  $\pi_2$  realizing the inverse functions of the computable coupling bijection  $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$ , in order to recover  $x_1, x_2, y_1, y_2$ . The reader can go back to Exercise 3-2.3 to convince himself that the bijection  $\langle \cdot \rangle$  and that its two inverse functions are computable. An abuse of notation will consist for example of writing  $\exists \langle x_1, x_2 \rangle \forall \langle y_1, y_2 \rangle R(x_1, x_2, y_1, y_2)$ , meaning that the predicate  $R$  takes care of recovering  $x_1, x_2$  from  $\langle x_1, x_2 \rangle$ , and the same for  $y_1, y_2$ .

### Predicates

We will sometimes speak of  $\Sigma_n^0$  predicates to denote a formula of the form  $\exists x_1 \forall x_2 \dots Q x_n R(x_1, x_2, \dots, x_n)$ , where  $R$  is a computable predicate.

Note that by using the fact that the complement of a computable set is computable, we can easily see that the  $\Sigma_n^0$  sets are the complements of the  $\Pi_n^0$  sets. It is also quite possible for a set to be both  $\Sigma_n^0$  and  $\Pi_n^0$ . For this, the following notion is introduced.

**Definition 1.2.** Let  $m \geq 1$ . A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Delta_n^0$  for  $n > 0$  if it is both  $\Sigma_n^0$  and  $\Pi_n^0$ . ◇

The arithmetic hierarchy establishes a first level of distinction between arithmetically definable sets. Intuitively, a  $\Sigma_{n+1}^0$  set is strictly more complex than a  $\Sigma_n^0$  or even  $\Pi_n^0$  set. Indeed, each of the last two can be written in a  $\Sigma_{n+1}^0$  form by simply adding unused quantifiers.

**Example 1.3.** The  $\Sigma_2^0$  set given by  $\{y : \exists x_1 \forall x_2 R(y, x_1, x_2)\}$  can also be written in a  $\Pi_3^0$  form:  $\{y : \forall z \exists x_1 \forall x_2 R(y, x_1, x_2)\}$  or in a  $\Sigma_3^0$  form:  $\{y : \exists x_1 \forall x_2 \exists z R(y, x_1, x_2)\}$ .

However, it is not always possible to use fewer quantifiers. We will show with Corollary 5.6 that for all  $n > 0$ , there exist  $\Delta_{n+1}^0$  sets which cannot be described in a  $\Sigma_n^0$  or  $\Pi_n^0$  way: the arithmetic hierarchy is strict.

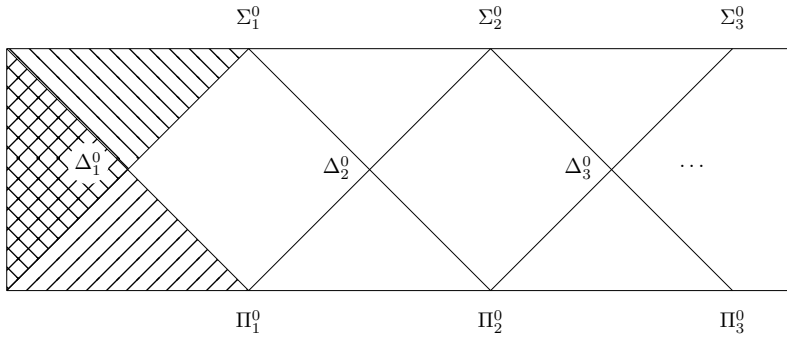


Figure 1.4: Representation of the arithmetic hierarchy. The upper left triangle represent  $\Sigma_1^0$  sets. The lower left hatched triangle represents  $\Pi_1^0$  sets. The intersection between the two — the double hatched triangle — represents  $\Delta_1^0$  sets. The remainder of the diagram follows similarly.

Before continuing, let us give some examples of sets of the arithmetic hierarchy.

**Example 1.5.**

- Any computable set  $A \subseteq \mathbb{N}$  is  $\Delta_1^0$  (we will see a proof of this with

Proposition 3.4). It suffices to add an unnecessary existential or universal quantification to the computable predicate  $A$  to be able to express it respectively as a  $\Sigma_1^0$  or  $\Pi_1^0$  set.

- Any computably enumerable set  $A \subseteq \mathbb{N}$  is  $\Sigma_1^0$  (we will see this precisely with Proposition 3.3). Indeed, such  $A$  is described as  $\{x : \exists t \Phi_e(x)[t] \downarrow\}$  for some  $e$ .
- We have seen with Example 1 that the set of total function codes is  $\Pi_2^0$ . Its complement, namely the set of partial function codes, is therefore  $\Sigma_2^0$ :  $\{e : \exists n \forall t \Phi_e(n)[t] \uparrow\}$
- The set of function codes which are total, except for a finite number of elements, is  $\Sigma_3^0$ :

$$\{e : \exists n \forall m \exists t \text{ such that } m < n \text{ or } \Phi_e(m)[t] \downarrow\}.$$

We will see a little later that the previous examples are optimal. For example, the set of function codes which are total except for a finite number of elements cannot be expressed in a  $\Pi_3^0$  way: it is a strict  $\Sigma_3^0$  set.

We now show a series of three propositions on the stability of  $\Sigma_m^0$  and  $\Pi_m^0$  sets under different operations. For example, the stability under finite union means that a finite union of  $\Sigma_m^0$  sets is still a  $\Sigma_m^0$  set. The propositions will be proved only by considering subsets of  $\mathbb{N}$ , but generalize without problem to subsets of  $\mathbb{N}^n$  for arbitrary  $n$ .

**Proposition 1.6.** The  $\Sigma_m^0$  (resp.  $\Pi_m^0$ ) sets are stable under finite unions and finite intersections. ★

PROOF. Let  $m > 0$ . Be

$$\begin{aligned} A_0 &= \{n : \exists x_1 \forall x_2 \dots Qx_m R_0(n, x_1, \dots, x_m)\} \\ A_1 &= \{n : \exists y_1 \forall y_2 \dots Qy_m R_1(n, y_1, \dots, y_m)\} \end{aligned}$$

two  $\Sigma_m^0$  sets, where  $Q$  is the symbol  $\exists$  if  $m$  is odd and  $\forall$  if  $m$  is even. We leave it to the reader to show, by inclusion in one direction and then in the other, that:

$$A_0 \cap A_1 = \left\{ n : \begin{array}{l} \exists \langle x_1, y_1 \rangle \forall \langle x_2, y_2 \rangle \dots Q \langle x_m, y_m \rangle \\ \text{such that } R_0(n, x_1, \dots, x_m) \wedge R_1(n, y_1, \dots, y_m) \end{array} \right\}.$$

The  $\Sigma_m^0$  sets are therefore stable under finite intersections. We also have in the same way:

$$A_0 \cup A_1 = \left\{ n : \begin{array}{l} \exists \langle x_1, y_1 \rangle \forall \langle x_2, y_2 \rangle \dots Q \langle x_m, y_m \rangle \\ \text{such that } R_0(n, x_1, \dots, x_m) \vee R_1(n, y_1, \dots, y_m) \end{array} \right\}.$$

The  $\Sigma_m^0$  sets are therefore stable under finite union. By passing to the com-

plement, the  $\Pi_m^0$  sets are also stable under finite unions and intersections. ■

Let us remember the concept of uniformly computable sequence, introduced on the occasion of the developments which follow Lemma 4-7.2. The concept of uniformity passes to the arithmetic hierarchy as follows: in the next proposition, a countable union  $(A_i)_{i \in \mathbb{N}}$  of sets *uniformly*  $\Sigma_m^0$  is therefore a union such that each set  $A_i$  admits the same  $\Sigma_m^0$  description, but with  $i$  as a parameter. Formally, if

$$A_i = \{n : \exists x_1 \forall x_2 \dots Qx_m R_i(n, x_1, \dots, x_m)\},$$

there must be a computable process allowing from  $i$  to return a code for the predicate  $R_i$ . Equivalently, we can consider that  $A_i$  is described as

$$A_i = \{n : \exists x_1 \forall x_2 \dots Qx_m R(i, n, x_1, \dots, x_m)\},$$

where  $R$  is a computable predicate taking  $i$  as an additional parameter.

**Proposition 1.7.** The  $\Sigma_m^0$  (resp.  $\Pi_m^0$ ) sets are stable under uniform countable unions (resp. uniform countable intersections). ★

PROOF. Let  $m > 0$  and let  $(A_i)_{i \in \mathbb{N}}$  be a uniform sequence of  $\Sigma_m^0$  sets:

$$A_i = \{n : \exists x_1 \forall x_2 \dots Qx_m R(i, n, x_1, \dots, x_m)\},$$

where  $Q$  is the symbol  $\exists$  if  $m$  is odd and  $\forall$  if  $m$  is even. It is easily shown by mutual inclusion that

$$\bigcup_i A_i = \{n : \exists \langle i, x_1 \rangle \forall x_2 \dots Qx_m R(i, n, x_1, \dots, x_m)\}.$$

The  $\Sigma_m^0$  sets are therefore stable under uniform countable unions. By passing to the complement, the  $\Pi_m^0$  sets are themselves also stable under countable uniform intersections. ■

Note that the stability of  $\Sigma_n^0$  sets by uniform countable union is equivalent to stability under existential quantification (resp. universal quantification for  $\Pi_n^0$  sets).

**Exercise 1.8.** Show that the  $\Sigma_n^0$  sets are not stable under non-uniform countable unions. ◇

We now show the stability under uniform bounded quantification. A bounded quantification of the form  $\forall x < m$  can be seen as a finite union of  $m$  sets parameterized by  $x$ . The following proposition is however not identical to Proposition 1.6: here we want uniformity as a function of the bound which can itself be a variable, or depend on other variables on which we quantify.

Roughly speaking, what the following proposition says is that the set

$$\{n : \exists x \forall y < x \exists t R(n, x, y, t)\},$$

where  $R$  is a computable predicate, can be rewritten as

$$\{n : \exists x \exists t \forall y < x \exists s < t R(n, x, y, s)\}.$$

The predicate  $\forall y < x \exists s < t R(n, x, y, s)$  being computable, the set is  $\Sigma_1^0$ .

**Proposition 1.9.** The  $\Sigma_m^0$  and  $\Pi_m^0$  sets are stable under uniform bounded quantifications. ★

**PROOF.** We show that we can always move the bounded quantification to the right, until it is found next to the computable predicate.

Let  $A = \{(n, k) : \forall y < k \exists x R(n, y, x)\}$ , where  $R$  is a computable or  $\Pi_m^0$  set for  $m > 0$ . So we also have

$$A = \{(n, k) : \exists x \forall y < k \exists z < x R(n, y, z)\}.$$

The equality comes from the fact that if for all  $y < k$  a certain  $x_k$  witnesses that a formula is true, since the number of witnesses is finite, these are bounded in  $\mathbb{N}$ . The variable  $x$  which previously served as a witness is now used as a bound on all possible witnesses. We have in the same way by passing to the complement

$$\{(n, k) : \exists y < k \forall x R(n, y, x)\} = \{(n, k) : \forall x \exists y < k \forall z < x R(n, y, z)\},$$

where  $R$  is a computable or  $\Sigma_m^0$  set.

If we now have  $A = \{(n, k) : \exists y < k \exists x R(n, y, x)\}$ , where  $R$  is a computable or  $\Pi_m^0$  set, then we also have  $A = \{(n, k) : \exists x \exists y < k R(n, y, x)\}$  immediately. We have in the same way by passing to the complement  $\{(n, k) : \forall y < k \forall x R(n, y, x)\} = \{(n, k) : \forall x \forall y < k R(n, y, x)\}$ , where  $R$  is a computable or  $\Sigma_m^0$  set.

By induction, we thus move the bounded quantifications to the right until they are all stuck to the computable predicate. Finally, we use the fact that the computable predicates are stable under bounded quantifications (see Exercize 3-2.4) to conclude. ■

## 2. Arithmetic hierarchy and computability

Now let's see what the first levels of the arithmetic hierarchy correspond to.

**Proposition 2.1.** A set  $A \subseteq \mathbb{N}$  is  $\Sigma_1^0$  iff it is computably enumerable. ★

PROOF. Suppose that the set  $A$  is computably enumerable. Then,  $A = \{n : \exists t \Phi_e(n)[t] \downarrow\}$  for some  $e$ . The predicate  $\Phi_e(n)[t] \downarrow$  being computable,  $A$  is  $\Sigma_1^0$ . Suppose now that  $A$  is  $\Sigma_1^0$ . Let  $A = \{n : \exists t R(n, t)\}$ , where  $R$  is a computable predicate. Then, we easily define the code machine  $e$  which on the input  $n$  searches for the smallest  $t$  such that  $R(n, t)$  and halts, or continues its search indefinitely otherwise. We then have  $\Phi_e(n) \downarrow$  iff  $\exists t R(n, t)$ . ■

**Proposition 2.2.** A set  $A \subseteq \mathbb{N}$  is  $\Delta_1^0$  iff it is computable. ★

PROOF. A set  $A$  is  $\Delta_1^0$  iff  $A$  and  $\mathbb{N} \setminus A$  are  $\Sigma_1^0$  iff  $A$  and  $\mathbb{N} \setminus A$  are computably enumerable (according to Proposition 2.1), or equivalently if  $A$  is computable (according to Proposition 3-7.4). ■

We have seen that there are computably enumerable sets which are not computable, and in particular whose complement is not computably enumerable. This implies that some sets are  $\Sigma_1^0$  but not  $\Delta_1^0$ , and in particular not  $\Pi_1^0$ . We will see in the next sections that the hierarchy is strict everywhere: for all  $n$ , there are  $\Sigma_n^0$  sets which are not  $\Pi_n^0$ , and there are  $\Delta_{n+1}^0$  sets which are not neither  $\Sigma_n^0$ , nor  $\Pi_n^0$ .

Intuitively, the number of quantifiers corresponds to the “number of times it would take to infinity” to determine the membership of an element in the set. Thus, for a  $\Sigma_1^0$  set written as  $\{n : \exists t R(t, n)\}$ , where  $R$  is a computable predicate, it would be necessary to test the truth value of  $R(t, n)$  for all integers  $t$ , so to find one which witnesses the belonging of  $n$  to the set, or else in order to be sure that  $n$  does not belong to it.

For a  $\Pi_2^0$  set of the form

$$\{n : \forall t_1 \exists t_2 R(t_1, t_2, n)\},$$

we would need a first procedure which tests all the integers  $t_1$ , and which for each of these tests, examines the truth value of  $R(t_1, t_2, n)$  for all the integers  $t_2$ . This would sort of correspond to two nested loops each ranging over the set of all integers.

### 3. Relativization to an oracle

The arithmetic hierarchy makes it possible to define more and more complex sets, and in a sense less and less computable. However, the class of arithmetically definable sets remains countable. So there remain “many”

sets, so to speak the majority, which cannot be computed, nor even defined by an arithmetic formula. It is obviously difficult to speak about them, and any attempt to define them more precisely would make them definable in a certain language, widening only a little more the inevitably countable class of the sets of which one manages to say something, leaving aside the majority of the other sets, hidden, inaccessible.

We'll work around this problem throughout the next few chapters, primarily by looking at “groups of sets” rather than each set individually. We will see in particular in the chapters to come many computational properties shared by certain sets, in general an uncountable quantity of them. We will then carry out in Part II a study of the typical sets from the point of view of Measure Theory, that is to say of the sets which one obtains with probability 1 if one selects their bits at random.

For the moment, we are content to relativize the arithmetic hierarchy to an oracle: given any set  $X$ , we consider the  $\Sigma_n^0$ ,  $\Pi_n^0$  and  $\Delta_n^0$  sets that we can define relative to the knowledge of  $X$ . We base ourselves for this on the oracle computations of Definition 4-2.2, and we iterate as in Definition 1.1.

**Definition 3.1.** Let  $m, n \geq 1$ . Let  $X \subseteq \mathbb{N}$ .

1. A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Sigma_n^0(X)$  if there exists an  $X$ -computable set  $R \subseteq \mathbb{N}^{n+m}$  such that

$$A = \{(y_1, \dots, y_m) : \overbrace{\exists x_1 \forall x_2 \dots Q x_n}^{n \text{ quantifiers}} (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

where  $Q$  is  $\exists$  if  $n$  is odd, and  $\forall$  if  $n$  is even.

2. A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Pi_n^0(X)$  if there exists an  $X$ -computable set  $R \subseteq \mathbb{N}^{n+m}$  such that

$$A = \{(y_1, \dots, y_m) : \overbrace{\forall x_1 \exists x_2 \dots Q x_n}^{n \text{ quantifiers}} (x_1, \dots, x_n, y_1, \dots, y_m) \in R\},$$

where  $Q$  is  $\forall$  if  $n$  is odd, and  $\exists$  if  $n$  is even. ◇

**Definition 3.2.** Let  $m \geq 1$ . Let  $X \subseteq \mathbb{N}$ . A set  $A \subseteq \mathbb{N}^m$  is said to be  $\Delta_n^0(X)$  for  $n > 0$  if it is both  $\Sigma_n^0(X)$  and  $\Pi_n^0(X)$ . ◇

The following propositions are proved in the same way as their respective equivalents of the previous section.

**Proposition 3.3.** A set  $A \subseteq \mathbb{N}$  is  $\Sigma_1^0(X)$  iff it is  $X$ -c.e. ★

**Proposition 3.4.** A set  $A \subseteq \mathbb{N}$  is  $\Delta_1^0(X)$  iff it is  $X$ -computable. ★

Note that an oracle will be able to provide additional computational power, allowing certain sets that are normally not  $\Sigma_n^0$ , to become  $\Sigma_n^0(X)$ . We can even for example define an oracle  $X$  such that all arithmetic sets, that is to say  $\Sigma_n^0$  for a certain  $n$ , become  $\Delta_1^0(X)$ . However, whatever the computational power of  $X$ , the arithmetic sets in  $X$  remain in countable quantity.

## 4. Many-one degrees

The classification of the arithmetic hierarchy in terms of  $\Sigma_n^0$  and  $\Pi_n^0$  sets is not a notion on Turing degrees, because a  $\Sigma_n^0$  set can be Turing equivalent to a set which is not. In fact, any  $\Sigma_n^0$  set  $A$  is Turing equivalent to its  $\Pi_n^0$  complement  $\bar{A}$ . In this sense, the Turing reduction is “coarse”, because it does not distinguish a set from its complement.

We are going to introduce a concept finer than the Turing reduction, in the sense that it implies the latter. This is the many-one reduction, which as we will see, preserves the arithmetic hierarchy. In fact, a lot of proofs of Turing reduction that we have seen are actually many-one reductions.

**Definition 4.1.** Let  $A, B$  be two sets. We say that  $A$  is *many-one reducible* to  $B$ , and we write  $A \leq_m B$  if there exists a total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $n \in A \leftrightarrow f(n) \in B$ . If  $A \leq_m B$  and  $B \leq_m A$ , we write  $A \equiv_m B$ . We write  $A <_m B$  if  $A \leq_m B$  but  $B \not\leq_m A$ . We call *degrees many-one* the equivalence classes of the relation  $\equiv_m$ . ◇

When  $A \leq_m B$  holds, the knowledge of  $B$  is sufficient to compute  $A$ , and we have in particular  $A \leq_T B$ : let  $f$  be the total computable function such that  $n \in A \leftrightarrow f(n) \in B$ . Then, to know if  $n \in A$ , we use  $f$  for “ask a question” to the set  $B$ : is that  $f(n) \in B$ ? If the answer is yes, then  $n \in A$ . Otherwise,  $n \notin A$ . The many-one reduction is very restrictive: if  $A \leq_m B$ , then to know a bit of  $A$  we only have the right to ask the oracle  $B$  only one question. As if that were not enough, the answer to the question directly determines whether the bit belongs to  $A$  without it being possible to reverse this decision. The importance of this very restrictive reduction comes from the fact that it preserves the arithmetic hierarchy.

**Proposition 4.2.** Let  $A \subseteq \mathbb{N}$  be a  $\Sigma_m^0(X)$  set (resp.  $\Pi_m^0(X)$ ) for a certain  $X \subseteq \mathbb{N}$ , and let  $B \leq_m A$ . Then,  $B$  is  $\Sigma_m^0(X)$  (resp.  $\Pi_m^0(X)$ ). ★

PROOF. Let  $A$  be a  $\Sigma_m^0(X)$  set. Then,

$$A = \{n : \exists x_1 \forall x_2 \dots Qx_m R(n, x_1, \dots, x_m)\},$$

where  $R$  is an  $X$ -computable set. Let  $f$  be the total computable function

such that  $n \in B$  iff  $f(n) \in A$ . So we have

$$B = \{n : \exists x_1 \forall x_2 \dots Qx_m R(f(n), x_1, \dots, x_m)\},$$

which is a  $\Sigma_m^0(X)$  description of  $B$ .

We show in the same way that if  $A$  is  $\Pi_m^0(X)$  and  $B \leq_m A$ , then  $B$  is  $\Pi_m^0(X)$ . ■

Among the  $\Sigma_1^0$  sets, the halting problem plays a particular role: it is the most “powerful” of the  $\Sigma_1^0$  sets, in the sense that any  $\Sigma_1^0$  set is many-one reducible to the halting problem.

**Proposition 4.3.** A set  $A$  is  $\Sigma_1^0$  iff  $A \leq_m \emptyset'$ . ★

PROOF. Suppose  $A$  is  $\Sigma_1^0$ . Then, there exists  $e$  such that  $n \in A$  iff  $\Phi_e(n) \downarrow$ . Using the SMT theorem, we define a computable function  $s : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $m$  we have  $\Phi_e(n) = \Phi_{s(n)}(m)$ . We will have in particular  $n \in A$  iff  $\Phi_e(n) \downarrow$ , or equivalently if  $\Phi_{s(n)}(s(n)) \downarrow$ , or even if  $s(n) \in \emptyset'$ . Then,  $A \leq_m \emptyset'$ .

Suppose now  $A \leq_m \emptyset'$ . Then, as  $\emptyset'$  is  $\Sigma_1^0$ , the set  $A$  is also  $\Sigma_1^0$ , according to Proposition 4.2. ■

We also say that  $\emptyset'$  is a  $\Sigma_1^0$ -complete set, as we will see with Definition 5.2.

## 5. Post's theorem

We will see that there is a precise correspondence between the iterations of the Turing jump and the arithmetic hierarchy.

**Definition 5.1.** Given a set  $X \subseteq \mathbb{N}$ , we define recursively on  $n \geq 0$ :

1.  $X^{(0)} = X$
2.  $X^{(n+1)} = X^{(n)'}.$  ◇

Thus,  $\emptyset^{(1)} = \emptyset'$ ,  $\emptyset^{(2)} = \emptyset''$ , etc. We are now going to show that, for all  $n$ , the set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ -complete: it is a  $\Sigma_n^0$  set, which also has a maximum computational power among the  $\Sigma_n^0$  sets, in the sense that any  $\Sigma_n^0$  set is many-one reducible to  $\emptyset^{(n)}$ .

**Definition 5.2.** A set  $A \subseteq \mathbb{N}$  is  $\Sigma_n^0(X)$ -complete (resp.  $\Pi_n^0(X)$ -complete)

if it is  $\Sigma_n^0(X)$  (resp.  $\Pi_n^0(X)$ ) and if, for any  $\Sigma_n^0(X)$  set (resp.  $\Pi_n^0(X)$ )  $B$ , we have  $B \leq_m A$ .  $\diamond$

**Proposition 5.3.** Let  $n > 0$ . The set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ -complete. Likewise, for any set  $X \subseteq \mathbb{N}$ , the set  $X^{(n)}$  is  $\Sigma_n^0(X)$ -complete.  $\star$

PROOF. Let us show by induction on  $n$  that for all  $n > 0$  the set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ . By definition, the set  $\emptyset^{(1)}$  is  $\Sigma_1^0$ . Suppose the set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ . Then, the set  $\emptyset^{(n+1)}$  is defined as:

$$\left\{ m : \exists t \in \mathbb{N} \exists \sigma \in 2^{<\mathbb{N}} \quad \left( \begin{array}{c} \forall s < |\sigma| \quad \left( \begin{array}{c} \sigma(s) = 1 \text{ and } s \in \emptyset^{(n)} \\ \sigma(s) = 0 \text{ and } s \notin \emptyset^{(n)} \end{array} \right) \end{array} \right) \text{ and } \Phi_m(\sigma, m)[t] \downarrow \right\}.$$

The description of  $\emptyset^{(n+1)}$  is therefore done with existential quantifiers, followed by a predicate using  $s \in \emptyset^{(n)}$ , which is  $\Sigma_n^0$  by induction, and using  $s \notin \emptyset^{(n)}$ , which is  $\Pi_n^0$  by induction, and therefore  $\Sigma_{n+1}^0$ . Using the stability properties of propositions 1.9 and 1.6, the predicate which follows existential quantifiers  $\exists t \in \mathbb{N} \exists \sigma \in 2^{<\mathbb{N}}$  is in particular  $\Sigma_{n+1}^0$  uniformly in  $m, \sigma$  and  $t$ . Now using the uniform countable union stability of Proposition 1.7, the set  $\emptyset^{(n+1)}$  is therefore  $\Sigma_{n+1}^0$ .

Let us now show by induction on  $n$  that any  $\Sigma_n^0$  set is many-one reducible to  $\emptyset^{(n)}$ . This is the case with Proposition 4.3 for  $n = 1$ . Suppose this is the case for some  $n$ . Let  $A = \{x : \exists y R(x, y)\}$ , where  $R$  is a  $\Pi_n^0$  set.

By induction, there exists a total computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $(x, y) \in R$  iff  $g(\langle x, y \rangle) \notin \emptyset^{(n)}$ . We define the total computable function  $f$  such that  $\Phi_{f(x)}^{\emptyset^{(n)}}$  halts on any input if  $\exists y g(\langle x, y \rangle) \notin \emptyset^{(n)}$ , and does not halt on any input otherwise. We therefore have  $f(x) \in \emptyset^{(n+1)}$  iff  $\Phi_{f(x)}(\emptyset^{(n)}, f(x)) \downarrow$ , in other words if  $\exists y g(\langle x, y \rangle) \notin \emptyset^{(n)}$ , or even if  $\exists y R(x, y)$ , or finally if  $x \in A$ . The set  $A$  is therefore many-one reducible to  $\emptyset^{(n+1)}$ .

The relativization to an oracle  $X$  is similar and does not present any particular difficulty.  $\blacksquare$

#### Corollary 5.4

A set  $A$  is  $\Sigma_n^0(X)$  iff  $A \leq_m X^{(n)}$ .

PROOF. By the previous proposition and by the definition of  $\Sigma_n^0(X)$ -completeness.  $\blacksquare$

We finally come to Post's theorem.

**Theorem 5.5 (Post's theorem)**

Let  $A$  be a set and  $n \geq 0$ .

- (1)  $A$  is  $\Sigma_{n+1}^0$  iff  $A$  is  $\Sigma_1^0(\emptyset^{(n)})$ , iff  $A$  is  $\emptyset^{(n)}$ -c.e.
- (2)  $A$  is  $\Delta_{n+1}^0$  iff  $A$  is  $\Delta_1^0(\emptyset^{(n)})$ , iff  $A \leq_T \emptyset^{(n)}$

PROOF. Let us show (1). By Corollary 5.4,  $A$  is  $\Sigma_{n+1}^0$  iff  $A \leq_m \emptyset^{(n+1)}$ . By relativizing Proposition 4.3 to  $\emptyset^{(n)}$ , we obtain  $A \leq_m \emptyset^{(n)'} (which is equal to  $\emptyset^{(n+1)}$ ) if  $A$  is  $\Sigma_1^0(\emptyset^{(n)})$ . Finally, according to Proposition 3.3,  $A$  is  $\Sigma_1^0(\emptyset^{(n)})$  iff  $A$  is  $\emptyset^{(n)}$ -c.e.$

Let us show (2). By definition,  $A$  is  $\Delta_{n+1}^0$  iff  $A$  and  $\bar{A}$  are both  $\Sigma_{n+1}^0$ . By the previous point, this is equivalent to saying that  $A$  and  $\bar{A}$  are  $\emptyset^{(n)}$ -c.e. By relativizing Proposition 3-7.4 to  $\emptyset^{(n)}$ ,  $A$  and  $\bar{A}$  are  $\emptyset^{(n)}$ -c.e. iff  $A \leq_T \emptyset^{(n)}$ . Finally, according to Proposition 3.4, we have  $A \leq_T \emptyset^{(n)}$  iff  $A$  is  $\Delta_1^0(\emptyset^{(n)})$ . ■

**Corollary 5.6**

The arithmetic hierarchy is strict. In other words,

- for all  $n > 0$ , there exists a  $\Sigma_n^0$  set which is not  $\Pi_n^0$  and a  $\Pi_n^0$  set which is not  $\Sigma_n^0$ ;
- for all  $n > 0$ , there exists a  $\Delta_{n+1}^0$  set which is neither  $\Sigma_n^0$  nor  $\Pi_n^0$ .

PROOF. According to Proposition 5.3, the set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ . It cannot be  $\Pi_n^0$  in which case it would be  $\Delta_n^0$ , and therefore computable in  $\emptyset^{(n-1)}$  according to Theorem 5.5, contradicting the fact that  $X$  never computes its Turing jump. We show the same way that  $\mathbb{N} \setminus \emptyset^{(n)}$  is  $\Pi_n^0$ , but not  $\Sigma_n^0$ .

Finally, we can construct for all  $n$  the following  $\Delta_{n+1}^0$  set: the set  $X$  such that  $X(2m) = \emptyset^{(n)}(m)$  and such that  $X(2m+1) = (\mathbb{N} \setminus \emptyset^{(n)})(m)$ . It is clear that  $X$  is  $\emptyset^{(n)}$ -computable, and therefore  $\Delta_{n+1}^0$  according to Theorem 5.5. Finally, if we assume by the absurd that  $X$  is  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ), this allows to give a  $\Sigma_n^0$  description of  $\mathbb{N} \setminus \emptyset^{(n)}$  by keeping only the odd bits of  $X$  (resp. a  $\Pi_n^0$  description of  $\emptyset^{(n)}$  keeping only the even bits of  $X$ ), which is in contradiction with the fact that  $\emptyset^{(n)}$  is not  $\Pi_n^0$  (resp. with the fact that  $\mathbb{N} \setminus \emptyset^{(n)}$  is not  $\Sigma_n^0$ ). ■

**Exercize 5.7. (\*)** Show that, for all  $X, Y \in 2^{\mathbb{N}}$ , we have  $X \equiv_T Y$  iff  $X' \equiv_m Y'$ . ◇

## 6. Rice's theorem

Rice's theorem basically says that no semantic property of programs is decidable. For example, as seen in Exercize 3-6.4, it is impossible to decide whether a computer program performs a multiplication by 2 or not. We mean *semantic properties* as opposed to *syntactic properties*. The latter are sensitive to code variations, for example “this program has three distinct variables” or “this program contains two loops **for**”. Semantic properties do not speak directly about programs, but about the functions they represent. For example, property “this function is defined on input 42” or property “this function only returns even values” do not depend on their code. Semantic properties do not depend on the implementation details of the function.

**Definition 6.1.** An *index set* is a set  $A$  included in  $\mathbb{N}$ , such that for all  $x, y \in \mathbb{N}$ ,

$$\text{if } x \in A \text{ and } \Phi_x = \Phi_y, \text{ then } y \in A.$$

Among the index sets, we will note the empty set  $\emptyset$  which corresponds to a property never satisfied, and the set  $\mathbb{N}$  representing a property that is always true. An index set is *non trivial* if  $A \neq \emptyset$  and  $A \neq \mathbb{N}$ .

### Theorem 6.2

If  $A$  is a non-trivial index set, then either  $\emptyset' \leq_m A$  or  $\emptyset' \leq_m \bar{A}$ .

PROOF. Let  $\Phi_{e_0}$  be the nowhere-defined function. Suppose that  $e_0 \in \bar{A}$ , the other case being treated by symmetry. Since  $A$  is non-trivial,  $A$  is non-empty. Let's fix a code  $e_1 \in A$ . In particular,  $\Phi_{e_0} \neq \Phi_{e_1}$ . By the SMN theorem (see Theorem 3-4.1), there exists a total and computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that

$$\Phi_{f(x)}(y) = \begin{cases} \Phi_{e_1}(y) & \text{if } x \in \emptyset' \\ \uparrow & \text{if } x \notin \emptyset'. \end{cases}$$

Let us show that the function  $f$  is a many-one reduction from  $\emptyset'$  to  $A$ . If  $x \in \emptyset'$ , then  $\Phi_{f(x)} = \Phi_{e_1}$ , and so  $f(x) \in A$ . Conversely, if  $x \notin \emptyset'$ ,  $\Phi_{f(x)} = \Phi_{e_0}$ , then  $f(x) \in \bar{A}$ . ■

Rice's theorem asserts that no non-trivial property on partial computable functions is decidable. Although the applications of this theorem are relatively limited in Computability Theory, Rice's theorem is of great importance for the understanding it brings about the nature of computability. In particular, the undecidability of the halting problem is not an isolated

phenomenon, because it is shared by all the properties on the partial computable functions.

**Corollary 6.3 (Rice's theorem)**

*Let  $\mathcal{C}$  be a class of partial computable functions  $\mathbb{N} \rightarrow \mathbb{N}$ . Then, the set  $A = \{x : \Phi_x \in \mathcal{C}\}$  is non-computable unless  $\mathcal{C} = \emptyset$  or  $\mathcal{C}$  is the class of all partial computable functions.*

PROOF. The set  $A$  is an index set. If  $\mathcal{C} = \emptyset$  or  $\mathcal{C}$  is the class of all partial computable functions, then  $A = \emptyset$  or  $A = \mathbb{N}$  and, in both cases,  $A$  is computable. If  $\mathcal{C}$  is neither empty nor the class of all partial computable functions, then  $A$  is non-trivial, and by Theorem 6.2 either  $\emptyset' \leq_m A$  or  $\emptyset' \leq_m \bar{A}$ . In both cases,  $\emptyset' \leq_T A$ , and  $A$  is not computable. ■

## 7. Arithmetic codes

Sets of integers are generally infinite objects, and therefore cannot be represented by natural integers without some of them being omitted from this representation. This is the object of Cantor's diagonal argument (see Section 2-4). Certain sets of integers can however be described in a finite way, starting with the computable sets.

**Definition 7.1.** A  $\Delta_1^0$  code of a computable set  $A$  is an integer  $e$  such that  $\Phi_e = A$  (ie  $\forall n \Phi_e(n) \downarrow = A(n)$ ). ◇

Note that a set is computable iff it has a  $\Delta_1^0$  code. By the Padding lemma 3-5.1, any computable set is represented by an infinity of  $\Delta_1^0$  codes. According to Rice's theorem, the set of  $\Delta_1^0$  codes of a fixed computable set is not decidable. There is not even a procedure for deciding whether an integer  $e$  is a  $\Delta_1^0$  code of a set. Indeed, that would amount to being able to enumerate in a computable way all the computable sets, and would leave place to Cantor's diagonal argument.

**Exercise 7.2. (★)** Let TOT be the set of  $\Delta_1^0$  codes, in other words

$$\text{TOT} = \{e : \Phi_e \text{ is total}\}$$

(assuming without loss of generality that  $\Phi_e(n)$  returns 0 or 1 for all  $n$ ). Prove that TOT is  $\Pi_2^0$ -complete, ie. that  $\mathbb{N} \setminus \emptyset'' \leq_m \text{TOT}$ . ◇

The  $\Delta_1^0$  codes allow to give a new definition of a uniformly computable sequence of sets. Recall that a sequence  $X_0, X_1, \dots$  of sets is uniformly computable if the function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  defined by  $f(x, s) = 1$  iff  $x \in X_s$  is total computable.

**Proposition 7.3.** A sequence  $X_0, X_1, \dots$  of sets is uniformly computable if, and only if, there exists a computable sequence  $e_0, e_1, \dots$  such that  $e_s$  is a  $\Delta_1^0$  code of  $X_s$  for all  $s$ . ★

PROOF.  $\Rightarrow$ . Suppose that  $X_0, X_1, \dots$  is uniformly computable through the function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$ . By the SMN theorem, there exists a total computable function  $h : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $e, x$ , we have the equality  $\Phi_{h(e)}(x) = f(x, e)$ . It follows that the sequence  $h(0), h(1), \dots$  is a computable sequence such that  $h(s)$  is a  $\Delta_1^0$  code of  $X_s$ .

$\Leftarrow$ . Suppose now that there exists a computable sequence  $e_0, e_1, \dots$  of  $\Delta_1^0$  codes. Then, given the existence of a universal machine, the function  $f$  defined on  $\mathbb{N} \times \mathbb{N}$  by  $f(x, s) = \Phi_{e_s}(x) \in \mathbb{N}$  is total computable, and shows that the sequence of sets  $X_0, X_1, \dots$  is uniformly computable. ■

Computationally enumerable sets are also representable by a code system. The following notation will often be used for this.

#### Notation

We denote by  $W_e$  the set  $\{n \in \mathbb{N} : \Phi_e(n) \downarrow\}$ , which is computably enumerable. The notation is relativized to an oracle  $X$ , and  $W_e^X$  or  $W_e(X)$  will thus denote the set  $\{n \in \mathbb{N} : \Phi_e(X, n) \downarrow\}$ .

**Definition 7.4.** A  $\Sigma_1^0$  code of a c.e. set  $A$  is an integer  $e$  such that  $W_e = A$ . ◇

Still by Rice's theorem, the set of  $\Sigma_1^0$  codes of a fixed c.e. set is not computable. However, unlike  $\Delta_1^0$  codes, *any integer* is a  $\Sigma_1^0$  code. The computable sets being *a fortiori* computably enumerable, they have both  $\Delta_1^0$  and  $\Sigma_1^0$  codes. The representation by  $\Delta_1^0$  codes is however computationally more informative, in the sense that it gives access to more information on the set that it represents. In particular, the partial function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  such that if  $e$  is a  $\Delta_1^0$  code of a set  $A$ , is defined for  $x \in \mathbb{N}$  by  $f(e, x) \downarrow = 1$  if  $x \in A$  and  $f(e, x) \downarrow = 0$  if  $x \notin A$ , is computable<sup>1</sup>, while the equivalent function for the  $\Sigma_1^0$  codes is not.

**Exercise 7.5.** Show that there is no function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  which is total, computable and such that  $f(e, x) = 1$  if  $x \in W_e$  and  $f(e, x) = 0$  if  $x \notin W_e$ . ◇

In general, we can represent any set of the arithmetic hierarchy by integers using Post's theorem.

---

<sup>1</sup>If  $e$  is not a  $\Delta_1^0$  code, the function  $f$  still acts as if it were the case, and we will eventually have  $f(e, x) \uparrow$ .

**Definition 7.6.** A  $\Sigma_{n+1}^0$  *code* (resp.  $\Delta_{n+1}^0$ ) of a set  $A$  is an integer  $e$  such that  $W_e(\emptyset^{(n)}) = A$  (resp.  $\Phi_e(\emptyset^{(n)}) = A$ ).  $\diamond$

**Remark**

Given a set  $A$ , we will tend to favor the type of code which provides the most computational information on  $A$ . Thus, if  $A$  is computable, it is often better to manipulate a  $\Delta_1^0$  code rather than a  $\Sigma_1^0$  code.

This recommendation also applies to more elaborate computability-theoretic concepts, such as low sets. As a reminder, a set  $A$  is low if  $A' \leq_T \emptyset'$ . As  $A \leq_T A' \leq_T \emptyset'$ , the low sets are in particular  $\Delta_2^0$ , and can therefore be represented by a  $\Delta_2^0$  code, that is to say an integer  $e$  such that  $\Phi_e(\emptyset') = A$ . However, this representation loses information specific to low sets (see Exercise 7.11), such as the ability of  $\emptyset'$  to decide  $A'$ . It is therefore preferable to represent the set  $A$  by a code  $e$  such that  $\Phi_e(\emptyset') = A'$ .

**Definition 7.7.** A *lowness code* of a set  $A$  is an integer  $e$  such that  $\Phi_e(\emptyset') = A'$ .  $\diamond$

Finally, in the case of finite sets, it is possible to store more information than the simple fact of being computable. Indeed, given a sequence  $e_0, e_1, \dots$  of  $\Delta_1^0$  codes of finite sets, it is not possible to compute uniformly the cardinality of these sets (see Exercise 7.10). We will therefore prefer the notion of canonical code which notably contains information on the size of the set.

**Definition 7.8.** The *canonical code* of a finite set  $F$  is the natural integer  $\sum_{i \in F} 2^i$ .  $\diamond$

We can easily verify via the following exercise that a canonical code contains all the information of a finite set, including the possibility of knowing its last element.

**Exercise 7.9.** Let  $D_0, D_1, \dots$  be the sequence of finite sets such that  $D_n$  has canonical code  $n$ .

1. Show that the function  $f : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(n) = |D_n|$  is computable.
2. Show that the function  $g : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  defined by  $g(n, x) = 1$  iff  $x \in D_n$  is computable.

$\diamond$

The following two exercises allow us to show that the information given to us by canonical codes of finite sets or by lowness codes cannot be obtained via computable or  $\Delta_2^0$  codes.

**Exercise 7.10. (\*\*)** Suppose absurdly that there exists a partial computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that if  $e$  is a  $\Delta_1^0$  code of a finite set, then  $f(e)$  returns the size of this set. Using the fixed point theorem, show that there exists a  $\Delta_1^0$  code  $a$  of a finite set such that the size of this set is different from  $f(a)$ .  $\diamond$

**Exercise 7.11. (\*\*)** Suppose absurdly that there exists a partial computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that if  $e$  is a  $\Delta_2^0$  code of a low set  $X$ , then  $f(e)$  returns a  $\Delta_2^0$  code for  $X'$ . Using the fixed point theorem, show that there exists a  $\Delta_2^0$  code  $a$  of a computable set  $X$  such that  $f(a)$  is the  $\Delta_2^0$  code of a different set of  $X'$ .  $\diamond$



# Chapter 6

## Church-Turing thesis

### 1. The Entscheidungsproblem and the quest for the Grail

In the year 1928, in a period troubled by deep questioning of the foundations of mathematics, David Hilbert and Wilhelm Ackermann asked the question of the existence of an algorithm allowing to decide the validity of any mathematical statement. Here, the word *algorithm* is to be taken in a broad sense, to designate a set of elementary computation steps that a mathematician can perform. This challenge raised to logicians, and passed down to posterity under the name of *Entscheidungsproblem* — decision problem —, marks the beginning of a long foundational quest on the formalization of the notion of algorithm.

The incompleteness theorems of Gödel<sup>1</sup> proved in 1931, stating the existence of fundamentally undecidable statements in any reasonable theory allowing formalization of arithmetic, were particularly unwelcome in a period which sought to initiate a new wave of optimism and confidence in mathematics. They also tipped the scales towards the existence of a negative solution to the Entscheidungsproblem.

A positive response to the Entscheidungsproblem would have been an algorithm or a series of deterministic steps, allowing to demonstrate any mathematical statement or its negation. A negative answer consists in the proof that such a systematic method cannot exist. This direction poses a completely different problem, namely to formally define the concept of

---

<sup>1</sup>See Chapter 9

algorithm, or in a roughly equivalent way, to find a robust and consensual formalization of the concept of computable function.

It should be noted that the first computer, the ENIAC, was built in 1940, a decade after the formulation of the Entscheidungsproblem. The notion of effectively computable function therefore does not refer to what can be computed by a computer, but by the human mind. It was a question of finding a systematic process, or algorithm in the informal sense of the term, allowing a mathematician to answer any mathematical question.

Several definitions were proposed in the years that followed, each conjectured more or less convincingly as exactly capturing the epistemological notion of effectively computable function. These definitions were fairly quickly proved to be equivalent, but struggled to convince the scientific community of their capacity to capture all the functions that could be effectively computed. It was not until 1936 that Alan Turing came to a consensus by presenting his computational model, the *Turing machine*, with a striking demonstration of its equivalence with other models, in particular with the one used by Gödel to show his famous incompleteness theorem, thus providing a definitive answer to the Entscheidungsproblem. The reader interested in the history of Computability Theory will find an excellent chapter dedicated to the subject in the work *Turing computability: Theory and Applications* by Robert Soare.

Although the different computational models have since been proven to be equivalent, we will detail the main among those which marked this history, each one presenting its own interest, by emphasizing a different aspect of the notion of computable function. In what follows, we will only consider partial functions, from  $\mathbb{N}^n$  to  $\mathbb{N}$  for  $n \geq 1$ . We will write  $g(\bar{x}) \downarrow = y$  to mean that  $g$  is defined on  $\bar{x} \in \mathbb{N}^n$  and is equal to  $y$ ; the notation  $\bar{x}$  being a shorthand for  $x_1, \dots, x_n$ .

**General recursive functions.** Recursive functions were introduced in a restricted form by Gödel as part of his incompleteness theorems in 1931. This class of functions was generalized by Gödel and Herbrand in 1934 to obtain the general recursive functions, capturing according to the thesis of Church-Turing — thesis detailed in the following section — the entirety of the computable functions.

The idea underlying the definition of general recursive functions is very simple: start from a few elementary functions, the computability of which leaves no room for doubt, then combine them to obtain new, more complex functions, which are still computable. What are the valid combinations for creating new functions? If two functions are computable, their composition should naturally be computable: it suffices to execute in series the steps

of computing each of the functions. In a slightly less obvious way, functions defined by recursion from computable functions are still computable. Indeed, if we define a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  by  $f(0) = v$  for an integer  $v$ , and  $f(n+1) = g(n, f(n))$  for a computable function  $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , to obtain the value of  $f(n)$ , it suffices to successively compute  $f(1), f(2), \dots, f(n)$  using the preceding values and by following the steps of computing the function  $g$ . Finally, if a function  $g : \mathbb{N} \rightarrow \mathbb{N}$  is computable, the search for the smallest input  $n$  such that  $g(n) = 0$  is also computable, via a loop which increments  $n$  until having  $g(n) = 0$  (this research may never be successful).

**Definition 1.1.** The class  $\mathcal{C}$  of *general recursive functions* is the smallest class of partial functions containing the following basic functions:

- (a) The successor function:  $\text{succ}(x) = x + 1$
- (b) The constant functions:  $c_m^n(x_1, \dots, x_n) = m$  for all  $n, m \geq 0$
- (c) The projections:  $p_i^n(x_1, \dots, x_n) = x_i$  for all  $n$  and all  $i \in 1, \dots, n$

and which is closed under the following operations:

- (i) Composition: if  $g_1, \dots, g_m \in \mathcal{C}$  are functions of  $n$  variables and  $h \in \mathcal{C}$  is a function of  $m$  variables, then the function

$$f(\bar{x}) = h(g_1(\bar{x}), \dots, g_m(\bar{x}))$$

is in  $\mathcal{C}$ .

- (ii) Primitive recursion: if  $g, h \in \mathcal{C}$  are partial functions of respectively  $n$  and  $n+2$  variables, the function  $f$  defined by  $f(\bar{x}, 0) = g(\bar{x})$  and  $f(\bar{x}, m+1) = h(\bar{x}, m, f(\bar{x}, m))$  is a partial function of  $n+1$  variables in  $\mathcal{C}$ .
- (iii) Minimization: if  $g \in \mathcal{C}$  is a partial function of  $n+1$  variables, then the partial function  $f$  which to  $\bar{x}$  associates the smallest integer  $m$ , if it exists, such that  $g(\bar{x}, i) \downarrow$  for all  $i \leq m$  and such that  $g(\bar{x}, m) = 0$ , is in  $\mathcal{C}$ . If  $m$  does not exist, then  $f$  is not defined on  $\bar{x}$ .

The class of *primitive recursive functions* is the smallest class of total functions containing the functions of (a), (b), (c) and closed under the schemes (i) and (ii) of composition and primitive recursion. ◇

General recursive functions have the advantage of highlighting the closure properties of the notion of computable function, starting with closure under composition. Note that the primitive recursive functions are total, unlike the general recursive functions, for which the minimization scheme introduces a possibility of partiality. Intuitively, its programming implementation would consist of a loop **while** exhaustively searching for an integer  $m$

satisfying the property. If this integer does not exist, program execution will never exit the loop, and the program will not halt. We will study this model in detail in Section 3 by trying to show that it corresponds well to the notion of effectively computable function. Let us mention before that two other computation models.

**$\lambda$ -calculus.** Defined by Church in the 1930s,  $\lambda$ -calculus is a minimalist formalism serving as a theoretical foundation for programming languages. Unlike general recursive functions, which handle two types of objects, namely integers and functions on integers,  $\lambda$ -calculus has only one primitive object: the  $\lambda$ -functions. These do not take integers as parameters, but  $\lambda$ -functions. It will therefore be necessary to resort to the coding of integers by  $\lambda$ -functions to define a notion of computable function on integers.

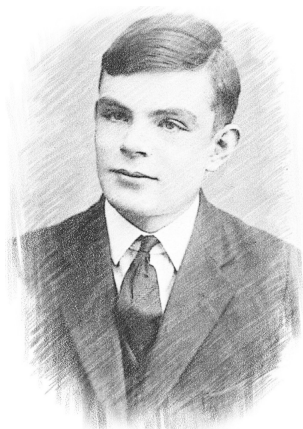
$\lambda$ -functions are defined by an expression language, as it is often the case in mathematics. For example,  $x, y \mapsto x + y$  is an expression defining the sum of two integers. However, in  $\lambda$ -calculus, the parameters themselves being  $\lambda$ -functions, the only valid operations are those for manipulating functions, namely, adding a parameter to a function, and the application of a function to its parameters. For example,  $f \mapsto (g \mapsto f(g))$  defines the function which takes as parameter a function  $f$ , and returns the function which takes as parameter a function  $g$ , and returns the result of applying  $f$  to  $g$ .

The minimalist aspect of  $\lambda$ -calculus facilitates reasoning on the formalism itself, but requires much more work to define non-trivial functions on integers. In particular, it is more difficult to convince oneself that all the computable functions can be represented by  $\lambda$ -functions. As expressed previously, it is necessary to fix a convention to represent the integers. It seems quite natural to identify the integer  $n$  with the  $\lambda$ -function taking as input a  $\lambda$ -function  $f$  and returning its  $n$ -th iteration. For example, the integer 0 is represented by the function which takes as input a function  $f$ , and returns its 0th iteration, in other words returns the function identity. In the formalism of  $\lambda$ -calculus, it is written  $f \mapsto (x \mapsto x)$ . Likewise, the integer 2 corresponds to the function  $f \mapsto (x \mapsto f(f(x)))$ . Let us call  $\bar{n}$  the  $\lambda$ -function representing the integer  $n$ . A function  $g : \mathbb{N} \rightarrow \mathbb{N}$  is  $\lambda$ -definable if there exists a  $\lambda$ -function  $h$  which to  $\bar{n}$  associates  $\overline{g(n)}$ .

The syntax of  $\lambda$ -calculus is rather abstruse at first glance, and requires a little manipulation to become familiar with the concepts. Nowadays, many variations and enrichments are studied in order to provide a theoretical basis for functional programming languages. This is a very active area of research.

**Turing machines.** If  $\lambda$ -calculus provides a theoretical basis for programming languages, Turing machines can be seen as precursors of the modern computer.

Designed in 1936 by Alan Turing, this machine was inspired by his father's typewriter. It has a *tape*, which can be seen as a succession of *cells* indexed by integers. We can make the analogy with bits which are accessed in the memory of a computer via an addressing system. The machine also has a read/write head which can move from one cell to another, then read or modify its content. The machine will run based on an input  $n$ . At the start of the computation, the read head is at the start of the tape, the first  $n$  cells of which are initialized to 1, while the remaining cells are equal to 0.



Alan Turing, 1912–1954

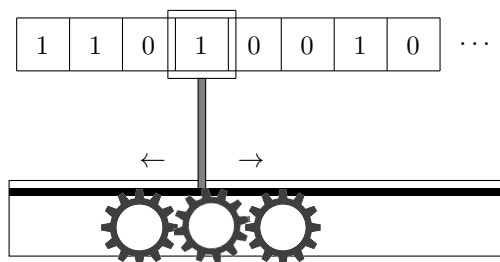


Figure 1.3: Representation of a Turing machine, with its head, moving along the cells of the tape

The machine also has a finite number of *states*, including a start state in which it is at the start of the computation, as well as a final state which indicates the end of the computation when the machine is in it. The movements of the read head, the replacement of the value of the current cell, as well as the changes of states, are subject to a set of instructions represented as follows: given the current state of the machine and the value from the cell where the head is located, a rule will decide the next state of the machine, possibly change the value of the cell, and move the head one cell to the right or to the left. A Turing machine is therefore an elaborate automaton designed to perform a specific task. In his founding article, Turing demonstrated the existence of a *universal* Turing machine: a machine able to simulate the computation of any other Turing machine.

Unlike general recursive functions or  $\lambda$ -calculus, Turing's model constitutes a very concrete mechanical process. We can in fact actually build a univer-

sal Turing machine. This model has in particular the advantage of making explicit the notion of atomic step of computation as well as of quantifying the memory space used. For these reasons, Turing machines are taken as a reference model to define the theory of Algorithmic Complexity.

**Getting away from models.** Like a quote from Michael Fellows<sup>2</sup>, Computability Theory is no more the study of computational models than astronomy is the science of telescopes. Computability is very quickly freed from the details of implementation of computational models, to manipulate programs in an abstract way. It is nevertheless instructive to see in detail at least one model of computation, in particular to forge an intuition when it is lacking, and this is what we will do very soon in Section 3.

## 2. Church-Turing thesis

In 1934, facing the success of  $\lambda$ -calculus, Church submitted the idea to Gödel that  $\lambda$ -definable functions would capture the notion of an effectively computable function, leaving Gödel doubtful. With the development of the general recursive functions of Herbrand and Gödel the same year and the proof of their equivalence with the  $\lambda$ -calculus, Church publicly formulated his thesis, known as *Church thesis*, asserting that general recursive functions coincided with effectively computable functions. His argument did not convince Gödel, although he was one of the instigators of general recursive functions. With his eponymous machine model, Alan Turing finally reached consensus in 1936, by demonstrating that Turing machines were equivalent to  $\lambda$ -calculus model and to general recursive functions, which led to what is called today the Church-Turing thesis.

### Thesis 2.1 (Church-Turing).

The effectively computable functions are those computable by a Turing machine, or equivalently the general recursive functions or the  $\lambda$ -definable functions. ■

If Church was the first to formulate the thesis according to which the  $\lambda$ -definable functions corresponded to the effectively computable functions, we generally attribute the fathership of Computability Theory to Turing. The Church-Turing thesis not being a mathematical statement, it is not possible to prove it formally. It can nevertheless be validated by what comes closest to a proof in the social sense of the term, that is to say by an argument generating a consensus in the scientific community. This is what Turing achieved by the following proof.

---

<sup>2</sup> “Computing is no more the study of computers than astronomy is that of telescopes.”  
[58]

**Turing's proof.** To justify his thesis, Turing resorted to three types of arguments: (1) a description of the process by which a mathematician performs a computation and its formalization by a Turing machine, (2) the proof of the equivalence of Turing machines with existing computational models, (3) the explicit development of large classes of functions computable by Turing machines. Here is the outline of Turing's first argument:

Consider a mathematician, Mr. Smith, performing a computation. Mr. Smith has a pencil, and a potentially unlimited amount of paper. Given the finite precision of his pencil, each sheet can only contain a finite number of symbols. For simplicity, and without loss of generality, we can consider that each sheet is a cell of an infinite tape, containing only one symbol belonging to a sufficiently large finite alphabet. The computation process is as follows: while he is in a mental state  $e_0$ , Mr. Smith is located in front of the current sheet, in his field of vision. He reads the notes, before correcting them, erasing and changing the symbol written on the sheet. This reading will change his thoughts, and he will find himself in a mental state  $e_1$ . He will potentially turn the page, or go back to reread previous notes, until he reaches the end of his computation. Mr. Smith will then consider his computation finished, and will find himself in the corresponding mental state which we will call *final state*.

**Proper use of the Church-Turing thesis.** It is important to get a clear idea of what the Church-Turing thesis says, its limits and its use. Church-Turing's thesis is neither a theorem nor a conjecture. It cannot be formally proven or refuted, by the simple fact that it is not a mathematical statement, but rather a bridge between a mathematical concept and an epistemological concept. This thesis is not, however, an arbitrary assertion, for it is supported by reasoning which can be subject to criticism.

If Church-Turing's thesis can be called into question, and even one day mostly rejected, the development of Computability Theory nonetheless rests on solid foundations, independent of this correspondence. The equivalence between functions computable by Turing machines, general recursive functions,  $\lambda$ -definable functions, and functions programmable in C, Java or Python, is indeed a theorem which does not depend on the Church-Turing thesis. While it is common in Computability Theory to informally describe an algorithm and then deduce the existence of a Turing machine implementing it, this process is not strictly speaking a call to the Church-Turing thesis. Rather, it is an informal proof to convince the interlocutor that, if necessary, it would be easy to program this algorithm in any programming language.

Moreover, if Church-Turing's thesis were to be invalidated, the conceptual edifice built around Computability Theory would remain, and would likely

continue to be studied. There is a hierarchy of formal languages and computational models, called *Chomsky hierarchy*. We find there for example the *rational languages*, corresponding to the languages recognized by a class of machines called *finite automata*. Although these models are for the most part less expressive than the Turing machines, they are nonetheless a very active subject of research today. If one day new computational paradigms were found, the existing notion of computable function would nonetheless remain a very robust class of functions, and would likely continue to be studied in the same way as rational languages or any other level of Chomsky hierarchy.

Some natural phenomena are studied in the hope of solving non-computable problems. These notions of computability are united under the name of *hypercomputing* (we will see a formal approach of it in Part IV). To date, there is no prospect of making such computations. The discovery of new computational phenomena in nature would however probably not invalidate the Church-Turing thesis because they would have little chance of satisfying the definition of an effectively computable function according to Rosser [183], i.e., say “a method in which each step is precisely predetermined and which will reliably produce an answer in a finite number of steps.”

### 3. Detailed study of recursive functions

The objective of this section is to convince the reader that general recursive functions coincide with the effectively computable functions as defined informally in sections 3-2 and 3-1, that is, “functions that can be programmed”. The interests of such a study are manifold. In the first place, it makes it possible to have a precise mathematical definition of what a computable function is: without loss of generality, it is a general recursive function. Then the study which we give will provide at the same time a mathematical proof of the existence of a universal machine such as defined in Theorem 3-3.1, concept used throughout this work. Finally, our study will isolate the notion of primitive recursive function as a strict subclass of computable functions; besides a certain epistemological importance (see Theorem 3.22), this subclass presents an undeniable interest in mathematical logic. We will see an example of its use in the study of Reverse Mathematics, in Section 23-7.

#### 3.1. Register machines and programming diagrams

In order to convince ourselves that the general recursive functions coincide with the computable functions, we start from a computational model close to modern programming languages: *structured programs*, executed by

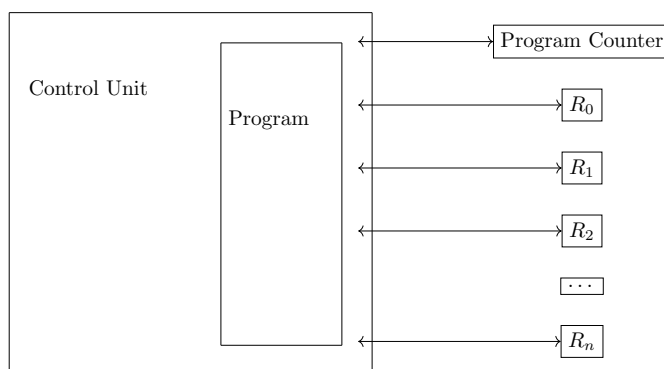


Figure 3.1: Model of register machines

*register machines.* The developments in this section comes in broad outline from the Computability Theory course of Arnaud Durand and Paul Rozière [50] of the Master of mathematical logic of the University of Paris Diderot.

**Register machines.** The scientific literature has declined several versions, under the name of “random access machine” (Melzak [154], Lambek [134], Shepherdson and Sturgis [196], Peter [173], Elgot and Robinson [54]). These are machines known as “random access memory” or RAM, a generic name today to designate the random access memory of computers. The term “random access” should be understood as opposed to “sequential access”, and refers to the fact that each memory slot can be accessed directly from its address, unlike, for example, the model of Turing machines, for which a read head must move cell after cell in memory to arrive at the desired location.

The memory of a register machine will however be more elementary than modern RAM: it is simply a finite number of *registers*  $R_0, R_1, \dots, R_k$  for  $k \in \mathbb{N}$  arbitrary. Each register can contain any positive or zero integer. Note that the integers can be arbitrarily large, and therefore that each register represents an “unbounded” memory space.

**Structured programs.** A register machine will execute a program which consists of a finite list of instructions for performing computations. There are many possible variations on the instruction set that one allows oneself. We present one deliberately close to that of a modern imperative programming language.

**Definition 3.2.** A *structured program* can contain the following instructions:

1.   • Incrementation of a register: “ $R_i := R_i + 1$ ”.
- Decrementation of a register “ $R_i := R_i - 1$ ”.
- Assignment of a register: “ $R_i := x$ ” for  $x \in \mathbb{N}$  or “ $R_i := R_j$ ”

*These instructions respectively increment the value of  $R_i$  by 1, decrement it by 1 (unless the value is 0 in which case nothing happens), set the value of  $R_i$  to  $x$  or set it to that of  $R_j$ ).*

2. The conditional statement: “if  $R_i = 0$  then  $S$  else  $S'$ ”, where  $S = (S_0, \dots, S_n)$  and  $S' = (S'_0, \dots, S'_m)$  are finite sequences of structured instructions.

*Each instruction in  $S$  is executed sequentially if  $R_i$  is equal to 0. Otherwise each instruction in  $S'$  is executed sequentially.*

3. The for loop statement: “for  $i = 1$  to  $R_i$  do  $S$ ”, where  $S = (S_0, \dots, S_n)$  is a finite sequence of structured instructions.

*Let  $N$  be the number present in register  $R_i$  when the program starts this instruction. Each  $S$  instruction is executed sequentially, all  $N$  times. Note that if the value of  $R_i$  changes while the instructions of  $S$  are being executed, this does not change the number of times the loop occurs.*

4. The while loop statement: “while  $R_i \neq 0$  do  $S$ ” where  $S = (S_0, \dots, S_n)$  is a finite sequence of structured instructions.

*Each instruction in  $S$  is executed sequentially as long as the register  $R_i$  is different from 0.*

The execution of a structured program halts after the last instruction has been executed. ◇

Note that a structured program has only a finite number of instructions and can therefore only use a finite number of registers.

#### — for loop vs while loop —

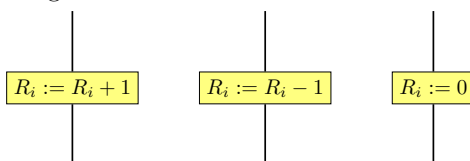
The reader may notice that the **for** loop statement is redundant in that it can always be replaced by a **while** loop statement. We will see that the reverse is not true. In particular the number of times that a **for** loop is executed is necessarily finite, which is not the case for **while** loops, within which a computation can get stuck in what is classically called in programming *an infinite loop*. We will see that the **while** loop

instruction is essential, and that some computable (and total) functions cannot be programmed using only `for` loops.

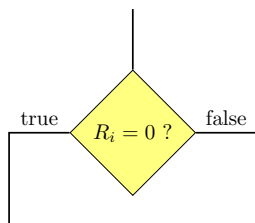
**Programming diagrams.** Structured programming finds its genesis in the work of Goldstine and von Neumann [74] who, from 1946, show a concern no longer to capture mathematically the notion of computation, but that of developing a programming system. They develop a way of presenting algorithms based on *programming diagrams*. We give here the simplified presentation that one finds in the reference work of Piergiorgio Odifreddi [168].

**Definition 3.3.** A programming diagram is obtained by connecting between them basic bricks of two types:

- Assignment instructions:



- A conditional instruction:



A programming diagram has one input and one or more outputs. The computation is carried out linearly by executing each block from the input to one of its outputs. ◇

By way of example, Figure 3.4 is a programming diagram corresponding to the addition function.

Programming diagrams can be programmed on register machines with an instruction set including conditional and unconditional jumps (the latter simply called “jumps”), that is, instructions of type *goto*, which allow to determine which is the next instruction of the program which will be executed. It is still today the mechanism at work in the different assembly languages of micro-processors.

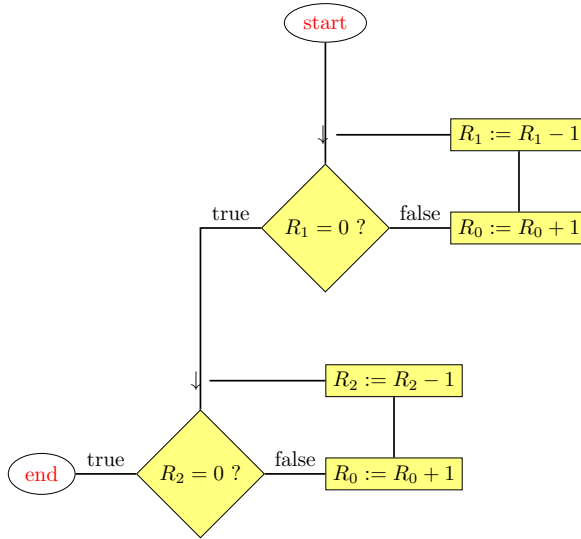


Figure 3.4: A programming diagram for the addition function of two integers. One supposes that the computation starts with  $R_0 = 0$ ,  $R_1 = n_1$ ,  $R_2 = n_2$ . At the end of the execution, one will have  $R_0$  equal to  $n_1 + n_2$ .

**Definition 3.5.** A *goto* program is a numbered sequence of instructions  $I_0, I_1, \dots, I_n$  where each instruction is of one of the following types:

1.   • Incrementation of a register: “ $R_i := R_i + 1$ ”.
- Decrementation of a register “ $R_i := R_i - 1$ ”.
- Assignment of a register: “ $R_i := 0$ ”.
2. The conditional jump instruction: “if  $R_i = 0$  goto  $n_1$  else goto  $n_2$ ”.  
*If  $R_i$  is equal to 0, instruction number  $n_1$  is the next to be executed, otherwise it is instruction number  $n_2$ .*
3. The unconditional jump: “goto  $m$ ”.  
*Instruction number  $m$  is the next to be executed.*

The execution of a goto program halts when there is no more next instruction to execute (which in particular can happen after an instruction of type “goto  $m$ ” in a program with less than  $m$  instructions.) ◇

**Remark**

Note that the unconditional jump “goto  $m$ ” can be replaced by a conditional jump “if  $R_i = 0$  goto  $n$  else goto  $n$ ”. A *goto* program can therefore be expressed without instruction 3.

It is clear that programming diagrams are interchangeable with goto type programs, and we will use one formalism or the other depending on the situation.

**Computable functions.** Notations  $\Phi_e(x_1, \dots, x_n) \downarrow = y$  and  $\Phi_e(x_1, \dots, x_n) \uparrow$  used throughout the book naturally apply to programs executed by register machines, once the conventions for passing parameters and retrieving the result have been fixed:

**Definition 3.6.** Given a structured or goto type program  $P$ , we write  $P(x_1, \dots, x_k)$  to denote the execution of  $P$  with the registers  $R_1, \dots, R_k$  initialized to  $x_1, \dots, x_k$ , and all the other registers initialized to 0. We write  $P(x_1, \dots, x_k) \downarrow = x$  to signify that such an execution halts with the value  $x$  in the  $R_0$  register, and  $P(x_1, \dots, x_k) \uparrow$  to signify that the execution does not halt.  $\diamond$

We can now use our computational model to give a precise mathematical definition of a computable function. Let us formalize beforehand the notations  $\text{dom } f$  and  $\text{Im } f$  which denote respectively the domain and the image of a function  $f$ .

**Notation**

Given a (possibly partial) function  $f : A \rightarrow B$ , we denote by  $\text{dom } f$  the domain of definition of  $f$ , and  $\text{Im } f = \{Y \in B : \exists X \in \text{dom } f \ f(X) = Y\}$  its image.

**Definition 3.7.** A (possibly partial) function  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  is *computable by structured program* (resp. *by goto program*) if there is a structured program (resp. a goto program)  $P_f$ , such that  $P_f(x_1, \dots, x_n) \downarrow = f(x_1, \dots, x_n)$  for all  $x_1, \dots, x_n \in \text{dom } f$  and such that  $P_f(x_1, \dots, x_n) \uparrow$  for all  $x_1, \dots, x_n \notin \text{dom } f$ .  $\diamond$

**Justification of the model of computation.** The programmer will perhaps not be convinced by the fact that the model of register machines with structured programs (or goto) indeed allows to program all the functions which he could write in his favorite language.

One aspect in particular may cause concern: A modern programming language allows the use of arrays, and even dynamic arrays, which can increase

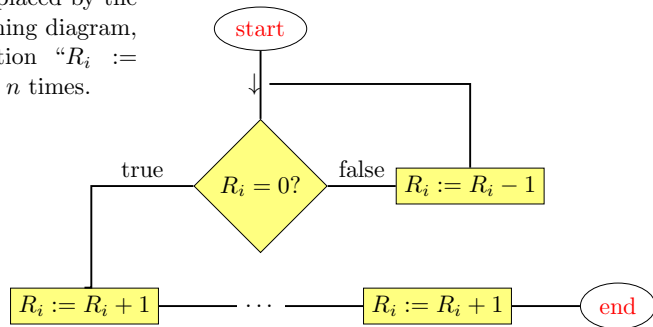
in size as needed. We will see for example in Definition 3.24 that the function known as Ackermann is computed naturally using a stack structure, and that it is not clear at all that we can do without it. Fortunately, we will show with Proposition 3.26 that it is perfectly possible to simulate the manipulations of dynamic arrays within register machines, by coding the latter in registers, which, let us remember, can contain arbitrarily large integers.

**Simulation of structured programs by goto programs.** We start by showing that goto programs, despite their simplicity, are sufficient to compute everything that structured programs can compute.

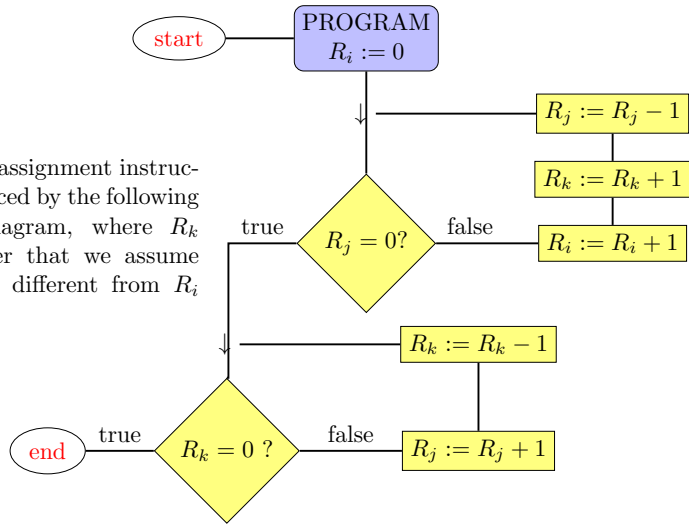
**Proposition 3.8.** Let  $n \in \mathbb{N}^*$  and  $f : \mathbb{N}^n \rightarrow \mathbb{N}$  be a function computable by a structured program. Then,  $f$  is computable by a goto program. ★

**PROOF.** It suffices to show that each instruction of a structured program can be replaced by a programming diagram.

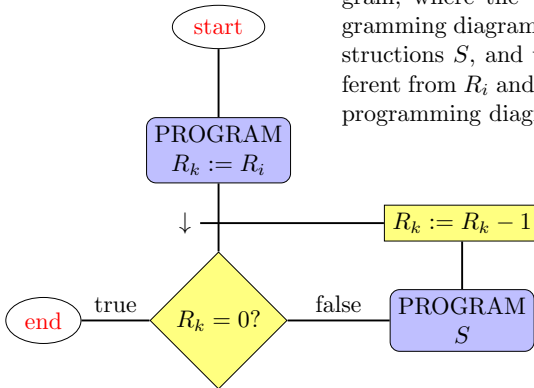
The “ $R_i := n$ ” assignment instruction can be replaced by the following programming diagram, where the instruction “ $R_i := R_i + 1$ ” is repeated  $n$  times.



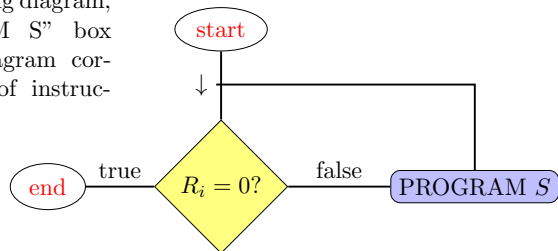
The “ $R_i := R_j$ ” assignment instruction can be replaced by the following programming diagram, where  $R_k$  is a new register that we assume to equal 0, and different from  $R_i$  and  $R_j$ .



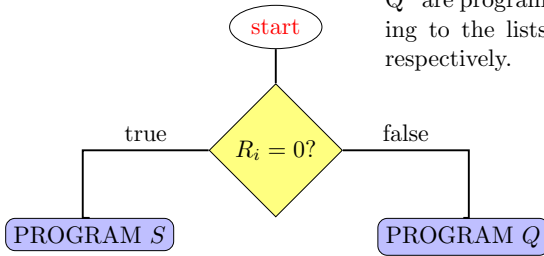
The “for  $i = 1$  to  $R_i$  do  $S$ ” loop instruction can be replaced by the following programming diagram, where the “PROGRAM  $S$ ” box is a programming diagram corresponding to the list of instructions  $S$ , and where  $R_k$  is a new register different from  $R_i$  and unused in the “PROGRAM  $S$ ” programming diagram.



The “while  $R_i \neq 0$  do  $S$ ” loop instruction can be replaced by the following programming diagram, where the “PROGRAM  $S$ ” box is the programming diagram corresponding to the list of instructions  $S$ .



The “if  $R_i = 0$  then  $S$  else  $Q$ ” conditional instruction can be replaced by the following programming diagram, where both boxes “PROGRAM  $S$ ” and “PROGRAM  $Q$ ” are programming diagrams corresponding to the lists of instructions  $S$  and  $Q$ , respectively.



■

### 3.2. Recursive functions are computable

Structured programs follow the concepts of so-called *imperative* programming: instructions modifying the state of the machine are executed one after the other. The general recursive functions follow the paradigm of so-called *functional* programming: a program is a composition of mathematical functions and a computation is the evaluation of these functions. An advantage of functional programming often highlighted is the absence of *side effects*: the result of a function depends on its parameters and only on its parameters (the state of the machine on which the function is executed has no effect). For example, you can connect the output of one function to the input of another without expecting any unpleasant surprises. By way of comparison, the combination of structured programs  $P_f, P_g$  computing functions  $f, g : \mathbb{N} \rightarrow \mathbb{N}$  in a program computing the function  $x \mapsto f(g(x))$  requires a little work, in order to avoid side effects.

**Definition 3.9.** A structured or goto type program is *clean* if it finishes its computation with all its registers — except  $R_0$  — in the same state as at the start of the computation. ◇

**Exercise 3.10.** Show that for any structured program  $M$ , there exists a clean structured program  $N$  such that  $M(\bar{x}) \Downarrow y \leftrightarrow N(\bar{x}) \Downarrow y$ . ◇

**Theorem 3.11 (Wang [228], Peter [174], Ershov [56])**

Any (possibly partial) general recursive function can be computed by a

*structured program. Any primitive recursive function can be computed by a structured program with no **while** loops.*

PROOF. We leave it to the reader to show that the constant functions, the projection functions and the successor function are all computable by a structured program. The following cases remain to be dealt with:

*Composition scheme.* Let

$$f(x_1, \dots, x_m) = g(h_1(x_1, \dots, x_m), \dots, h_k(x_1, \dots, x_m))$$

for functions  $g : \mathbb{N}^k \rightarrow \mathbb{N}$  and  $h_1, \dots, h_k : \mathbb{N}^m \rightarrow \mathbb{N}$  computable by structured programs  $G, H_1, \dots, H_m$ . By Exercize 3.10 we can assume that  $G, H_1, \dots, H_m$  are proper. Suppose each of these programs uses at most the registers  $R_0, \dots, R_z$  for  $z > m$ . The program  $F$  for computing  $f$  is given by the following sequence of instructions.

---

Program  $F$

---

$\langle$ Instructions of  $H_1$  $\rangle$

$R_{z+1} := R_0$

$R_0 := 0$

$\langle$ Instructions of  $H_2$  $\rangle$

$R_{z+2} := R_0$

$R_0 := 0$

$\dots$

$\langle$ Instructions of  $H_k$  $\rangle$

$R_{z+k} := R_0$

$R_0 := 0$

$R_1 := R_{z+1}$

$\dots$

$R_k := R_{z+k}$

$\langle$ Instructions of  $G$  $\rangle$

---

Note that if  $G, H_1, \dots, H_m$  does not use a **while** loop, then the program to compute  $F$  does not use one either.

*Primitive recursion scheme.* Let

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n) \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)) \end{aligned}$$

for  $g$  and  $h$  computable respectively by clean structured programs  $G$  and  $H$ , using at most the registers  $R_0, \dots, R_z$  for  $z > n + 2$ . The following program  $F$  allows to compute  $f$ :

---

 Program  $F$ 


---

 $R_{z+1} := R_{n+1}$ 
 $\langle \text{Instructions of } G \rangle$ 
 $R_{n+1} := 0$ 
 $R_{n+2} := R_0$ 
**for**  $i = 1$  **to**  $R_{z+1}$  **do**
 $R_0 := 0$ 
 $\langle \text{Instructions of } H \rangle$ 
 $R_{n+1} := R_{n+1} + 1$ 
 $R_{n+2} := R_0$ 
**end**


---

Note once again that if  $G, H$  does not use **while** loops, then the program to compute  $F$  does not use any either.

*Minimization scheme.* Let

$$f(x_1, \dots, x_n) = \min\{x \in \mathbb{N} : \forall i \leq x (g(x_1, \dots, x_n, i) \downarrow \wedge g(x_1, \dots, x_n, x) = 0)\}$$

for  $g$  computable by a clean program  $G$ , using at most the registers  $R_0, \dots, R_z$  for  $z > n + 1$ . The following program computes  $f$ :

---

 Program  $F$ 


---

 $R_{z+1} := 1$ 
 $R_{n+1} := 0$ 
**while**  $R_{z+1} \neq 0$  **do**
 $R_0 := 0$ 
 $\langle \text{Instructions of } G \rangle$ 
 $R_{z+1} := R_0$ 
 $R_{n+1} := R_{n+1} + 1$ 
**end**
 $R_{n+1} := R_{n+1} - 1$ 
 $R_0 := R_{n+1}$ 


---

This concludes the proof. ■

### 3.3. Study of primitive recursive functions

We now move on to the more difficult part of this chapter. In order to show that the functions computable by structured programs are general recursive functions, we need a certain number of tools, in particular on the primitive recursive functions. We alternate different propositions and

exercises allowing to see that it is a large class of functions. Let us recall the notations of Definition 1.1 for the basic primitive recursive functions:

**Notation**

We denote by  $p_i^n : \mathbb{N}^n \rightarrow \mathbb{N}$  the projection such that  $p_i^n(x_1, \dots, x_n) = x_i$ . We denote by  $c_i^n : \mathbb{N}^n \rightarrow \mathbb{N}$  the constant function such that  $c_i^n(x_1, \dots, x_n) = i$ . We denote by  $\text{succ} : \mathbb{N} \rightarrow \mathbb{N}$  the successor function, defined by  $\text{succ}(n) = n + 1$ .

**Exercise 3.12. (★)** Show that addition, multiplication and the exponential function are primitive recursive.  $\diamond$

**Exercise 3.13. (★)** Show that the predecessor and subtraction functions are primitive recursive (as our functions are valued in  $\mathbb{N}$ , we will use 0 instead of the result if it is negative).  $\diamond$

**Exercise 3.14. (★)** Show that the function  $\text{sg} : \mathbb{N} \rightarrow \mathbb{N}$  which associates 0 with 0 and which associates 1 with all other integers, is primitive recursive. Show that the function  $\overline{\text{sg}} : \mathbb{N} \rightarrow \mathbb{N}$  which associates 0 with 1 and which associates 0 with all other integers, is primitive recursive.  $\diamond$

**Definition 3.15.** A predicate  $P \subseteq \mathbb{N}^n$  is primitive recursive (resp. general recursive) if there exists a primitive recursive (resp. general recursive) function  $f : \mathbb{N}^n \rightarrow \{0, 1\}$  such that  $(x_1, \dots, x_n) \in P \leftrightarrow f(x_1, \dots, x_n) = 1$  for all  $x_1, \dots, x_n \in \mathbb{N}$ .  $\diamond$

**Example 3.16.** The comparison predicates  $\leq, <, \geq, >, =, \neq$  are primitive recursive via the following functions:

$$\begin{array}{llll} a \leq b & = & \text{sg}(\text{succ}(b) - a) & a > b & = & b < a \\ a < b & = & \text{sg}(b - a) & a = b & = & (a \leq b) \times (b \leq a) \\ a \geq b & = & b \leq a & a \neq b & = & \overline{\text{sg}}(a = b) \end{array}$$

Formally, the projections are used if necessary, for example to reverse the order of the parameters in the definition of  $\geq$  from that of  $\leq$ .

**Proposition 3.17.** The primitive recursive functions are closed under definition by case on a primitive recursive predicate: if  $g$  and  $h$  are two primitive recursive functions from  $\mathbb{N}^p$  to  $\mathbb{N}$ , and if  $P$  is a primitive recursive predicate on  $\mathbb{N}^p$ , then the function

$$\begin{aligned} f(x_1, \dots, x_p) &= g(x_1, \dots, x_p) && \text{if } P(x_1, \dots, x_p) \\ &= h(x_1, \dots, x_p) && \text{otherwise} \end{aligned}$$

is primitive recursive. ★

PROOF. Since  $P$  is a primitive recursive predicate, there exists a function  $d : \mathbb{N}^n \rightarrow \mathbb{N}$  such that

$$\begin{aligned} d(x_1, \dots, x_p) &= 1 && \text{if } P(x_1, \dots, x_p) \\ &= 0 && \text{otherwise.} \end{aligned}$$

We therefore define:

$$\begin{aligned} f(x_1, \dots, x_p) &= g(x_1, \dots, x_p) \times \text{sg}(d(x_1, \dots, x_p)) + \\ &\quad h(x_1, \dots, x_p) \times \overline{\text{sg}}(d(x_1, \dots, x_p)) \end{aligned} \quad \blacksquare$$

The proof of the following proposition provides an example of application of the definition by case.

**Proposition 3.18.** The integer division is primitive recursive. ★

PROOF. We use the primitive recursion scheme coupled with the case definition scheme. We define  $\text{div}(a, b) = \text{div}(a, b, a)$  where:

$$\begin{aligned} \text{div}(a, b, 0) &= 0 \\ \text{div}(a, b, n+1) &= \text{succ}(n) && \text{if } \text{succ}(n) \times b = a \\ &= \text{div}(a, b, n) && \text{otherwise.} \end{aligned} \quad \blacksquare$$

**Exercize 3.19. (★)** Show that the primitive recursive predicates are closed under conjunction, disjunction, negation, bounded existential quantification and bounded universal quantification. ◇

**Exercize 3.20. (★)** Show that the primitive recursive functions are closed under bounded minimization: if  $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$  is primitive recursive, then the function  $g : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$  which to  $x_1, \dots, x_p, n$  associates the smallest  $t \leq n$  such that  $f(x_1, \dots, x_p, t) = 0$  (and 0 if such a  $t \leq n$  does not exist), is primitive recursive. ◇

Let us remember Cantor's bijections as defined in Section 2-3. The encoding of the pairs  $\langle x_1, x_2 \rangle$  is a notation for the application of the bijection  $\alpha_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$  defined by

$$\alpha_2(x, y) = y + \frac{(x + y + 1)(x + y)}{2}$$

The  $\langle x_1, \dots, x_k \rangle$  encoding of the  $n$ -tuples is a notation for the application of the bijections  $\alpha_k : \mathbb{N}^k \rightarrow \mathbb{N}$  defined inductively by  $\alpha_{k+1}(x_1, x_2, \dots, x_{k+1}) = \alpha_2(x_1, \alpha_k(x_2, \dots, x_{k+1}))$ .

**Proposition 3.21.** For all  $n$  the bijection

$$x_1, \dots, x_n \mapsto \langle x_1, \dots, x_n \rangle$$

is primitive recursive. For all  $n$  and all  $i \leq n$  the function

$$\langle x_1, \dots, x_n \rangle \mapsto x_i$$

is primitive recursive. ★

PROOF. Addition, multiplication and integer division being primitive recursive, the function  $(x, y) \mapsto y + \frac{(x+y+1)(x+y)}{2}$  is also primitive recursive by the composition scheme. It is therefore the same for all  $n$  for the functions  $x_1, \dots, x_n \mapsto \langle x_1, \dots, x_n \rangle$  which are defined inductively by composition.

The function which to  $\langle x_1, x_2 \rangle$  associates  $x_1$  is primitive recursive using bounded minimization and the closure of primitive recursive predicates by bounded existential quantification:

$$f(n) = \min\{x \leq n : \exists y \leq n \langle x, y \rangle = n\}.$$

We leave it to the reader to embroider on this idea to show that all the projections are thus primitive recursive. ■

We have at this stage all the elements necessary to show an important first theorem. We saw with Theorem 3.11 that primitive recursive functions can be programmed by structured programs with no **while** loops. The converse is true:

**Theorem 3.22**

*Any function computable by a structured program without **while** loops is primitive recursive.*

PROOF. For  $k \in \mathbb{N}$  given, and for a structured program  $P$  without **while** loops and using at most the registers  $R_0, \dots, R_k$ , we define the function  $f_P : \mathbb{N} \rightarrow \mathbb{N}$  by  $f_P(\langle x_0, \dots, x_k \rangle) = \langle v_0, \dots, v_k \rangle$  where  $v_i$  is the value of the register  $R_i$  at the end of the execution of the program  $P$ , when its execution begins with its registers initialized to the values  $x_0, \dots, x_k$ .

Let us show that for any structured program  $P$  without **while** loops, the corresponding function  $f_P$  is primitive recursive. It is clear that this is the case for the empty program. Let  $Q$  be the program whose only instruction is  $R_i := R_i + 1$ . Then, the function  $f_Q$  is given by  $f_Q(\langle x_0, \dots, x_k \rangle) = \langle x_0, \dots, x_i + 1, \dots, x_k \rangle$ . We leave it to the reader to find the primitive recursive function corresponding to the instructions  $R_i := R_i - 1$ ,  $R_i := c$  for  $c \in \mathbb{N}$  and  $R_i := R_j$ .

Suppose now that the proposition is true for programs  $P, P'$ , via functions  $f_P, f_{P'}$  and let  $Q$  be the program “if  $R_j = 0$  then  $P$  else  $P'$ ”. The function  $f_Q$  is therefore given by

$$\begin{aligned} f_Q(\langle x_0, \dots, x_k \rangle) &= f_P(\langle x_0, \dots, x_k \rangle) && \text{if } x_j = 0 \\ &= f_{P'}(\langle x_0, \dots, x_k \rangle) && \text{otherwise.} \end{aligned}$$

Suppose now that the proposition is true for programs  $P$ , via functions  $f_P$  and let  $Q$  of the following form: “for  $i = 1$  to  $R_j$  do  $P$ ”. The function  $f_Q$  is given by:

$$f_Q(\langle x_0, \dots, x_k \rangle) = g(\langle x_0, \dots, x_k \rangle, x_j)$$

where

$$\begin{aligned} g(\langle x_0, \dots, x_k \rangle, 0) &= \langle x_0, \dots, x_k \rangle \\ g(\langle x_0, \dots, x_k \rangle, z + 1) &= f_P(g(\langle x_0, \dots, x_k \rangle, z)). \end{aligned}$$

Suppose now the proposition true for programs  $P, P'$ , via functions  $f_P, f_{P'}$  and let  $Q$  consist of the instructions of  $P$  followed by those of  $P'$ . Then,  $f_Q = f_{P'}(f_P(\langle x_0, \dots, x_k \rangle))$ .

Using each of the cases described, we show by induction that the state of the registers of any structured program without **while** loops is a primitive recursive function. All we have to do is retrieve the value of the  $R_0$  register. ■

### 3.4. Study of the Ackermann function

There are several ways of seeing that not all computable functions are primitive recursive. The following is the most natural for the expert in Computability Theory for whom effective diagonalizations no longer hold any secrets:

**Exercize 3.23. (★)** Show that there exists a computable set  $A \subseteq \mathbb{N}$  such that  $n \mapsto \Phi_e(n)$  is a primitive recursive function for all  $e \in A$  and such that every primitive recursive function has a code in  $A$ . Deduce that there exists a total computable function which is not primitive recursive. ◇

A commonly given example of a computable non-primitive recursive function is Ackermann's function:

**Definition 3.24 (Fonction d'Ackermann [3]).** Define the functions  $A_n : \mathbb{N} \rightarrow \mathbb{N}$  by induction on  $n \in \mathbb{N}$  as follows.

- $A_0$  is the function  $x \mapsto 2^x$ ;

- $A_{n+1}(x)$  is the application  $x$  times of the function  $A_n$  on 1:

$$A_n(A_n(\dots(A_n(1)))).$$

Formally:

$$\begin{aligned} A_0(x) &= 2^x \\ A_{n+1}(0) &= 1 \\ A_{n+1}(x) &= A_n(A_{n+1}(x-1)). \end{aligned}$$

Ackermann's function is the  $n \mapsto A_n(n)$  function. ◇

The Ackermann function is growing extremely fast:  $A(0) = 1$ ,  $A(1) = 2$ ,  $A(2) = 16$ , and  $A(3)$  is already equal to 65,536 iterations of the function  $x \mapsto 2^x$  (starting at 0), that is:

$$A(3) = 2^{\left(2^{\left(\dots^{2^0}\right)}\right)} \text{ where the power is iterated 65,536 times.}$$

Despite its very strong growth, the Ackermann function is computable: to compute  $A_n(n)$ , we can use a stack containing either functions  $A_n$  (in practice a representation of these functions), or integers. For example, if we stack  $A_n$ ,  $A_{n+1}$  and then  $k$ , this corresponds to the computation  $A_n(A_{n+1}(k))$ . So the top of the stack is always an integer, and the element that follows (if it exists) is always a function. Also to compute  $A_n(n)$  we proceed as follows:

1. We stack  $A_n$ , then we stack  $n$ .
2. As long as the stack contains more than one element:
  - (a) We unstack the integer  $k$ , then we unstack the function  $A_m$ .
  - (b) If  $m = 0$  we stack  $2^k$ .
  - (c) Otherwise, if  $k = 0$  we stack 1.
  - (d) Otherwise we stack  $A_{m-1}$ , then  $A_m$  and finally  $k - 1$ .

The algorithm will halt when the stack contains only one element, the result of the computation of  $A_n(n)$ .

We leave in exercise the proof that Ackermann's function grows faster than any primitive recursive function, and is therefore not itself primitive recursive.

**Exercise 3.25** (Cori and Lascar[41]). (★)

- (1) Show that  $A_n(x) > x$  for all  $n, x$ .
- (2) Show that the function  $A_n$  is strictly increasing for all  $n$ .

- (3) Show that the function  $n \mapsto A_n(x)$  is increasing for all  $x$ .
- (4) Show that  $A_{n+1}(x+y) \geq A_n^{(y)}(x)$  for all  $n, x, y$ , where  $f^{(m)}(x)$  denotes  $f(f(\dots f(x)\dots))$  where the function  $f$  is iterated  $m$  times.
- (5) Show that for all  $n$ , we have  $A_{n+2}(x) > A_n^{(x+1)}(x+1)$  for almost all  $x$ .
- (6) Let  $n > 0$  and  $f : \mathbb{N}^n \rightarrow \mathbb{N}$ . We denote by  $P(f)$  the predicate

$$\exists k \forall^\infty x_1, \dots, x_n A_k(\max(x_1, \dots, x_n)) > f(x_1, \dots, x_n).$$

Show that  $P(f)$  is true for any primitive recursive function  $f$ .

Deduce that Ackermann's function is not primitive recursive.  $\diamond$

Loops of type **while** are therefore essential to compute certain functions, and with them comes the possibility of writing programs that do not halt.

### 3.5. Computable functions are recursive

We are now going to show that the recursive function minimization scheme allows us to compute any programmable function with or without **while** loops. To do this we start by seeing how to simulate list structures of arbitrary size by integers. This indeed seems at the moment to be a lack of our register machine model and structured programs. For example, we need a stack to compute the Ackermann function via the algorithm mentioned above.

**Proposition 3.26.** There exists a bijection  $[] : \bigcup_{n \in \mathbb{N}} \mathbb{N}^n \rightarrow \mathbb{N}$  (where we denote by  $[x_1, \dots, x_n]$  the integer corresponding to  $n$ -tuple  $(x_1, \dots, x_n)$ ), such that the following operations are primitive recursive:

- 1. The function  $::$  to add to the top of the list defined by  $a :: [x_1, \dots, x_n] = [a, x_1, \dots, x_n]$ .
- 2. The head and tail functions **hd** and **tl**, defined by

$$\begin{array}{llll} \text{hd}([]) & = & 0 & \text{tl}([]) & = & 0 \\ \text{hd}(x :: l) & = & x & \text{tl}(x :: l) & = & l. \end{array}$$

- 3. The  $||$  size function of a list defined by  $|[x_1, \dots, x_n]| = n$ .
- 4. The **get** and **set** functions such that

$$\begin{array}{ll} \text{get}([x_0, \dots, x_{n-1}], i) & = x_i \text{ if } i < n \\ \text{get}([x_0, \dots, x_{n-1}], i) & = 0 \text{ otherwise} \end{array}$$

and

$$\begin{aligned} \text{set}([x_0, \dots, x_{n-1}], a, i) &= [x_0, \dots, x_{i-1}, a, x_{i+1}, \dots, x_{n-1}] & \text{if } i < n \\ \text{set}([x_0, \dots, x_{n-1}], a, i) &= [x_0, \dots, x_{n-1}] & \text{otherwise} \end{aligned}$$

★

PROOF. The bijection  $[] : \bigcup_{n \in \mathbb{N}} \mathbb{N}^n$  is defined by induction starting with the empty list  $[] = 0$  and applying the operation of adding to the head of the list defined by  $a :: l = 1 + \alpha_2(a, l)$  where  $\alpha_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$  is the bijection of Exercize 2-3.7. It is clear that the addition function at the head is primitive recursive, as are the functions  $\text{hd}$  and  $\text{tl}$  obtained thanks to the inverse functions of  $(x_1, x_2) \mapsto \langle x_1, x_2 \rangle$ . Let us show that the coding of the lists thus obtained is indeed a bijection.

Let us show by induction that two lists of different sizes cannot be encoded by the same element. The code for an empty list is 0 and the code for a non-empty list is of the form  $1 + \alpha_2(a, l) \neq 0$ . So the code for an empty list is always different from the code for a non-empty list.

Now suppose that all the lists of size  $n$  have a different code than the lists of size  $m > n$ . Let us show that all lists of size  $n + 1$  have a different code from that of lists of size  $m > n + 1$ . The codes of lists of size  $n + 1$  are of the form  $1 + \alpha_2(a, l_1)$  for  $l_1$  the code of a list of size  $n$ . The codes of lists of size  $m > n + 1$  are of the form  $1 + \alpha_2(b, l_2)$  for  $l_2$  the code of a list of size  $m > n$ . By induction hypothesis, we necessarily have  $l_1 \neq l_2$  and since  $\alpha_2$  is injective, we necessarily have  $1 + \alpha_2(a, l_1) \neq 1 + \alpha_2(b, l_2)$ . So the codes of lists of size  $n + 1$  are different from the codes of lists of size  $m > n + 1$ . By induction, we deduce that the codes of lists of different sizes are different.

Let us now show by induction on  $k$  that if  $(a_1, \dots, a_k) \neq (b_1, \dots, b_k)$ , then we have  $[a_1, \dots, a_k] \neq [b_1, \dots, b_k]$ . For  $k = 1$  we have that  $a_1 \neq b_1$  implies  $1 + \alpha_2(a_1, 0) \neq 1 + \alpha_2(b_1, 0)$  because  $\alpha_2$  is injective. We therefore have  $[a_1] \neq [b_1]$ . Suppose this is the case for  $k$  and show that this is the case for  $k + 1$ . Suppose  $(a_1, \dots, a_{k+1}) \neq (b_1, \dots, b_{k+1})$ . If  $a_1 \neq b_1$ , then we have  $1 + \alpha_2(a_1, [a_2, \dots, a_{k+1}]) \neq 1 + \alpha_2(b_1, [b_2, \dots, b_{k+1}])$  because  $\alpha_2$  is injective. If  $(a_2, \dots, a_{k+1}) \neq (b_2, \dots, b_{k+1})$ , then by induction hypothesis we have  $[a_2, \dots, a_{k+1}] \neq [b_2, \dots, b_{k+1}]$  and therefore  $1 + \alpha_2(a_1, [a_2, \dots, a_{k+1}]) \neq 1 + \alpha_2(b_1, [b_2, \dots, b_{k+1}])$  because  $\alpha_2$  is injective. By induction, for all  $k$  we have therefore  $(a_1, \dots, a_k) \neq (b_1, \dots, b_k)$  implies  $[a_1, \dots, a_k] \neq [b_1, \dots, b_k]$ . The function  $[]$  is therefore injective.

Let us now show that  $[]$  is surjective. Let us assume absurdly that this is not the case. In this case there exists a smallest  $n$  such that  $n$  is not the code of any list. Note that we necessarily have  $n > 0$  because 0 is the code of the empty list. Also as  $\alpha_2$  is surjective, there exists  $(a, b)$  such that  $\alpha_2(a, b) = n - 1$  and therefore such that  $1 + \alpha_2(a, b) = n$ . We also

necessarily have  $b \leq n - 1 < n$ . Also by minimality of  $n$ , there must exist a list  $l$  of which  $b$  is the code. So  $n$  is the code of the list  $a :: l$ , which contradicts our hypothesis on  $n$ .

In order to show that the functions  $||$ ,  $\text{get}$  and  $\text{set}$  are primitive recursive, we give a primitive recursive definition of the function  $\text{tl}(l, n)$  which cuts off a list  $l$  of its  $n$  first elements:

$$\begin{aligned}\text{tl}(l, 0) &= l \\ \text{tl}(l, n + 1) &= \text{tl}(\text{tl}(l, n)).\end{aligned}$$

The size is then defined via the minimization scheme bounded by  $|l| = \min\{n \leq l : \text{tl}(l, n) = []\}$ . The  $\text{get}$  function has the following primitive recursive definition:  $\text{get}(l, n) = \text{hd}(\text{tl}(l, n))$ . The  $\text{set}$  function is defined in two steps, by first adding an additional parameter:

$$\begin{aligned}\text{set}(l, a, i) &= \text{set}(l, a, i, i) \\ \text{set}(l, a, n, 0) &= a :: \text{tl}(l, \text{succ}(n)) \\ \text{set}(l, a, n, i + 1) &= \text{get}(l, n - \text{succ}(i)) :: \text{set}(l, a, n, i)\end{aligned}$$

■

We now have all the elements necessary to show that functions computable by structured programs are recursive.

**Theorem 3.27**

*Any function that can be computed by a goto program (and therefore also by a structured program) is a general recursive function.*

The rest of the section is devoted to the proof, for which we need to fix an encoding of goto programs and register machines.

**Coding of goto programs.** We code the instructions of goto programs as follows:

- “ $R_i = R_i + 1$ ” is encoded by  $\langle 0, i \rangle$
- “ $R_i = R_i - 1$ ” is encoded by  $\langle 1, i \rangle$
- “ $R_i = 0$ ” is encoded by  $\langle 2, i \rangle$
- “if  $R_i = 0$  goto  $n_1$  else  $n_2$ ” is encoded by  $\langle 3, \langle i, \langle n_1, n_2 \rangle \rangle \rangle$
- “goto  $n$ ” is encoded by  $\langle 4, n \rangle$

For uniformity reasons, it will be useful to have a bound on the maximum index of the registers used. The code of a goto program is simply given by the code:  $\langle k, [I_1, \dots, I_n] \rangle$  where  $k$  is such that the program uses at most the registers  $R_0, \dots, R_k$ , and where  $I_e$  is the code of the  $e$ -th instruction for  $1 \leq e \leq n$ .

**Coding of register machines.** We now fix an encoding of the state of a  $k$  register machine. This state is given by the value of the registers as well as by the index of the next instruction to be executed. For a given number of registers  $k$ , this code is  $\langle m, [R_0, \dots, R_k] \rangle$  where  $m$  is the instruction number and  $R_i$  is the value of register number  $i$  for  $0 \leq i \leq k$ .

**Initialization function.** Let us now fix an initialization function  $\text{init}$ , which given a code  $e = \langle k, I \rangle$  of a program (where  $I$  is a list of instructions), and given values  $x_1, \dots, x_n$  (for  $n \leq k$ ), returns the code representing the state of the  $k$  register machine, at the start of the computation.

$$\text{init}(\langle k, I \rangle, x_1, \dots, x_n) = \langle 0, 0 :: x_1 :: \dots :: x_n :: \text{aux}(k - n) \rangle$$

with  $\text{aux}$  defined by

$$\begin{aligned} \text{aux}(0) &= [] \\ \text{aux}(k + 1) &= 0 :: \text{aux}(k). \end{aligned}$$

It is clear that the function  $\text{init}$  is primitive recursive.

**Transition functions 1.** We define a transition function  $\text{tr}_1$ , which given the code  $e = \langle k, I \rangle$  of a program and the code  $p = \langle m, R \rangle$  of the state of a machine, returns the number of the next instruction to be executed at the next computation step. We will use for that a function  $\text{cur} : \mathbb{N} \rightarrow \mathbb{N}$  which, given the code  $e = \langle k, I \rangle$  of a program and the code  $p = \langle m, R \rangle$  of the state of a machine, allows to obtain the current instruction of the machine:

$$\text{cur}(\langle k, I \rangle, \langle m, R \rangle) = \text{get}(I, m).$$

Using the function  $\text{cur}$ , the functions  $\pi_1, \pi_2$  such that  $n = \langle \pi_1(n), \pi_2(n) \rangle$  we define  $\text{tr}_1(e, p)$  as being:

$$\begin{aligned} \pi_1(p) + 1 & \quad \text{if } \pi_1(\text{cur}(e, p)) \leq 2 \\ \pi_1(\pi_2(\pi_2(\text{cur}(e, p)))) & \quad \text{if } \pi_1(\text{cur}(e, p)) = 3 \text{ and} \\ & \quad \text{get}(\pi_2(p), \pi_1(\pi_2(\text{cur}(e, p)))) = 0 \\ \pi_2(\pi_2(\pi_2(\text{cur}(e, p)))) & \quad \text{if } \pi_1(\text{cur}(e, p)) = 3 \text{ and} \\ & \quad \text{get}(\pi_2(p), \pi_1(\pi_2(\text{cur}(e, p)))) \neq 0 \\ \pi_2(\text{cur}(e, p)) & \quad \text{if } \pi_1(\text{cur}(e, p)) = 4. \end{aligned}$$

It is clear that  $\text{tr}_1$  is primitive recursive.

**Transition functions 2.** We now define a transition function  $\text{tr}_2$  which, given the code  $e$  of a program and the code  $p$  of the state of a machine, allows to obtain the state of the registers from the machine to the next computation step.

For that, we will use two primitive recursive functions  $\text{inc} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  and  $\text{dec} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , such that

$$\begin{aligned}\text{inc}([x_0, \dots, x_n], i) &= [x_0, \dots, x_i + 1, \dots, x_n] \\ \text{dec}([x_0, \dots, x_n], i) &= [x_0, \dots, \max(0, x_i - 1), \dots, x_n]\end{aligned}$$

defined as follows.

$$\begin{aligned}\text{inc}(l, i) &= \text{set}(l, \text{succ}(\text{get}(l, i)), i) \\ \text{dec}(l, i) &= \text{set}(l, \text{pred}(\text{get}(l, i)), i).\end{aligned}$$

We can now define  $\text{tr}_2(e, p)$  as being:

$$\begin{aligned}\text{inc}(\pi_2(p), \pi_2(\text{cur}(e, p))) &\quad \text{if} \quad \pi_1(\text{cur}(e, p)) = 0 \\ \text{dec}(\pi_2(p), \pi_2(\text{cur}(e, p))) &\quad \text{if} \quad \pi_1(\text{cur}(e, p)) = 1 \\ \text{set}(\pi_2(p), 0, \text{cur}(e, p)) &\quad \text{if} \quad \pi_1(\text{cur}(e, p)) = 2 \\ \pi_2(p) &\quad \text{otherwise.}\end{aligned}$$

It is clear that  $\text{tr}_2$  is primitive recursive.

**End of the proof.** We now define the function  $\text{tr} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  of transition from one state to another:

$$\text{tr}(e, p) = \langle \text{tr}_1(e, p), \text{tr}_2(e, p) \rangle.$$

Then we define the primitive recursive function  $\text{st} : \mathbb{N} \times \mathbb{N}^n \times \mathbb{N} \rightarrow \mathbb{N}$  such that  $\text{st}(e, x_1, \dots, x_n, t)$  returns the state of the machine which executes the program  $e$ , with the registers  $R_1, \dots, R_n$ , initialized respectively to  $x_1, \dots, x_n$ , after  $t$  computation steps.

$$\begin{aligned}\text{st}(e, x_1, \dots, x_n, 0) &= \text{init}(e, x_1, \dots, x_n) \\ \text{st}(e, x_1, \dots, x_n, t + 1) &= \text{tr}(e, \text{st}(e, x_1, \dots, x_n, t)).\end{aligned}$$

We finally arrive at the stage for which we need a minimization scheme, leaving the possibility for a function not to be defined on certain inputs. The recursive function  $\text{time} : \mathbb{N} \times \mathbb{N}^n \rightarrow \mathbb{N}$  gives the smallest computation time necessary for the machine to halt, i.e., arrives at an instruction number greater than the number of instructions in the program. The function will be defined if and only if the machine halts for the program and the corresponding inputs.

$$\text{time}(e, x_1, \dots, x_n) = \min\{t \in \mathbb{N} : \pi_1(\text{st}(e, x_1, \dots, x_n, t)) \geq |\pi_2(e)|\}.$$

Finally, here is the recursive function which corresponds to the computation of the code machine  $e$ . We launch the transition function for the number

of steps necessary before the machine halts, and we return the value of the register  $R_0$ :

$$f(x_1, \dots, x_n) = \text{hd}(\pi_2(\text{st}(e, x_1, \dots, x_n, \text{time}(e, x_1, \dots, x_n))))).$$

This concludes the demonstration.

### 3.6. Consequences

According to the previous proof, given our encoding of a program by an integer  $e$ , its execution for  $t$  computation steps is a primitive recursive function and therefore itself computable by a structured program. The search for this smaller computation time can be done using a **while** loop. Note further that the proof is uniform: the same primitive recursive function adapts according to any code  $e$  of a program.

This makes it possible to obtain Theorem 3-3.1 of the existence of a universal program, used throughout the book, via the notation  $\Phi_e$  for the code program  $e$ .

#### **Theorem (3-3.1)**

Let  $n \in \mathbb{N}^*$ . There exists a code  $e$  of a computer program for which  $\Phi_e : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$  is such that for all  $x_1, \dots, x_n$  we have

- $\Phi_e(a, x_1, \dots, x_n) \uparrow$  iff  $\Phi_a(x_1, \dots, x_n) \uparrow$
- $\Phi_e(a, x_1, \dots, x_n) \downarrow = y$  iff  $\Phi_a(x_1, \dots, x_n) \downarrow = y$

The code  $e$  of the above theorem is a code of the function

$$(x_1, \dots, x_n) \mapsto \text{hd}(\pi_2(\text{st}(a, x_1, \dots, x_n, \text{time}(a, x_1, \dots, x_n))))$$

given at the end of the previous section. The function  $(a, x_1, \dots, x_n) \mapsto \text{time}(a, x_1, \dots, x_n)$  which looks for the smallest computation time such that the program halts is the only one which uses the minimization scheme. Note that this also makes it possible to give a precise mathematical definition to the notations  $\Phi_a(x_1, \dots, x_n)[t] \downarrow$  and  $\Phi_a(x_1, \dots, x_n)[t] \uparrow$ : they correspond respectively to the primitive recursive predicates:

$$\begin{aligned} \exists s \leq t \ \pi_1(\text{st}(e, x_1, \dots, x_n, s)) &\geq |\pi_2(e)| \\ \text{et} \quad \forall s \leq t \ \pi_1(\text{st}(e, x_1, \dots, x_n, s)) &< |\pi_2(e)| \end{aligned}$$



# Chapter 7

## Immunity and function growth

Computability Theory studies the computational power of sets of integers, modulo the Turing reduction. In this chapter, we will study in particular two large families of computational properties, namely, the ability to compute sets that are difficult to describe (immune, hyperimmune, effectively immune set), and the ability to compute fast-growing functions (hyperimmune function, dominating function). Let us take a few examples.

**Example 1.** According to Exercise 3-7.10, any infinite c.e. set contains a computable infinite subset. What computational power is needed to obtain an infinite set that does not have any computable infinite subset? We will study this in Section 1 under the concept of immune set.

**Example 2.** According to Kleene's fixed point theorem 3-6.2, for any total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , there exists a program code  $e$  such that  $\Phi_{f(e)} = \Phi_e$ . What is the computational power of a function with no fixed point? This notion will be studied in Section 2.

**Example 3.** Every computable function is trivially dominated by a computable function. How much computational power does it take to compute a function that is not dominated by any computable function? This will be the subject of Section 4.

It is difficult to form an intuition on the *a priori* computational power of properties formulated in such a diverse way, and in particular to compare them. The properties taken from the three preceding examples however

admit characterizations which will make this comparison easier. In general, the existence of numerous characterizations of the same computational power with very diverse formulations is a guarantee of the robustness of the concept. This is particularly the case for the properties studied in this chapter.

## 1. Immune sets

The first family of computational properties on sets relates to the ability to approximate the elements of a set. A set is computable if it is possible to compute effectively which elements belong to it or not. At the next level, a set is computable enumerable if there is a computable procedure for listing all of the elements that belong to it, but potentially out of order, so it is usually not possible to be certain that an element does not belong to the set. We are now going to study the computational power of infinite sets for which it is not even possible to enumerate in a computable way an infinite quantity of its elements.

**Definition 1.1.** An infinite set  $A \subseteq \mathbb{N}$  is *immune* if it does not contain any infinite c.e. subset. ◇

As we have seen, every infinite c.e. set contains a computable infinite subset. Thus, in an equivalent manner, an infinite set is immune if and only if it does not contain a computable infinite subset. In particular, any immune set  $A$  is necessarily non-computable, because  $A$  would then be its own infinite computable subset contradicting its immunity.

Immunity is a concept of set, but not of degree. Indeed, if  $A$  is an immune set, the set  $A \oplus \mathbb{N} = \{2n : n \in A\} \sqcup \{2n+1 : n \in \mathbb{N}\}$  has the same Turing degree as  $A$ , but  $A \oplus \mathbb{N}$  has the infinite computable subset  $\{2n+1 : n \in \mathbb{N}\}$ . Conversely, any non-computable Turing degree contains an immune set, as the following proposition shows.

**Proposition 1.2.** Any non-computable set is Turing equivalent to an immune set. ★

**PROOF.** Let  $A$  be a non-computable set. Let  $B = \{\sigma \in 2^{<\mathbb{N}} : \sigma \prec A\}$  be the set of initial segments of  $A$ . In particular,  $A \equiv_T B$ . It is also obvious that any infinite subset of  $B$  allows to compute arbitrarily large initial segments of  $A$ , and therefore  $A$  (as well as  $B$ ). Since  $A$  is not computable,  $B$  does not have any computable infinite subset. ■

The notion of immunity has two orthogonal reinforcements, namely effective immunity and hyperimmunity. These two notions are fundamental

computational properties in Computability Theory, and we will see for each of them several equivalent definitions. Recall that  $W_e$  denotes the c.e. set of code  $e$ :  $\{n : \Phi_e(n) \downarrow\}$ .

**Definition 1.3.** An infinite set  $A \subseteq \mathbb{N}$  is *effectively immune* if there exists a total computable function  $h : \mathbb{N} \rightarrow \mathbb{N}$  such that for any code  $e$ , if  $|W_e| \geq h(e)$  then  $W_e \not\subseteq A$ .  $\diamond$

Intuitively, an infinite set  $A$  is effectively immune if not only do infinite sets end up erring and enumerating an element outside  $A$ , but even more so, this error must occur after sufficiently few elements enumerated, depending on the enumeration code. In particular, any group that is effectively immune is immune.

We will see that the concept of effective immunity is particularly worthy of interest from the point of view of Turing degrees. The computational power corresponding to the capacity to compute an effectively immune set has many characterizations which will be studied in Section 2.

Recall that the *canonical code* of a finite set  $F$  is the integer  $n = \sum_{i \in F} 2^i$ . Let  $D_0, D_1, \dots$  be the collection of finite sets such that  $D_n$  is canonically encoded by  $n$  for all  $n \in \mathbb{N}$ .

**Definition 1.4.** An *array* is a collection of mutually disjoint finite sets  $F_0, F_1, \dots$ . An array  $F_0, F_1, \dots$  is *c.e.* if there exists a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $n$ ,  $F_n = D_{f(n)}$ . An infinite set  $A$  is *hyperimmune* if for any c.e. array  $F_0, F_1, \dots$ , there exists an integer  $n$  such that  $F_n \cap A = \emptyset$ .  $\diamond$

This more complex definition formalizes the idea according to which not only is it not possible to computably list an infinity of elements of  $A$ , but even more so it is not even possible to approximate infinitely many elements by block, that is, to list an infinity of pairwise disjoint finite “blocks”, such that each block contains at least one element of  $A$ .

As with the concept of effective immunity, it is the extension of hyperimmunity to Turing degrees that will be of particular interest to us in what follows. The computational power corresponding to the capacity to compute a hyperimmune set has many characterizations which will be studied in Section 4.

**Exercise 1.5.** Show that any hyperimmune set is immune.  $\diamond$

## 2. DNC functions

We now see an example of a remarkable Turing degree, the study of which undoubtedly dates back to the work of Arslanov [8], who studied Turing

degrees allowing to escape the famous Kleene fixed point theorem: given a computable function  $f$ , there exists  $e$  such that  $\Phi_e = \Phi_{f(e)}$ . What power is needed to compute a function  $f$  for which there is no such  $e$ ? This work was extended by Jockusch, Lerman, Soare, and Solovay who found an equivalent characterization, which today constitutes the modern definition of DNC degree.

**Definition 2.1.** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is *diagonally non-computable* (DNC) if  $f(n) \neq \Phi_n(n)$  for all  $n$ .  $\diamond$

Let us insist on the fact that the function  $f$  must be total in the definition above. Note that if  $\Phi_n(n) \uparrow$ , there is no restriction on the value of  $f(n)$ . A Turing degree is DNC if it contains a DNC function.

**Exercise 2.2. ( $\star$ )** Show that the DNC degrees are upward-closed, that is to say that if a set computes a DNC function, its degree is itself DNC.  $\diamond$

The notion of DNC degree has many applications in the links between Computability Theory and Algorithmic Randomness, as well as in Reverse Mathematics. We will see in particular with Corollary 18-4.3 that the DNC functions are numerous from the point of view of Measure Theory, but according to Proposition 10-3.36 rare from the point of view of Baire categories (we will see in particular the existence of non-DNC degrees which are not computable).

Let us observe first of all that there is no computable DNC function, by a simple diagonal argument, which is at the origin of the name “diagonally non-computable”. On the other hand, the following proposition shows that we can compute a DNC function using the halting problem.

**Proposition 2.3.**  $\mathbf{0}'$  is of DNC degree.  $\star$

PROOF. Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the  $\emptyset'$ -computable function, which for an input  $e$ , returns  $1 - \Phi_e(e)$  if  $e \in \emptyset'$  and 0 otherwise. This function is DNC, because it is total, and when  $\Phi_e(e) \downarrow$ , it returns a different value. Since  $\emptyset'$  computes a DNC function, and the DNC degrees are upward-closed,  $\mathbf{0}'$  is DNC.  $\blacksquare$

We will see in Section 3 that  $\mathbf{0}'$  is the only degree which is both DNC and c.e. It is clear that the DNC degrees are in uncountable quantity, because it is also the case of the degrees above  $\mathbf{0}'$ . We will see with Proposition 10-3.36 that the non-DNC degrees are also uncountable.

For the moment, we endeavor to show that the notion of DNC degree is natural, in the sense that it has many characterizations via very different formulations (and we will see others still in Part II on Algorithmic Randomness).

**Definition 2.4.** A function  $f$  is *fixed-point free* if  $\Phi_n \neq \Phi_{f(n)}$  for all  $n$ .  $\diamond$

**Theorem 2.5 (Jockusch, Lerman, Soare, and Solovay [100])**

Let  $X \in 2^{\mathbb{N}}$ . Then, the following statements are equivalent:

- (1)  $X$  computes a diagonally non-computable function.
- (2)  $X$  computes a free fixed-point function.

PROOF. Let us show (1)  $\Rightarrow$  (2). Let  $g \leq_T X$  be such that  $g(n) \neq \Phi_n(n)$  for all  $n$ . Then also,  $f \leq_T X$  such that  $f(n)$  is a code for a computable function defined only on the input  $n$ , and which to  $n$  associates  $g(n)$ . We have in particular  $\Phi_{f(n)}(n) \downarrow = g(n)$ . Suppose that there exists  $n$  such that  $\Phi_{f(n)}(m) = \Phi_n(m)$  for all  $m$ . Then,  $\Phi_n(n) \downarrow = \Phi_{f(n)}(n) \downarrow = g(n)$ , which contradicts the definition of  $g$ . So  $f$  is a free fixed-point function.

Let us show (2)  $\Rightarrow$  (1). Let  $f \leq_T X$  be a free fixed-point function. Then, from  $X$  we compute the function  $g : \mathbb{N} \rightarrow \mathbb{N}$  which on the integer  $n$  creates the code  $e_n$  of the function  $m \mapsto \Phi_{\Phi_n(n)}(m)$ , and returns  $f(e_n)$ . We use here the same abuse of notation as in the proof of the fixed point of Kleene: if  $\Phi_n(n) \uparrow$  then  $m \mapsto \Phi_{\Phi_n(n)}(m)$  denotes the function nowhere defined. Note that  $g$  is total because it does not try to do the computation  $\Phi_n(n)$ . Suppose that  $g$  is not DNC, that is, there is  $n$  such that  $g(n) = \Phi_n(n)$ . By definition of  $g$ ,  $f(e_n) = \Phi_n(n)$ . In particular,  $\Phi_{f(e_n)} = \Phi_{\Phi_n(n)} = \Phi_{e_n}$ . The function  $f$  is therefore not free of a fixed point, which contradicts the definition of  $f$ . So  $g$  is a DNC function.  $\blacksquare$

Let us now see the equivalence between DNC degree and effectively immune degree. The third equivalence of the theorem below is more technical, but is of great interest and will be reused later. It is in fact a reinforcement of the notion of being DNC: Let us consider the sequence  $(A_n)_{n \in \mathbb{N}}$  of c.e. sets defined by

$$A_n = \begin{cases} \{\Phi_n(n)\} & \text{if } \Phi_n(n) \downarrow \\ \emptyset & \text{otherwise} \end{cases}$$

Being of DNC degree is the ability to compute a function  $f$  such that  $f(n) \notin A_n$  for any  $n$ . The third equivalence extends this result to the uniform enumeration  $(W_n)_{n \in \mathbb{N}}$  of all c.e. sets whose number of elements is known to be bounded by an integer  $n$ .

**Theorem 2.6**

Let  $X \in 2^{\mathbb{N}}$ . The following statements are equivalent:

- (1)  $X$  computes a diagonally non-computable function.

(2)  $X$  computes an effectively immune set.

(3)  $X$  compute a function  $h : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  such that for  $e, n \in \mathbb{N}$ ,

$$|W_e| \leq n \Rightarrow h(e, n) \notin W_e$$

PROOF. (1)  $\Rightarrow$  (3): Let  $f \leq_T X$  be a DNC function. We describe a uniform process in  $e, n$  in order to compute a value which under the assumption  $|W_e| \leq n$ , is not in  $W_e$ . For each  $0 \leq i < n$  we compute the code  $u(e, i)$  of the partial computable function which, for any input, looks for the  $i$ -th element  $k$  in the enumeration of  $W_e$ , s' it exists. If such an element  $k$  is found, the function interprets it as the  $n$ -tuple  $\langle k_0, \dots, k_{n-1} \rangle$ , and returns  $k_i$ . Otherwise, the function does not halt. Note that  $\Phi_{u(e, i)}$  is either a constant function or the function defined nowhere.

Let us show that the  $X$ -computable function

$$h(e, n) = \langle f(u(e, 0)), \dots, f(u(e, n-1)) \rangle$$

satisfied (3). Let us reason by the absurd, and suppose that  $h(e, n) \in W_e$  with  $|W_e| \leq n$ . Let's say that  $h(e, n)$  is the  $i$ -th element of  $W_e$  in the order of enumeration. Then,  $\Phi_{u(e, i)}$  is the constant function which finds

$$k = h(e, n) = \langle f(u(e, 0)), \dots, f(u(e, n-1)) \rangle$$

and returns the  $i$ -th element of the tuple, that is,  $f(u(e, i))$ . In particular,  $\Phi_{u(e, i)}(u(e, i)) \downarrow = f(u(e, i))$ , which contradicts the hypothesis that  $f$  is DNC.

(3)  $\Rightarrow$  (2): Let  $h \leq_T X$  be a function satisfying (3). Let  $D_0, D_1, \dots$  be an effective enumeration of all finite sets such that  $n$  is the canonical code of  $D_n$ . Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be a partial computable function such that if  $|W_e| \geq e+1$ , then  $D_{g(e)} \subseteq W_e$  and  $|D_{g(e)}| = e+1$ . We are going to define a sequence  $X$ -computable infinite increasing of integers  $x_0 < x_1 < \dots$  such that for all  $s$ ,

$$\forall e \leq s (|W_e| > e \Rightarrow D_{g(e)} \not\subseteq \{x_i : i \leq s\}) \quad (\star)$$

It follows that  $H = \{x_n : n \in \mathbb{N}\}$  is effectively immune, because if  $W_e \subseteq H$ , then  $|W_e| \leq e$ . The fact that we want  $x_i < x_{i+1}$  is simply so that the set  $H$  is  $X$ -computable. Suppose that we have already defined  $x_0 < \dots < x_s$  satisfying  $(\star)$ . Let

$$W_{v(s)} = \{y : y \leq x_s\} \cup \bigcup_{e \leq s+1 \text{ t.q. } g(e) \downarrow} D_{g(e)}$$

The goal is to use our function  $h$  to find an element which is not in  $W_{v(s)}$ . Let  $x_{s+1}$  be such an element - we'll see how to get it later. Note first that  $x_{s+1}$  is strictly greater than  $x_s$ . Note then that for any  $e \leq s+1$

such that  $|W_e| > e$  and such that  $D_{g(e)} \not\subseteq \{x_i : i \leq s\}$ , then also  $D_{g(e)} \not\subseteq \{x_i : i \leq s+1\}$ . Note finally that for  $e \leq s$  such that  $|W_e| > e$  we have  $D_{g(e)} \not\subseteq \{x_i : i \leq s\}$  by hypothesis, and for  $e = s+1$ , if  $|W_e| > e$ , we necessarily have  $D_{g(e)} \not\subseteq \{x_i : i \leq s\}$  because  $D_{g(e)}$  has then  $s+2$  elements. In all cases we will have  $(\star)$  for the following  $(x_i)_{i \leq s+1}$ . Now let's show how to find  $x_{s+1}$  using  $h$ . The set  $W_{v(s)}$  has at most  $x_s + 1$  plus  $1 + 2 + 3 + \dots + s + 2$  elements, which gives by the sum of the terms of an arithmetic sequence  $t_s = x_s + 1 + (s+2)(s+3)/2$ . We then define  $x_{s+1} = h(v(s), t_s)$ . By definition of  $h$ ,  $x_{s+1} \notin W_{v(s)}$ . In particular,  $x_{s+1} > x_s$ , and  $(\star)$  is satisfied for  $s+1$ .

(2)  $\Rightarrow$  (1): Let  $H \leq_T X$  be an effectively immune set. Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be a total computable function such that if  $W_e \subseteq H$ , then  $|W_e| < g(e)$ . We will show that  $X$  computes a free function with a fixed point. Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the  $X$ -computable function such that  $W_{f(e)}$  is the set of  $g(e)$  first elements of  $H$ . Note that  $f$  is  $X$ -computable, but that the set  $W_{f(e)}$  is a c.e. set which in particular does not need  $X$  for its enumeration: the elements to be enumerated can be seen as “hardcoded” in  $f(e)$ . Let us show that  $f$  is free of a fixed point. Let  $e \in \mathbb{N}$ . If  $W_{f(e)} = W_e$ , then  $W_e \subseteq H$ , but  $|W_e| = |W_{f(e)}| = g(e)$ , which contradicts the choice of  $H$  and  $g$ . As  $X$  computes a free function of fixed point, then by Theorem 2.5,  $X$  computes a DNC function. ■

We will stop there for now. We will see other characterizations of the notion of DNC degree in relation to Algorithmic Randomness. Finally, let us mention a hierarchy which follows naturally from the definition of DNC degree.

**Definition 2.7.** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be a function such that  $2 \leq f(n) \leq f(n+1)$ . A set  $X \subseteq \mathbb{N}$  is of  $\text{DNC}_f$  degree if  $X$  computes a function  $g$  such that  $g(n) < f(n)$  and  $g(n) \neq \Phi_n(n)$  for all  $n$ . ◇

An examination of the definition indicates that the slower the function  $f$  grows, the more difficult it seems for a function to be  $\text{DNC}_f$ . This is indeed the case, as we can see with the following theorem, due to Ambos-Spies, Kjos-Hanssen, Lempert and Slaman [6]:

**Theorem 2.8**

Let  $f$  be a function such that  $2 \leq f(n) \leq f(n+1)$ . There exists a function  $g$  such that  $2 \leq g(n) \leq g(n+1)$  and with  $f < g$  for which  $\text{DNC}_f \subsetneq \text{DNC}_g$ , i.e., there exists a set  $X$  which computes a  $\text{DNC}_g$  function but which does not compute any  $\text{DNC}_f$  function.

As announced earlier, we will see many examples of non-DNC and non-computable degrees, but it is informative to ensure their existence by direct construction.

**Exercise 2.9. (★★)** Show by the finite extension method that there are  $\Delta_2^0$  degrees which are simultaneously non-DNC and non-computable.  $\diamond$

### 3. Arslanov completeness criterion

The Arslanov completeness criterion reflects in a certain way an incompatibility between the c.e. and DNC degrees.

#### C.e. degree

A Turing degree is said to be *computably enumerable* or *c.e.* if it contains a computably enumerable set. We will see in Chapter 13 that  $\mathbf{0}'$  is far from being the only c.e. degree

It is easy to compute a DNC function using the halting problem, as shown in Theorem 2.3. On the other hand, the Arslanov completeness criterion proves that  $\mathbf{0}'$  is the only degree at the same time c.e. and DNC.

#### Theorem 3.1 (Arslanov completeness criterion [8])

Let  $A \in 2^{\mathbb{N}}$  be a c.e. set. Then,  $A$  is Turing complete iff  $A$  computes a DNC function.

PROOF. By Theorem 2.3, if  $A$  is Turing complete, it computes a DNC function. Let's show the converse. Let  $(A_s)_{s \in \mathbb{N}}$  be a c.e. approximation of  $A$ , that is to say with  $\lim_{s \rightarrow \infty} A_s = A$ , and  $A_s \subseteq A_{s+1}$ . Note that if  $A_s \upharpoonright_n = A \upharpoonright_n$  then also for all  $t > s$  we will have  $A_t \upharpoonright_n = A \upharpoonright_n$ .

Let  $\Phi$  be a total functional on the oracle  $A$  and such that  $\Phi(A, n) \neq \Phi_n(n)$  for all  $n$ . Uniformly in  $n$ , we compute the code  $a_n$  of the partial function which on any input  $k$  acts as follows: look for the smallest  $t$  such that  $\emptyset'[t](n) = 1$ , then if that happens and if  $\Phi(A_t \upharpoonright_m, a_n)[t]$  halts for some  $m \leq t$ , then return the value of  $\Phi(A_t \upharpoonright_m, a_n)[t]$  and otherwise diverge. Note that the code  $a_n$  is defined as a function of  $a_n$  itself, which is made possible by the fixed point theorem.

We claim that for all  $n$ , if ever  $\emptyset'(n) = 1$ , then the smallest  $t$  such that  $\Phi(A_t \upharpoonright_m, a_n)[t] \downarrow$  for  $m \leq t$  such that  $A_t \upharpoonright_m = A \upharpoonright_m$ , is strictly greater than the smallest  $t$  such that  $\emptyset'[t](n) = 1$ . Suppose this is not the case, i.e., there exists  $n$  for which  $\emptyset'(n) = 1$  and for which given  $t$  the smallest integer such that  $\emptyset'[t](n) = 1$ , we have also  $\Phi(A_t \upharpoonright_m, a_n)[t] \downarrow$  for  $m \leq t$  such

that  $A_t \upharpoonright_m = A \upharpoonright_m$ . In this case, by the procedure described above, we will have  $\Phi_{a_n}(a_n) = \Phi(A_t \upharpoonright_m, a_n)$  and therefore  $\Phi_{a_n}(a_n) = \Phi(A \upharpoonright_m, a_n)$  which contradicts the fact that  $\Phi$  computes a DNC function on the oracle  $A$ .

The oracle  $A$  can therefore compute  $\emptyset'$  by simply looking for the smallest computation time  $t$  such that  $A_t \upharpoonright_m = A \upharpoonright_m$  and  $\Phi(A_t \upharpoonright_m, a_n)[t] \downarrow$  for  $m \leq t$ , and then looking if  $\emptyset'[t](n) = 1$ . If so then  $\emptyset'(n) = 1$ . Otherwise  $\emptyset'(n) = 0$ . ■

The Arslanov completeness criterion knows several extensions, in particular by Jockusch and al. [100]. We give one of them in an exercise that will allow us to manipulate the principles of the previous proof.

**Exercise 3.2. (★★)** (*Bienvenu and al. [16]*). A function  $g$  is DNC relative to  $C$ , denoted  $\text{DNC}(C)$ , if  $g(n) \neq \Phi_n(C, n)$  for all  $n$ . Let  $X \in 2^{\mathbb{N}}$  of DNC degree, and  $C$  a c.e. set Show that either  $X \oplus C \geq_T \emptyset'$  or  $X$  is of DNC degree relative to  $C$ .

Indication .– Let  $g$  be a DNC function. Define codes  $a_{n,m}$  such that  $\Phi_{a_{n,m}}(a_{n,m})$  is equal to  $\Phi_m(C_s, m)$  for the smallest  $s$  such that  $n \in \emptyset'[s]$ . Show that either there exists  $n$  such that the function  $m \mapsto g(a_{n,m})$  is DNC relative to  $C$ , or  $g \oplus C$  computes  $\emptyset'$ . ◇

Note that the previous exercise implies the Arslanov completeness criterion, because if  $X$  and  $C$  are of the same degree, either  $C \geq_T \emptyset'$ , or  $C$  is of DNC degree relative to  $C$ , which is impossible.

## 4. Hyperimmune functions

We now approach a second family of computational properties, based on the ability to compute fast-growing functions. Exponential functions, or even exponential towers, can be computed, and therefore do not provide additional computational power. We are talking here about functions whose growth rate makes any mental representation difficult.

We have already seen in Section 4-7 that the capacity to grow faster than certain functions allowed to compute the  $\Delta_2^0$  sets. We are now going to study the computational power related to functions which are not dominated by any computable function. The study of these functions was initiated by Martin and Miller [156].

**Definition 4.1.** A function  $g$  *dominates* a function  $f$  if  $g(x) \geq f(x)$  for all  $x \in \mathbb{N}$ . A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is *hyperimmune* if it is not dominated by any computable function. ◇

In particular, the computable functions being stable under finite modifications, a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is hyperimmune if for any computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$ ,  $f(x) > g(x)$  for an infinity of values  $x$ :

**Exercise 4.2.** Show that a function  $f$  is hyperimmune iff for any total computable function  $g$ , there is an infinite number of integers  $x$  such that  $f(x) > g(x)$ .  $\diamond$

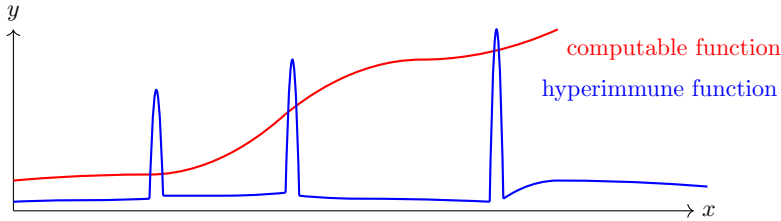


Figure 4.3: Illustration of a hyperimmune function: it does not necessarily grow very fast, but for every computable function  $f$ , it is infinitely often above  $f$ .

Like DNC degrees, Turing degrees for hyperimmune functions are upward-closed. A Turing degree is *hyperimmune* if it contains a hyperimmune function, or equivalently, if one of its elements computes a hyperimmune function. The name “hyperimmune function” comes from the following correspondence with hyperimmune sets.

**Proposition 4.4.** An infinite set  $X$  is hyperimmune iff the function  $p_X : \mathbb{N} \rightarrow \mathbb{N}$  which to  $n$  associates the  $n$ -th element of  $X$  is hyperimmune.  $\star$

PROOF.  $\Rightarrow$  Let  $X$  be a hyperimmune set and let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be a total computable function. Let us show that  $p_X$  is not dominated by  $g$ . Let  $h(x) = g(x) + x + 1$ , and let  $(F_n)_{n \in \mathbb{N}}$  be the c.e. array defined by  $F_n = [h^{(n)}(0), h^{(n+1)}(0)[$ , where  $h^{(n)}$  is the  $n$ -th iteration of  $h$ , with  $h^{(0)}$  the identity function. By hyperimmunity of  $X$ , there exists  $n$  such that  $F_n \cap X = \emptyset$ . This means that  $p_X(h^{(n)}(0)) \geq h^{(n+1)}(0) = h(h^{(n)}(0))$ . By taking  $x = h^{(n)}(0)$ , we have  $p_X(x) \geq h(x) = g(x) + x + 1$ , so  $p_X$  is not dominated by  $g$ .

$\Leftarrow$  Suppose that  $p_X$  is a hyperimmune function. Let us reason by the absurd, supposing that the set  $X$  is not hyperimmune. In other words, there exists a c.e. array  $(F_n)_{n \in \mathbb{N}}$  such that  $F_n \cap X \neq \emptyset$  for all  $n$ . Then, the function  $g : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $g(n) = \max \bigcup_{i \leq n} F_i$  is total computable, and dominates  $p_X$ , contradicting the hyperimmunity of  $p_X$ .  $\blacksquare$

It follows that a degree is hyperimmune if and only if it contains a hyperimmune set.

**Exercise 4.5.** Show that hyperimmune degrees are upward-closed. In other words, if  $X$  computes a hyperimmune function, then the Turing degree of  $X$  contains a hyperimmune function.  $\diamond$

**Exercise 4.6.** Show by the finite extension method that there exists a set of hyperimmune degree.  $\diamond$

The existence of non-computable and non-hyperimmune degrees is currently unclear. We will see in the next section that such degrees do exist, even if we will understand later that this is not granted, in the sense that “many” sets are of hyperimmune degree (see Proposition 10-3.35 and Theorem 19-3.4). In fact, we will have to work to show off one that is not. We see in particular right now that the construction of a non-computable and non-hyperimmune degree cannot be done using  $\emptyset'$ . In particular constructions by finite extensions as seen in Section 4-8, which are all effective in  $\emptyset'$ , will not work.

**Proposition 4.7 (Martin and Miller [156]).** Every non-computable  $\Delta_2^0$  set is hyperimmune.  $\star$

PROOF. Let  $A$  be a non-computable  $\Delta_2^0$  set, and let  $A_0, A_1, \dots$  be a  $\Delta_2^0$  approximation of  $A$ . Recall that the *computation function* (Definition 4-7.7) is defined as the function  $c_A : \mathbb{N} \rightarrow \mathbb{N}$  which to  $x$  associates the smallest integer  $n \geq x$  such that  $A_n \upharpoonright_x = A \upharpoonright_x$ . In particular,  $c_A \leq_T A$ . According to Theorem 4-7.9, any function dominating  $c_A$  computes  $A$ . As  $A$  is not computable,  $c_A$  is not dominated by any computable function, in other words  $c_A$  is hyperimmune.  $\blacksquare$

We now move on to the existence of non-computable and non-hyperimmune degrees, which are called *computably dominated*.

## 5. Computably dominated degrees

By upward-closure of the hyperimmune degrees, a Turing degree is not hyperimmune if any function  $f$  that it computes is dominated by a computable function  $g$  (which depends on  $f$ ).

**Definition 5.1.** A set  $X$  is *computably dominated* if for any function  $f \leq_T X$  there exists a computable function  $g$  dominating  $f$ . A Turing degree  $\mathbf{d}$  is *computably dominated*<sup>a</sup> if all  $X \in \mathbf{d}$  is computably dominated.  $\diamond$

<sup>a</sup>The terminologies “computably dominated” and “hyperimmune-free” coexist.

Note that being computably dominated is a weakness property and is therefore downward-closed in the Turing degrees. In fact the inverse property — being of hyperimmune degree — is a strength property. The simplest example of a computably dominated degree is the Turing degree of computable sets. The aim of this section is to prove the existence of computably dominated degrees different from  $\mathbf{0}$ . We will in fact see a little later (Theorem 8-5.1) that the computably dominated degrees are in uncountable quantity: it is possible to construct an injection  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  such that for all  $X$ , the set  $f(X)$  is of computably dominated degree, and even such that  $f(X)$  and  $f(Y)$  are in different Turing degrees for  $X \neq Y$  (Exercise 8-5.3). The concept at the heart of the construction which will follow is that of *f-tree*.

**Definition 5.2.** An *f-tree* is a total function  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that for all  $\sigma, \tau \in 2^{<\mathbb{N}}$ ,  $\sigma \preceq \tau$  if and only if  $T(\sigma) \preceq T(\tau)$ .  $\diamond$

Let  $T$  be an *f-tree*. Note that the image of  $T$  ( $\text{Im} T$ ) is not closed under prefix in general. Note also that for all  $\sigma \in 2^{<\mathbb{N}}$ ,  $T(\sigma 0)$  and  $T(\sigma 1)$  are two incompatible strings extending  $T(\sigma)$ . Only the image of an *f-tree*  $T$  is important. The tree structure of the domain of  $T$  canonically induces that of  $\text{Im} T$ . The reader may refer to Figure 5.4 for a graphical representation of an *f-tree*.

**Definition 5.3.** Let  $T$  be an *f-tree*. We call *nodes* the elements of  $\text{Im} T$ . A *path* of  $T$  is a sequence  $P \in 2^{\mathbb{N}}$  of which an infinity of initial segments belong to  $\text{Im} T$ . We will denote by  $[T]$  the set of paths of  $T$ .  $\diamond$

Figure 5.4 illustrates the fact that an *f-tree*  $T$  induces an injection  $f_T : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ , computable using the *f-tree*: for  $X \in 2^{\mathbb{N}}$ , the sequence  $f_T(X)$  is computed little by little as being  $T(X \upharpoonright_1) \prec T(X \upharpoonright_2) \prec \dots$ . Moreover by the properties of an *f-tree*, if  $X \neq Y$  then  $f_T(X) \neq f_T(Y)$ .

**Definition 5.5.** A *sub-f-tree* of an *f-tree*  $T$  is an *f-tree*  $S$  such that  $\text{Im} S \subseteq \text{Im} T$ .  $\diamond$

It follows that if  $S$  is a sub-*f-tree* of  $T$ , then  $[S] \subseteq [T]$ . We now have the necessary ingredients to proceed to the proof of the announced theorem.

**Theorem 5.6 (Martin and Miller [156])**

*There is a computably dominated degree  $\mathbf{d} > \mathbf{0}$ .*

**PROOF.** We want to build a set  $A$  of computably dominated degree by satisfying the following requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$\mathcal{R}_e : W_e \neq A \quad \mathcal{S}_e : \Phi_e^A \text{ total} \Rightarrow \Phi_e^A \text{ dominated by a computable function}$

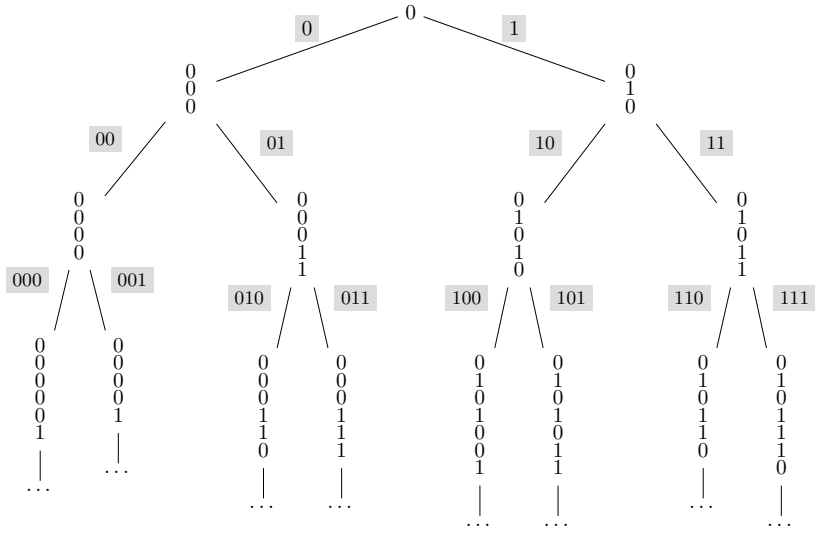


Figure 5.4: Illustration of an f-tree  $T$ , whose domain is represented by light grey strings :  $T(\epsilon) = 0$ ,  $T(0) = 000$ ,  $T(1) = 010$ ,  $\dots$

We are going to build an infinite sequence of computable f-trees  $T_0, T_1, T_2, \dots$  such that for all  $e \in \mathbb{N}$ ,

- (1)  $T_{e+1}$  is a sub-f-tree of  $T_e$ ;
- (2)  $|T_e(\epsilon)| \geq e$ ;
- (3) For any path  $P \in [T_{2e+1}]$ , the requirement  $\mathcal{R}_e$  is satisfied;
- (4) For any path  $P \in [T_{2e+2}]$ , the requirement  $\mathcal{S}_e$  is satisfied.

**Satisfaction of a requirement  $\mathcal{R}_e$ .** Let  $T$  be a computable f-tree. Since  $\sigma_0 = T(0)$  and  $\sigma_1 = T(1)$  are incompatible strings, there is some  $i \in \mathbb{N}$  such that  $\sigma_0(i) \neq \sigma_1(i)$ . So either  $\sigma_0(i) \neq W_e(i)$  or  $\sigma_1(i) \neq W_e(i)$ . Suppose we are in the first case, the other being symmetrical. Then, the sub-tree  $S$  of  $T$  defined by  $S(\rho) = T(0\rho)$  ensures that for all paths  $P \in [S]$ ,  $\sigma_0 \leq P$ , therefore  $P \neq W_e$ . Moreover,  $S$  is computable in  $T$ , therefore computable.

**Satisfaction of a requirement  $\mathcal{S}_e$ .** Let  $T$  be a computable f-tree. We are going to build a computable sub-tree  $S$  such that either  $\Phi_e^P$  is partial for all  $P \in [S]$ , or  $\Phi_e^P$  is total and dominated by the same computable function for all  $P \in [S]$ . Note that in addition to satisfying the requirement  $\mathcal{S}_e$ , the functional  $\Phi_e^P$  will be either partial or total whatever the path  $P$ . Two cases arise:

Case 1: There is a node  $\sigma \in \text{Im } T$  and an input  $x \in \mathbb{N}$  such that  $\Phi^\tau(x) \uparrow$  for any  $\tau \in \text{Im } T$  such that  $\tau \succeq \sigma$ . Let  $\rho \in 2^{<\mathbb{N}}$  be such that  $T(\rho) = \sigma$ . Then, the subtree  $S$  of  $T$  defined by  $S(\mu) = T(\rho\mu)$  ensures that for all paths  $P \in [S]$ , arbitrarily long initial segments  $\tau$  of  $P$  will satisfy  $\Phi^\tau(x) \uparrow$ . By the use property, it follows that  $\Phi^P(x) \uparrow$ .

Case 2: For any node  $\sigma \in \text{Im } T$  and any input  $x \in \mathbb{N}$ , there exists  $\tau \in \text{Im } T$  such that  $\sigma \preceq \tau$  and  $\Phi^\tau(x) \downarrow$ . We will define  $S$ , a computable sub-tree of  $T$  such that for all  $\rho \in 2^{<\mathbb{N}}$ ,  $\Phi_e^{S(\rho)}(|\rho|) \downarrow$ . We compute  $S(\epsilon)$  as being the first node of  $\text{Im } T$  which we find such that  $\Phi_e^{S(\epsilon)}(0) \downarrow$ . Suppose we have computed  $S(\rho)$ . Let  $\mu$  be such that  $S(\rho) = T(\mu)$ . We compute  $S(\rho 0)$  and  $S(\rho 1)$  as follows: for each  $i < 2$ ,  $S(\rho i)$  is the first node  $\tau \in \text{Im } T$  that we find, with  $\tau \succeq T(\mu i)$  and such that  $\Phi^\tau(|\rho| + 1) \downarrow$ . The algorithm which searches for the nodes  $S(\rho 0)$  and  $S(\rho 1)$  will always succeed, by the assumption that we are in case 2. By construction,  $S$  is indeed an f-tree, and  $\text{Im } S \subseteq \text{Im } T$ . Even more,  $\Phi_e^P$  is a total function for all  $P \in [S]$ .

Let us show that  $\Phi_e^P$  is dominated by a computable function for all  $P \in [S]$ . Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be the function defined by  $g(n) = \max\{\Phi_e^{S(\rho)}(n) : |\rho| = n\}$ . Then, for all  $n \in \mathbb{N}$  and  $P \in [S]$ ,  $\Phi_e^P(n) \leq g(n)$ . The sub-f-tree  $S$  of  $T$  satisfies the requirement  $\mathcal{S}_e$ .

Finally, note to satisfy point (2) above that for any computable f-tree  $T$  and any  $n$ , there exists a sub-f-tree  $S$  such that  $S(\epsilon) \geq n$ . Indeed, it suffices to fix a string  $\rho$  of length  $n$ , and to define  $S(\mu) = T(\rho\mu)$ . We can therefore combine the satisfactions of the different requirements and this last observation to construct a sequence of f-trees  $T_0, T_1, \dots$  satisfying the properties (1) (2) (3) and (4) given above.

### Remark

Each f-tree  $T_e$  of the sequence taken independently is computable, but the sequence  $T_0, T_1, \dots$  is not itself computable.

In order to complete the proof, we need the following lemma.

**Lemma 5.7.** The intersection  $\bigcap_e [T_e]$  contains exactly one element. ★

PROOF. Since  $\text{Im } T_{e+1} \subseteq \text{Im } T_e$  and since any string of  $\text{Im } T_e$  is an extension of  $T_e(\epsilon)$ , it is clear that we have  $T_0(\epsilon) \preceq T_1(\epsilon) \prec T_2(\epsilon) \prec \dots$ .

Moreover as  $|T_e(\epsilon)| \geq e$ , the sequence  $T_0(\epsilon) \preceq T_1(\epsilon) \prec T_2(\epsilon) \prec \dots$  converges to a unique infinite sequence  $X \in 2^\mathbb{N}$ . Since  $X$  has an infinity of prefixes in each  $T_e$  then  $X \in [T_e]$  for all  $e$  and therefore  $X \in \bigcap_e [T_e]$ . ■

Let  $A$  be the element of  $\bigcap_e [T_e]$ . For all  $e \in \mathbb{N}$ , as  $A \in [T_{2e+2}]$ , then the requirement  $\mathcal{R}_e$  is satisfied for  $A$ , hence  $W_e \neq A$ . Moreover, as  $A \in [T_{2e+2}]$ ,

then the requirement  $\mathcal{S}_e$  is satisfied for  $A$ , so if  $\Phi_e^A$  is total,  $\Phi_e^A$  is dominated by a computable function. It follows that  $A$  is computably dominated and not computable. ■

### Remark

A careful analysis of the previous construction shows that the oracle  $\emptyset''$  is sufficient to compute the sequence  $(T_e)_{e \in \mathbb{N}}$  uniformly. Indeed, the only non-computable parts are the case analysis, which are  $\Sigma_2^0$  properties. Thus, there exists a degree  $\mathbf{d} > \mathbf{0}$  both  $\Delta_3^0$  and computably dominated. As we saw with Proposition 4.7, it is not possible to lower this bound to  $\Delta_2^0$ .

We will find the structure of f-trees in Chapter 14 to show the existence of minimal Turing degrees.

**Exercise 5.8. (★★)** Build an f-tree  $T$ , by the finite extension method, such that all  $X, Y \in [T]$  are in different Turing degrees. ◇

Note that we had announced the existence of an uncountable quantity of computably dominated degrees. This will be done with Theorem 8-5.1. We now give some equivalences to better understand this notion.

### 5.1. Truth-table reduction

We suppose in what follows that the functionals considered try to compute sets of integers and not functions, that is, if  $\Phi(Y, n) \downarrow$  for a certain oracle  $Y$  and a certain integer  $n$ , then  $\Phi(Y, n) \downarrow \in \{0, 1\}$  (if  $\Phi(Y, n) \downarrow \notin \{0, 1\}$  we will consider that the functional diverges).

Consider a functional  $\Phi$  and an oracle  $X$  such that  $\forall n \Phi(X, n) \downarrow \in \{0, 1\}$ . Given another set  $Y \neq X$ , there is no reason why we should also have  $\forall n \Phi(Y, n) \downarrow$ . Totality with one oracle is not necessarily totality with others, and it should not be very hard for the reader to construct such examples. But are such partialities necessary? Assuming  $X \geq_T Y$ , can we still compute  $Y$  from  $X$  via a total functional over all oracles? We will see that this is not necessarily the case via a restriction of the notion of Turing reduction.

**Definition 5.9.** For any set  $X, Y \subseteq \mathbb{N}$ , we say that  $X$  is *truth-table reducible* to  $Y$ , and we write  $X \leq_{tt} Y$  if there is a functional  $\Phi$  such that  $\Phi(Y) = X$  and such that  $\Phi(Z)$  is total for any oracle  $Z$ . We write  $X \equiv_{tt} Y$  if  $X \leq_{tt} Y$  and  $Y \leq_{tt} X$ . We write  $X <_{tt} Y$  if  $X \leq_{tt} Y$

and  $Y \not\leq_{tt} X$ . We call *truth-table degrees* the equivalence classes of the relation  $\equiv_{tt}$ .  $\diamond$

Notice the notation  $\Phi(Y) = X$  meaning  $\forall n \Phi(Y, n) = X(n)$ . We will see several definitions equivalent to the truth-table reduction, starting with the one justifying its name.

**Definition 5.10.** A reduction by truth table is given by a computable sequence of pairs  $(\langle C_{0,n}, C_{1,n} \rangle)_{n \in \mathbb{N}}$  such that for all  $n$  the set  $C_{0,n} \cup C_{1,n} \subseteq 2^{<\mathbb{N}}$  contains exactly all the strings of a certain size  $m_n$ , and such that  $C_{0,n} \cap C_{1,n} = \emptyset$ . The set  $X$  computes  $Y$  via this reduction if for all  $n$  we have  $Y(n) = i$  iff  $\sigma \in C_{i,n}$  for a prefix  $\sigma$  of  $X$ .  $\diamond$

The sets  $C_{i,n}$  are the “truth tables”. Any oracle  $X$  has a prefix in  $C_{0,n} \cup C_{1,n}$ . If the prefix belongs to  $C_{0,n}$  then  $X$  computes 0 on the input  $n$  and if the prefix belongs to  $C_{1,n}$  then  $X$  computes 1 on the input  $n$ . Let us now see the different equivalences to the notion of truth-table reduction.

**Theorem 5.11**

Let  $X, Y$  be sets. The following statements are equivalent:

- (1)  $Y \leq_{tt} X$
- (2)  $X$  computes  $Y$  via a reduction by truth table.
- (3) There exists a functional  $\Phi$  and a total computable function  $b : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\Phi(X, n)[b(n)] \downarrow = Y(n)$  for all  $n$ .

PROOF. Let us show (1)  $\Rightarrow$  (2). Suppose  $\Phi(X) = Y$  via a total functional  $\Phi$  over all oracles. Given  $n$ , we search for the smallest computation time  $t_n$  such that for a certain  $m_n \leq t_n$  and for any string  $\sigma \in 2^{\mathbb{N}}$  of size  $m_n$  we have  $\Phi(\sigma, n)[t_n] \downarrow$ . In order to show that such a computing time  $t_n$  necessarily exists for all  $n$ , we must anticipate a little on Definition 8-1.1 of tree and Lemma 8-1.4 of König to come. Let us assume absurdly that for a certain  $n$ , one cannot find  $t_n$ . This implies in particular that for any  $m$ , there exists a string  $\sigma$  of size  $m$  such that  $\forall t \Phi(\sigma, n)[t] \uparrow$ . Moreover if  $\forall t \Phi(\sigma, n)[t] \uparrow$  and  $\tau \preceq \sigma$  then also  $\forall t \Phi(\tau, n)[t] \uparrow$ . We can therefore construct an infinite tree  $T$  such that  $\sigma \in T$  implies  $\forall t \Phi(\sigma, n)[t] \uparrow$ . According to König’s lemma  $T$ , contains an infinite path  $Y$ , which is therefore such that  $\Phi(Y, n) \uparrow$  which contradicts the fact that  $\Phi$  is total over all its oracles. We can therefore at each step find  $t_n$  and  $m_n \leq t_n$ , with the set  $C_{0,n}$  of strings of sizes  $m_n$  on which the computation returns 0 and the set of strings  $C_{1,n}$  of sizes  $m_n$  on which the computation returns 1.

Let us show (2)  $\Rightarrow$  (3). Given a reduction by truth table given by the computable sequence  $(\langle C_{0,n}, C_{1,n} \rangle)_{n \in \mathbb{N}}$ , for any input  $n$  we can limit the

computation time that the functional takes to halt on  $n$  with any oracle: this is simply the time required to produce the computation of  $\langle C_{0,n}, C_{1,n} \rangle$ .

Let us show (3)  $\Rightarrow$  (1). Let  $\Phi$  be a functional and  $b : \mathbb{N} \rightarrow \mathbb{N}$  a total computable function such that  $\Phi(Y, n)[b(n)] \downarrow = X(n)$  for all  $n$ . Then, we construct the functional  $\Psi$  which on all oracles  $Z$  and on any input  $n$  launches the computation of  $\Phi(Z, n)$  for  $b(n)$  steps. If the computation returns a value in  $b(n)$  steps, then  $\Psi$  returns that value, otherwise  $\Psi$  returns 0. The result of the computation is the same between  $\Phi$  and  $\Psi$  on the oracle  $Y$ , but  $\Psi$  is now total on all oracles. ■

We now show that the sets  $X$  for which  $X \geq_T Y$  implies  $X \geq_{tt} Y$  are exactly the computably dominated sets.

**Theorem 5.12 (Jockusch [96], Martin (non publié))**

*A set  $X$  is computably dominated iff  $Y \leq_T X \Leftrightarrow Y \leq_{tt} X$  for all  $Y \in 2^{\mathbb{N}}$ .*

PROOF. Suppose  $X$  is computably dominated. Suppose  $X \geq_T Y$  via the functional  $\Phi$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be such that  $\Phi(X, n)[f(n)] \downarrow = Y(n)$  for all  $n$ . Note that  $f$  is an  $X$ -computable function. There is therefore a computable function  $g > f$ . We therefore have  $\Phi(X, n)[g(n)] \downarrow = Y(n)$ . According to Theorem 5.11 we have then  $X \geq_{tt} Y$ .

Suppose now that for all  $Y$  we have  $Y \leq_T X \Leftrightarrow Y \leq_{tt} X$ . Then, also for any function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , we have  $f \leq_T X \Leftrightarrow f \leq_{tt} X$ , via the canonical representation of  $f$  by a sequence of  $2^{\mathbb{N}}$ . Let  $f \leq_T X$ . Let us show that  $f$  is dominated by a computable function  $g$ . By hypothesis,  $f \leq_{tt} X$ , therefore by Theorem 5.11, there exists a functional  $\Phi$  and a total computable function  $b : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\Phi(X, n)[b(n)] \downarrow = f(n)$  for all  $n$ . We can assume without loss of generality that if  $\Phi(X, n)[b(n)] \downarrow$ , then the use of the computation is lower than  $b(n)$  (if this is not the case we can slow down the computation so that it is), therefore  $\Phi(X \upharpoonright_{b(n)}, n)[b(n)] \downarrow$ . Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be the function which for an input  $n$ , executes  $\Phi(\sigma, n)[b(n)]$  for any  $\sigma \in 2^{<\mathbb{N}}$  of length  $b(n)$ , and returns the maximum of the values obtained. In particular,  $g(n) \geq \Phi(X \upharpoonright_{b(n)}, n)[b(n)] = f(n)$ . The function  $g$  dominates  $f$ . It follows that  $X$  is of computably dominated degree. ■

We now have three notions of reductions: the many-one reduction, the truth-table reduction and the Turing reduction. The following proposition recapitulates some results seen so far, which attest that none coincides with another.

**Proposition 5.13.** For all  $X, Y$  we have  $X \leq_m Y \Rightarrow X \leq_{tt} Y \Rightarrow X \leq_T Y$ . No reverse implication is true in the general case. ★

PROOF. The implications are clear. Let us show that no reciprocal implication holds. The set  $\mathbb{N} \setminus \emptyset'$  is  $tt$ -reducible to  $\emptyset'$  but not  $m$ -reducible to  $\emptyset'$  because it would then be  $\Sigma_1^0$  according to Proposition 5-4.3, and therefore  $\emptyset'$  would be computable, which is false. The existence of sets  $X, Y$  such that  $X \geq_T Y$  but  $X \not\leq_{tt} Y$  is a consequence of Theorem 5.12 and of the fact that non-computably dominated degrees exist (for example  $\mathbf{0}'$ ). ■

## 6. Martin's domination theorem

Hyperimmune functions are by definition those functions which are not dominated by any computable function. It is natural to wonder about the computational power of the functions which dominate all the computable functions. Of course, no function  $f : \mathbb{N} \rightarrow \mathbb{N}$  dominates all the computable functions, because the constant function  $f(0) + 1$  is not dominated by  $f$ . We can however weaken the property, and wonder about the computational power of a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for any computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$ ,  $f$  *eventually* dominates  $g$ , that is, for all but finitely many inputs.

### Notation

We will use the notations  $\forall^\infty m$  and  $\exists^\infty m$  to signify respectively  $\exists n \forall m > n$  and  $\forall n \exists m > n$ . Thus,  $\forall^\infty m$  means “for all but finitely many  $m$ ” and  $\exists^\infty m$  means “for infinitely many  $m$ ”.

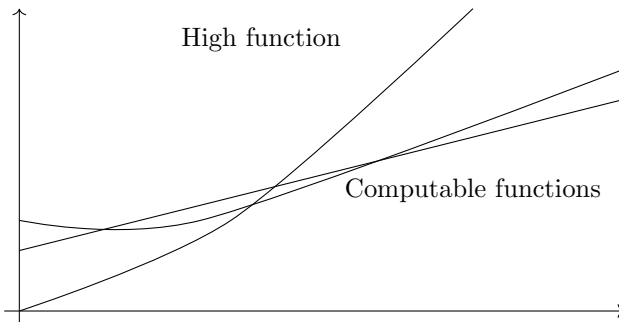


Figure 6.1: Illustration of a dominating function, which for every computable function  $f$ , is always above  $f$  after some point.

Martin's domination theorem ((1)  $\Leftrightarrow$  (2) in the following theorem [147]) gives a magnificent characterization of the Turing degrees of these functions.

The equivalence (2)  $\Leftrightarrow$  (3) shown by Jockusch [97] came later and is also of interest. Recall that a set  $A$  is high if  $A' \geq_T \emptyset''$  (see Definition 4-10.1).

**Theorem 6.2**

*Let  $A \subseteq \mathbb{N}$  be a set. The following statements are equivalent:*

- (1)  *$A$  is high.*
- (2)  *$A$  computes a function  $g : \mathbb{N} \rightarrow \mathbb{N}$  which eventually dominates all computable functions. That is to say that for any computable function  $f$  we have  $\forall^\infty n \, f(n) \leq g(n)$ .*
- (3)  *$A$  computes a list  $(X_n)_{n \in \mathbb{N}}$  containing (possibly with repetitions) exactly the computable sets.*

PROOF. Let us show (1)  $\Rightarrow$  (2). Suppose  $A$  is high. We thus have a  $\Delta_2^0(A)$  description of  $\emptyset''$ , that is to say an  $A$ -computable function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that  $\lim_{s \rightarrow \infty} f(n, s) = \emptyset''(n)$  for all  $n$ . Note that being the code of a total function is a  $\Pi_2^0$  property. We can in particular, by using the fact that  $\emptyset''$  is  $\Sigma_2^0$ -complete (see Proposition 5-5.3), compute for all  $e$  a code  $a_e$  such that  $\emptyset''(a_e) = 0$  iff  $\Phi_e$  is a total function.

We define  $g$  as follows: on the input  $t$ , for any functional  $\Phi_e$  for  $e \leq t$ , we search for the smallest computation time  $s \geq t$  such that  $f(a_e, s) = 1$  or such that  $\Phi_e(t)[s] \downarrow$ . Note that one of the two events necessarily happens: either  $\Phi_e$  is total and therefore  $\Phi_e(t) \downarrow$ , or  $\Phi_e$  is partial and therefore  $\lim_{s \rightarrow \infty} f(a_e, s) = \emptyset''(a_e) = 1$ . In the first case, we define  $v_{t,e} = 0$  and in the second  $v_{t,e} = \Phi_e(t)$ . We finally define  $g(t) = \sum_{e \leq t} v_{t,e}$ .

It is clear that for any total function  $\Phi_e$ , starting from the smallest  $t \geq e$  such that  $f(a_e, s) = 0$  for  $s \geq t$ , we will have  $g(s) \geq \Phi_e(s)$  for all  $s \geq t$ . So  $g$  eventually dominates all computable functions.

Let us show (2)  $\Rightarrow$  (3). Suppose now that  $A$  computes a function  $g$  which eventually dominates any computable function. We use the fact that if  $\Phi_e$  is total with value in  $\{0, 1\}$ , then the computable function  $t : \mathbb{N} \rightarrow \mathbb{N}$  which on  $n$  returns the smallest computation time such that  $\Phi_e(n)[t(n)] \downarrow$  for all  $n$ , is eventually dominated by  $g$ .

For each functional  $\Phi_e$  we compute the set  $Y_e$  as being  $Y_e(n) = \Phi_e(n)[g(n)]$  if  $\Phi_e(n)[g(n)] \downarrow \in \{0, 1\}$  and  $Y_e(n) = 0$  otherwise. The list  $(Y_e)_{e \in \mathbb{N}}$  therefore contains, up to finite modification, exactly the computable sets. To obtain them all, we finally compute the sequence  $(X_n)_{n \in \mathbb{N}}$  as being all the possible finite modifications of the sets  $Y_e$ .

Let us show (3)  $\Rightarrow$  (1). The reader can use Figure 6.4 to understand this implication. Suppose that  $A$  computes a list  $(X_n)_{n \in \mathbb{N}}$  containing

(possibly with repetitions) exactly the computable sets. Let  $P = \{e : \forall x_1 \exists x_2 R(e, x_1, x_2)\}$  be an arbitrary  $\Pi_2^0$  set. Let us show that  $P$  is  $\Sigma_2^0(A)$ . For that, one defines uniformly for all  $e$  a partial computable function  $f_e : \mathbb{N} \rightarrow \{0, 1\}$  such that:

- (a)  $e \in P$  implies that  $f_e$  is a total computable function.
- (b)  $e \notin P$  implies that  $f_e$  is a partial function which cannot be completed into a total computable function.

Let  $e$  be fixed. We describe a uniform process in  $e$ . At the stage of computation  $t$ , for any value  $n$  smaller than  $t$  and such that  $f_e$  does not halt for the moment on  $n$ , we proceed as follows : If  $\Phi_n(n)[t] \downarrow \neq 0$  we define  $f_e(n) = 0$ . Otherwise if  $\Phi_n(n)[t] \downarrow = 0$  we define  $f_e(n) = 1$ . Otherwise if for all  $k \leq n$  there exists  $m_k \leq t$  such that  $R(e, k, m_k)$  then we define  $f_e(n) = 0$ .

The process is clearly computable. Let us show (a). Suppose  $e \in P$ . Then, for all  $n$  there exists a smallest  $t$  such that for all  $k \leq n$  there exists  $m_k \leq t$  for which  $R(e, k, m_k)$ . When that happens then  $f_e(n)$  takes a value at step  $t$  if it has not taken one so far. So  $f_e$  is total. Let us now show (b). Suppose  $e \notin P$ . Let  $n$  be the largest integer such that for all  $k \leq n$  there exists  $m_k$  for which  $R(e, k, m_k)$ . For  $m > n$ ,  $f_e(m)$  halts iff  $\Phi_m(m)$  halts in which case  $f_e(m) \neq \Phi_m(m)$ . Suppose absurdly that  $f_e$  has a computable completion. Then, by the padding lemma (Lemma 3-5.1) it has a computable code completion  $a > n$ . In this case  $f_e(a) = \Phi_a(a)$  which contradicts the definition of  $f_e$ . So we have (b).

It follows that  $P$  can be written as a  $\Sigma_2^0(A)$  set as follows.

$$P = \{e : \exists n \forall m \forall t f_e(m)[t] \uparrow \vee f_e(m)[t] \downarrow = X_n(m)\}.$$

Indeed if  $e \in P$  then  $f_e$  is computable and therefore coincides with a certain  $X_n$ . Conversely if  $e \notin P$  then  $f_e$  has no computable completion and therefore no completion in  $(X_n)_{n \in \mathbb{N}}$ . Since  $P$  is  $\Sigma_2^0(A)$  and  $\bar{P}$  is  $\Sigma_2^0$  by definition, the set  $P$  is then  $\Delta_2^0(A)$  and therefore  $P$  is  $A'$  computable. It suffices to apply this for  $P = \mathbb{N} \setminus \emptyset''$  to get that  $\emptyset''$  is  $\Delta_2^0(A)$  and therefore  $A'$ -computable. ■

The implication (3)  $\Rightarrow$  (1) of Theorem 6.2 is far from obvious. We hope the reader will appreciate the argument, the subtle complexity of which is characteristic of Jockusch's work. The following exercise gives similar alternative characterizations, simpler to demonstrate:

**Exercise 6.3. (★)** Show the following equivalences:

- (1)  $A$  computes a function which eventually dominates any computable function.

- (2)  $A$  computes a sequence  $(f_n)_{n \in \mathbb{N}}$  containing all the computable functions from  $\mathbb{N}$  to  $\mathbb{N}$ .
- (3)  $A$  compute a sequence  $(X_n)_{n \in \mathbb{N}}$  containing only infinite sets, and containing all infinite computable sets.

◇

$f_e$	$X_0$	$X_1$	$X_2$	$\dots$
$f_e(0)$	$X_0(0)$	$X_1(0)$	$X_2(0)$	
$f_e(1)$	$X_0(1)$	$X_1(1)$	$X_2(1)$	
↑	$X_0(2)$	$X_1(2)$	$X_2(2)$	
↑	$X_0(3)$	$X_1(3)$	$X_2(3)$	
$f_e(4)$	$X_0(4)$	$X_1(4)$	$X_2(4)$	
↑	$X_0(5)$	$X_1(5)$	$X_2(5)$	
$f_e(6)$	$X_0(6)$	$X_1(6)$	$X_2(6)$	
↑	$X_0(7)$	$X_1(7)$	$X_2(7)$	
$\dots$	$\dots$	$\dots$	$\dots$	

Figure 6.4: Illustration of the direction (3)  $\Rightarrow$  (1) of Theorem 6.2 : if  $e \notin P$ , one constructs a partial computable function  $f_e : \mathbb{N} \rightarrow \{0, 1\}$  which has no total computable completion. In the illustration, the  $f_e$  function cannot equal any of the sets  $X_0, X_1, \dots$  on all the values for which it is defined. Indeed, the list  $(X_n)_{n \in \mathbb{N}}$  contains only computable sets. In the inverse case, the  $f_e$  function is total computable, hence equals at least one of the sets  $X_i$ , since the list  $(X_n)_{n \in \mathbb{N}}$  contains all the computable sets.

Let us discuss a little about the characterization (1)  $\Leftrightarrow$  (2) of Theorem 6.2, illustrated by Figure 6.1. Being able to compute a very fast growing function from  $\mathbb{N}$  into  $\mathbb{N}$  is likely to provide a lot of computational power. Thus, for example, as we have seen, computing a function which dominates the halting time of computer programs makes it possible to compute  $\emptyset'$ .

The previous theorem indicates that there is a first level between the functions able to compute the halting set simply because they grow very fast, and the functions with computable growth: there exists, according to Proposition 4-10.2, high sets that do not compute the halting set, and therefore of functions of “intermediate” growth. We will also see that the faster a function grows, the more computational power it has. A function that grows fast enough will be able to compute  $\emptyset''$ , one that grows even

faster will be able to compute  $\emptyset'''$ , and so on. In spite of everything, we will see with Theorem 29-5.4 that there is a limit to the computational power conferred by rapid growth. The class of sets computable by any function which grows “sufficiently” fast has a precise characterization and remains countable.

**Exercise 6.5. (\*\*)** The notion of maximal c.e. set was introduced in Exercise 3-7.13. A c.e. set  $X$  is maximal if  $\mathbb{N} \setminus X$  is infinite, and if any c.e. set  $Y \supseteq X$  is such that  $Y \setminus X$  is finite or such that  $\mathbb{N} \setminus Y$  is finite. Let  $X$  be a maximal c.e. set. Show that  $X$  is high.  $\diamond$

## 7. High or DNC degrees

We end this chapter with a result that combines high and DNC degrees, in order to obtain a natural characterization in terms of computational power: the possibility of computing a function which differs almost everywhere from any computable function.

### Theorem 7.1 (Kjos-Hanssen, Merkle and Stephan [112])

Let  $X \in 2^{\mathbb{N}}$ . The following statements are equivalent:

- (1)  $X$  is of high or DNC degree.
- (2)  $X$  computes a function which is different almost everywhere from any computable function.

**PROOF.** Let us show  $(1) \Rightarrow (2)$ . Suppose first that  $X$  is high. Let  $g \leq_T X$  be a function which eventually dominates any computable function. So also,  $m \mapsto g(m) + 1$  is almost everywhere different from any computable function. Now suppose that  $X$  is DNC. For each  $n$  we compute the code  $e_n$  such that  $W_{e_n}$  enumerates all the values  $\Phi_e(n)$  for  $e \leq n$  for which  $\Phi_e(n) \downarrow$ . According to Theorem 2.6,  $X$  computes a function  $h : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that for all  $e, n \in \mathbb{N}$ , if  $|W_e| \leq n$  then  $h(e, n) \notin W_e$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the  $X$ -computable function defined by  $f(n) = h(e_n, n)$ . We then have  $f(n) \notin W_{e_n}$  for all  $n$ . Thus,  $f(n)$  is different from  $\Phi_0(n), \Phi_1(n), \dots, \Phi_n(n)$  for all  $n$ , so  $f$  differs almost everywhere from any computable function (and even from any partial computable function halting on an infinity of values).

Let us show  $(2) \Rightarrow (1)$ . If  $X$  is high there is nothing to check. So suppose that  $X$  is not high. Let  $g \leq_T X$  be such that total  $\Phi_e$  implies  $\forall^\infty m \ g(m) \neq \Phi_e(m)$ . We claim that we also have  $\forall^\infty e \ g(e) \neq \Phi_e(e)$ . Let us assume the contrary to be absurd. Let  $f$  be the  $X$ -computable function which on  $n$  returns the smallest computation time  $t$  such that

we have  $g(m) = \Phi_m(m)[t] \downarrow$  for an integer  $m > n$ . Since  $X$  is not high there exists a computable function  $b$  such that  $\exists^\infty n \ b(n) \geq f(n)$ . Note that we can assume without loss of generality  $b(n) \leq b(n+1)$ . We now define the total computable function  $h$  by  $h(n) = \Phi_n(n)[b(n)]$  if  $\Phi_n(n)[b(n)] \downarrow$  and  $h(n) = 0$  otherwise. Suppose now  $b(n) > f(n)$ . By definition of  $f$ , there exists a value  $m > n$  such that  $\Phi_m(m)[f(n)] \downarrow = g(m)$  and therefore such that  $\Phi_m(m)[b(n)] \downarrow = g(m)$ . As  $b(m) \geq b(n)$  we will have  $h(m) = \Phi_m(m)[b(m)] \downarrow = g(m)$ . As we can restart the argument for arbitrarily large values of  $n$ , we will have  $h(m) = g(m)$  for an infinity of  $m$ . This contradicts the fact that  $g$  differs almost everywhere from any computable function. So we have  $\forall^\infty e \ g(e) \neq \Phi_e(e)$ . It suffices then to modify a finite number of values of  $g$  to obtain a DNC function. ■

Let us note, to give all its brilliance to the preceding theorem, that it is possible to construct DNC degrees which are not high (by combining Corollary 18-4.3 with Corollary 19-3.9 or more simply by considering Corollary 8-6.6) just like high degrees which are not are not DNC (see Corollary 10-3.34).



# Chapter 8

## $\Pi_1^0$ classes and PA degrees

We have so far mainly concentrated on the study of sets of natural integers taken individually, or in an equivalent manner, of functions over the integers. We will now turn to the study of *classes* of sets or functions, i.e., sets of sets of integers. We have already seen many classes of sets, in particular the class of sets of high degree  $\{X \in 2^{\mathbb{N}} : X' \geq_T \emptyset''\}$ , or that of sets of low degree  $\{X \in 2^{\mathbb{N}} : X' \equiv_T \emptyset'\}$ .

### Set vs class

In order to distinguish sets of integers from subsets of Cantor space, we will call them “sets” and “classes” when we are in a context relating to the study of sets  $X \in 2^{\mathbb{N}}$  or classes  $\mathcal{A} \subseteq 2^{\mathbb{N}}$ , respectively.

The classes that we will consider will be defined by predicates, and the study of the complexity of these predicates will allow us to deduce information about the elements that the class contains.

The study of classes will quickly prove to be central in the study of sets of integers. We start here with the classes of the simplest possible complexity: the effectively open and closed classes of Cantor space. The study of classes of higher complexity will be continued in Chapter 17. Despite the apparent simplicity of open and closed classes, we will quickly see the wide range of possibilities and the great richness they contain.

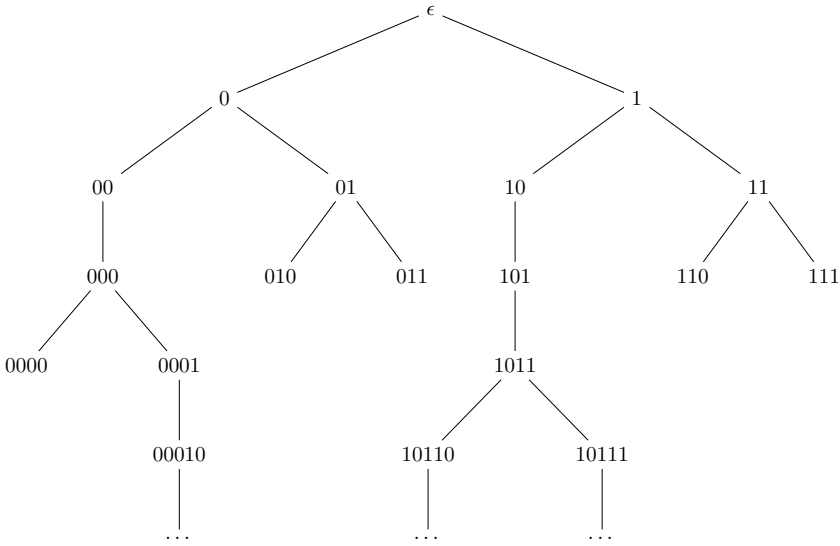


Figure 1.2: Illustration of a tree. The root  $\epsilon$  is the empty string. Each node has potentially a left and a right successor. If both, then it is a *branching node*, and if none, it is a *leaf*.

## 1. Binary trees

We have defined the notion of f-tree in order to show the existence of a non-computable computably dominated degree (Theorem 7-5.6). We introduce here a similar notion and in a certain way more primitive, namely, the notion of tree.

**Definition 1.1.** A set  $T \subseteq 2^{<\mathbb{N}}$  is a *tree* if  $T$  is closed under prefix, i.e., for all  $\sigma \in T$  and  $\tau \preceq \sigma$ , then  $\tau \in T$ .  $\diamond$

The elements  $\sigma$  of a  $T \subseteq 2^{<\mathbb{N}}$  are called *nodes*. A node is *branching* if both  $\sigma 0, \sigma 1 \in T$ . In the opposite case it is *non branching*. Based on the graphical representation of a tree, we will consider that  $\sigma 0$  and  $\sigma 1$  are on the left and on the right of  $\sigma$ , respectively. Accordingly, if  $\sigma 0 \in T$  then  $\sigma 0$  is a *left successor* of  $\sigma$ . If  $\sigma 1 \in T$  then  $\sigma 1$  is a *right successor* of  $\sigma$ . A node with no successor will be called a *leaf*.

**Definition 1.3.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a tree. A *path* through the tree  $T$  is a sequence  $P \in 2^{\mathbb{N}}$  such that  $P \upharpoonright_n \in T$  for all  $n \in \mathbb{N}$ . We denote by  $[T]$  the class of paths of  $T$ .  $\diamond$

Intuitively, a path  $P$  can be thought of as a sequence of binary instructions,

literally indicating “a path” to follow through the tree. A bit at 0 in the path tells us to continue our journey following the left successor and a bit at 1 tells us to follow the right successor. We only consider infinite paths.

### Remark

Contrary to Graph Theory, a path is not represented as a set of nodes in the tree. However, there is a computable bijection between a path  $P$  and the set  $\{P \upharpoonright_n : n \in \mathbb{N}\}$ . Representing a path as an infinite binary sequence is therefore mainly a conventional choice which will prove to be very useful later.

If  $T$  is a finite tree, i.e., which has only a finite number of nodes, then  $[T]$  is necessarily the empty set. What about the converse? This is a central tool on trees: König’s lemma, which states that any infinite, finitely-branching tree has an infinite path. Note that trees  $T \subseteq 2^{<\mathbb{N}}$  are necessarily 2-branching. The restriction of König’s lemma to binary trees is known as *weak* König’s lemma.

**Lemma 1.4 (Weak König’s lemma).** Let  $T \subseteq 2^{<\mathbb{N}}$  be an infinite tree, that is, such that  $|T| = \infty$ . Then,  $[T]$  is non-empty. ★

PROOF. We construct a path  $X$  by induction on  $n$ . As  $T$  is infinite, by the pigeonhole principle there exists  $i \in \{0, 1\}$  and an infinity of nodes  $\sigma \in T$  which extend  $i$  (ie with  $i \prec \sigma$ ). We define  $X(0) = i$ . Suppose that  $\tau = X(0)X(1) \dots X(n)$  is defined with  $\tau \in T$  and such that there is an infinity of nodes  $\sigma \in T$  for which  $\tau \preceq \sigma$ . By the pigeonhole principle, there exists  $i \in \{0, 1\}$  and an infinity of nodes  $\sigma \in T$  which extend  $\tau i$ . We define  $X(n+1) = i$ .

By induction on  $n$ , we thus define a set  $X$  such that  $X \upharpoonright_n \in T$  for all  $n$ . ■

König’s lemma may seem trivial at first glance. Readers with a keen intuition and comfortable with the manipulation of infinite objects may have wondered whether there really was a need to tidy up this statement in a lemma. We will see that despite appearances, this lemma is not as trivial as it seems. Although simple, it constitutes a central tool, particularly interesting for its computational content.

## 1.1. Computable trees

A tree  $T \subseteq 2^{<\mathbb{N}}$  is computable if  $T$  is computable as a set, in other words, if there is a procedure to decide whether a node belongs to the tree or not. In this chapter, we will try to answer the following question.

**Question 1.5.** Given an infinite computable tree, what is the computational power required to compute an infinite path of  $T$ ? ★

The proof of König's lemma provides a clear construction: when we have computed a prefix  $\tau$  of our infinite path, we determine the next bit to be 0 if the tree contains an infinity of nodes which extend  $\tau 0$  and as being 1 otherwise. The problem is that it is not possible to know a priori in a computable way if an infinity of nodes extend  $\tau 0$ : it is a question for which the halting problem seems necessary. This leads us to define the notion of extendible node.

**Definition 1.6.** A node  $\sigma$  of a tree  $T \subseteq 2^{<\mathbb{N}}$  is *extendible* in  $T$  if the set  $\{\tau \in T : \tau \succeq \sigma\}$  is infinite.  $\diamond$

In other words, a node  $\sigma$  is extendible in a tree if the subtree of nodes compatible with  $\sigma$  is infinite. Let us remember the notation  $[\sigma]$  which denotes the class of sets  $X$  having  $\sigma$  as a prefix. By König's lemma, a node  $\sigma$  is extendible in  $T$  if and only if  $[\sigma] \cap [T] \neq \emptyset$ . Note that in any infinite tree, the root  $\epsilon$  is an extendible node, and that if  $\sigma$  is extendible, then so is at least one node among  $\sigma 0$  and  $\sigma 1$ . The following exercise shows that the extendible nodes are sufficient to describe the set of paths of a tree.

**Exercise 1.7.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a tree, and  $S$  the set of extendible nodes in  $T$ . Show that  $S$  is a tree, and that  $[T] = [S]$ .  $\diamond$

It follows from the definition of extendible node that leaves are not extendible. On the other hand, if the set of leaves of a computable tree is decidable, this is not generally the case for the set of extendible nodes.

**Exercise 1.8.** Let  $T \subseteq 2^{<\mathbb{N}}$  be an infinite computable tree containing only extendible nodes. Show that  $T$  contains a computable infinite path.  $\diamond$

In general, determining whether a computable set is infinite or not requires the oracle  $\emptyset''$ . In the case of trees, we can exploit the closure under prefix to reduce the complexity of the oracle to  $\emptyset'$ . The notation  $2^n$  of the following proposition denotes the set of strings of size  $n$ .

**Proposition 1.9.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable tree. The set of its extendible nodes is  $\Pi_1^0$ .  $\star$

PROOF. The trees being downward-closed,  $\{\tau \in T : \tau \succeq \sigma\}$  is infinite iff  $\forall n > |\sigma| \exists \tau \in 2^n$  such that  $\tau \succeq \sigma$  and  $\tau \in T$ , which is a  $\Pi_1^0$  predicate. Thus, the set of extendible nodes of  $T$  is the following  $\Pi_1^0$  set.

$$\{\sigma \in T : \forall n > |\sigma| \exists \tau \in 2^n \text{ such that } \tau \succeq \sigma \text{ and } \tau \in T\} \quad \blacksquare$$

By complementation, the set of non-extendible nodes of a computable tree is  $\Sigma_1^0$ , which means that if a node is non-extendible, we will end up realizing

it after a finite time. We will see how to use the notion of extendible node to create infinite computable trees having no computable path. In the construction of a computable tree, we must be able to decide in a finite time whether a node belongs to it or not. On the other hand, it is possible to defer the decision to make a node extendible or not, by default adding descendants over time to it, until deciding at one point  $t$  to stop adding more to it in order to make it non-extendible. This technique known as “time trick” allows to show the following result.

**Proposition 1.10.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a  $\Pi_1^0$  tree. There exists a computable tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $[T] = [S]$ . ★

PROOF. Let  $(T_n)_{n \in \mathbb{N}}$  be a  $\Pi_1^0$  approximation of  $T$ , that is to say a uniformly computable sequence of sets decreasing by the inclusion relation (with  $T_{n+1} \subseteq T_n$ ) such that  $\bigcap_n T_n = T$ . Let  $S = \{\sigma \in 2^{<\mathbb{N}} : \forall \tau \preceq \sigma \ \tau \in T_{|\sigma|}\}$ . The set  $S$  is computable.

Let us show that  $S$  is closed under prefix. Let  $\sigma \in S$  and  $\rho \preceq \sigma$ . By definition of  $S$ ,  $\forall \tau \preceq \sigma \ \tau \in T_{|\sigma|}$ . As  $|\rho| \leq |\sigma|$ ,  $T_{|\rho|} \supseteq T_{|\sigma|}$ , so  $\forall \tau \preceq \sigma \ \tau \in T_{|\rho|}$ . In particular, for all  $\tau \preceq \rho$ , also  $\tau \preceq \sigma$ , so  $\tau \in T_{|\rho|}$ . Thus, by definition of  $S$ ,  $\rho \in S$ .

Let us now show that  $[S] = [T]$ . We have  $P \in [S]$  iff  $\forall \sigma \prec P \ \sigma \in S$  iff  $\forall \sigma \prec P \ \forall \tau \preceq \sigma \ \tau \in T_{|\sigma|}$  iff  $\forall \tau \prec P \ \forall n \geq |\tau| \ \tau \in T_n$  iff  $\forall \sigma \prec P \ \sigma \in T$  iff  $P \in [T]$ . ■

This chapter mainly concerns the study of the set classes corresponding to paths through a computable tree. We will see with Proposition 3.5 that there are infinite computable trees that do not contain any infinite computable path. Then, we will determine the exact computational power that is needed to compute a path in any infinite computable tree. This study constitutes one of the basic building blocks of Reverse Mathematics, which we will see in Part III.

## 2. Topology on Cantor space

Topology is a branch of mathematics which abstractly formalizes the notions of limit and continuity, and which by extension studies the properties of geometric objects invariant by continuous deformation. The reader who has never studied this branch can be reassured: for the development of the chapters to come, we only need very basic elements of this theory, which we present here.

**Notation**

We will denote by  $i^\infty$  the infinite sequence which repeats the bit  $i \in \{0, 1\}$ .

## 2.1. Open and closed classes

As we have already mentioned, Cantor space  $2^\mathbb{N}$  is similar to the class of reals of the interval  $[0, 1]$ . An element  $X \in 2^\mathbb{N}$  can also be seen as the binary expansion of the real  $0.X(0)X(1)X(2)\dots$ , with however a subtle difference: the elements  $\sigma 10^\infty$  and  $\sigma 01^\infty$  are two distinct elements of  $2^\mathbb{N}$  but correspond to the same real number. This difference aside, we can see  $2^\mathbb{N}$  as an interval, and the simplest subsets of  $2^\mathbb{N}$  will simply be intervals of the form  $[\sigma]$ , which we will also call *cylinder*:

**Notation**

Given a string  $\sigma \in 2^{<\mathbb{N}}$  we write  $[\sigma]$  for the set  $\{X \in 2^\mathbb{N} : X \succeq \sigma\}$ . We will call *cylinder* a class of the form  $[\sigma]$ .

Given a string  $\sigma \in 2^{<\mathbb{N}}$ , we can see  $[\sigma]$  as an interval of infinite binary sequences: those which are lexicographically between  $\sigma 0^\infty$  and  $\sigma 1^\infty$ . With this in mind, the so-called *open* classes of Cantor are simply any union of intervals.

**Definition 2.1.** The *open* classes of Cantor space are arbitrary unions of cylinders, i.e., sets of the form  $\bigcup_{\sigma \in W} [\sigma]$  for a set  $W \subseteq 2^{<\mathbb{N}}$ . The *closed* classes are the complements of the open classes.  $\diamond$

The reader not familiar to these concepts can demonstrate, to get his hands on the definitions, that the finite unions of cylinders are both open and closed.

**Notation**

Given a set  $W \subseteq 2^{<\mathbb{N}}$  we will write  $[W]$  to denote its corresponding open set, i.e., the class  $\bigcup_{\sigma \in W} [\sigma]$ .

It is clear from the definition that the open classes are closed under arbitrary union and therefore, by passage to the complement, that the closed are closed under any intersection. We introduce an element of vocabulary that will often come up in the manipulation of open and closed classes:

### Notation

Given a countable union of classes  $\bigcup_n \mathcal{B}_n$ , we will say that the union is *increasing* if  $\mathcal{B}_n \subseteq \mathcal{B}_{n+1}$  for all  $n$ . In the same way we will say that an intersection  $\bigcap_n \mathcal{B}_n$  is *decreasing* if  $\mathcal{B}_{n+1} \subseteq \mathcal{B}_n$  for all  $n$ .

The mental representation of an open class should be fairly clear to the reader: the unions of simple bricks that are the cylinders. Here is an illustrative example.

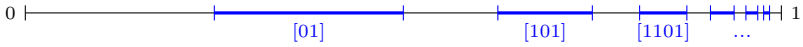


Figure 2.2: Illustration of the open class  $[01] \cup [101] \cup [1101] \cup [11101] \cup \dots$

We now show that we can consider without loss of generality that the intersections of open classes are decreasing and the unions of closed classes increasing (in particular because  $\bigcap_n \mathcal{U}_n = \bigcap_n (\bigcap_{m \leq n} \mathcal{U}_m)$ ):

**Proposition 2.3.** A finite intersection of open classes is an open. By passing to the complement a finite union of closed classes is closed. ★

PROOF. Let  $\mathcal{U}_0, \mathcal{U}_1 \subseteq 2^{\mathbb{N}}$  be two open classes. A set  $X$  belongs to  $\mathcal{U}_0 \cap \mathcal{U}_1$  iff it belongs to a cylinder  $[\sigma_0] \subseteq \mathcal{U}_0$  as well as to a cylinder  $[\sigma_1] \subseteq \mathcal{U}_1$ . So  $\mathcal{U}_0 \cap \mathcal{U}_1 = \bigcup_{[\sigma_0] \subseteq \mathcal{U}_0, [\sigma_1] \subseteq \mathcal{U}_1} [\sigma_0] \cap [\sigma_1]$ . ■

Closed sets are more difficult to describe. This is not necessarily surprising. By way of analogy, let's say that you can get to know your neighborhood well, without having a precise idea of the rest of the world. The awareness of this complexity and the way of apprehending it already figure in the work of Cantor, for example through the famous Cantor-Bendixson theorem. However, there is a simple way to represent the closed in  $2^{\mathbb{N}}$  as the class of all infinite paths of a tree.

**Proposition 2.4.** A class  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  is closed iff there is a tree  $T \subseteq 2^{<\mathbb{N}}$  such that  $\mathcal{P} = [T]$ . ★

PROOF. Let  $\mathcal{P}$  be a closed class. Let  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$  be its complement with  $W \subseteq 2^{<\mathbb{N}}$ . We define the tree  $T \subseteq 2^{<\mathbb{N}}$  as being the set of strings  $\sigma$  having no prefix in  $W$ . By definition  $T$  is closed under prefix and is therefore a tree. Let us show  $[T] = \mathcal{P}$ . We have  $X \in [T]$  iff no prefix  $\sigma \prec X$  is in  $W$  iff  $X \notin \bigcup_{\sigma \in W} [\sigma]$  iff  $X \in \mathcal{P}$ . So  $[T] = \mathcal{P}$ .

Conversely, if  $T \subseteq 2^{<\mathbb{N}}$  is a tree, the class  $[T]$  of its paths is closed, because it is the complement of the class  $\bigcup_{\sigma \notin T} [\sigma]$ . ■

By way of example, the reader can consult Figure 2.5, representing the tree corresponding to the complement of the open class described in Figure 2.2.

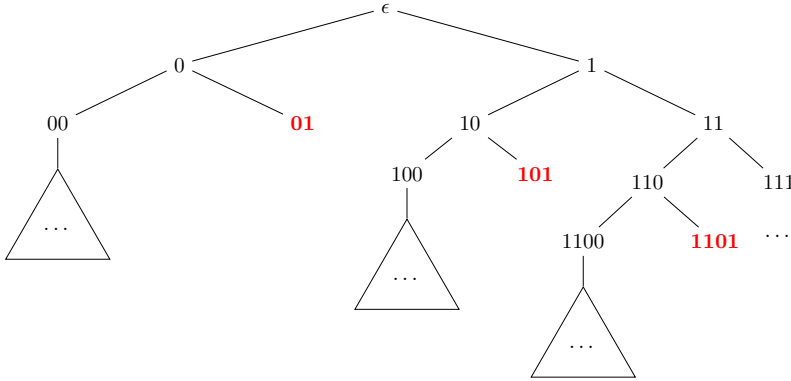


Figure 2.5: Illustration of the tree representing the complement of the open class  $[01] \cup [101] \cup [1101] \cup [11101] \cup \dots$ . The bold nodes correspond to the cylinders constituting the basic bricks of the open class. The triangles represent a “full” subtree, starting from the node where they are.

## 2.2. Compactness

*Compactness* is a fundamental notion of topology. It is generally defined via the Borel-Lebesgue property, which in Cantor space is formulated as follows:

**Definition 2.6.** A class  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  has the *Borel-Lebesgue property*<sup>a</sup> if for any collection of open classes  $(\mathcal{O}_n)_{n \in \mathbb{N}}$  such that  $\mathcal{P} \subseteq \bigcup_n \mathcal{O}_n$ , there exists a finite set  $F \subseteq \mathbb{N}$  such that  $\mathcal{P} \subseteq \bigcup_{n \in F} \mathcal{O}_n$ . We will say that a class possessing the Borel-Lebesgue property is *compact*.  $\diamond$

<sup>a</sup>Also called “Heine-Borel property”.

We show with the following proposition that in Cantor space, the compact classes are exactly the closed ones, and the reader will be able to note by reading the proof, that weak König’s lemma can be seen as a reformulation of the fact that closed classes are compact.

**Proposition 2.7.** A class of  $2^{\mathbb{N}}$  is closed iff it has the Borel-Lebesgue property.  $\star$

PROOF. Let  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  be closed and let  $(\mathcal{O}_n)_{n \in \mathbb{N}}$  be a collection of open classes such that  $\mathcal{P} \subseteq \bigcup_n \mathcal{O}_n$ . Let  $T \subseteq 2^{<\mathbb{N}}$  be a tree such that  $[T] = \mathcal{P}$ . Let  $S \subseteq 2^{<\mathbb{N}}$  be the tree of strings  $\sigma \in T$  such that  $[\sigma] \not\subseteq \bigcup_{n < |\sigma|} \mathcal{O}_n$ . If

the tree  $S$  is finite, then there exists a length  $\ell$  such that for all  $\sigma \in T$  such that  $|\sigma| = \ell$ ,  $[\sigma] \subseteq \bigcup_{n < \ell} \mathcal{O}_n$ . It follows that  $[T] \subseteq \bigcup_{\sigma \in T, |\sigma| = \ell} [\sigma] \subseteq \bigcup_{n < \ell} \mathcal{O}_n$ . If  $S$  is infinite, by weak König's lemma,  $[S] \neq \emptyset$ . Let  $P \in [S]$ . Let us show that  $P \notin \bigcup_n \mathcal{O}_n$  to deduce a contradiction, because  $[S] \subseteq [T] \subseteq \bigcup_n \mathcal{O}_n$ .  $i \in \mathbb{N}$ . By definition of  $[S]$ , for all  $\ell$ ,  $P \upharpoonright_\ell \in S$ , so by definition of  $S$  we have  $[P \upharpoonright_\ell] \not\subseteq \bigcup_{n < \ell} \mathcal{O}_n$ . In particular, for all  $\ell > i$ ,  $[P \upharpoonright_\ell] \not\subseteq \mathcal{O}_i$ . Since  $\mathcal{O}_i$  is open, it follows that  $P \notin \mathcal{O}_i$ .

Suppose now that a class  $\mathcal{B}$  admits the Borel-Lebesgue property. For any  $X \notin \mathcal{B}$ , let  $\mathcal{O}_X$  be the open class corresponding to the complement of the class  $\{X\}$  (it is the union of the cylinders  $[\sigma i]$  for any string  $\sigma$  and all  $i$  such that  $X(\lvert\sigma\rvert) \neq i$ ). In particular we have  $\mathcal{B} = \bigcap_{X \notin \mathcal{B}} \mathcal{O}_X$ . Note that each  $\mathcal{O}_X$  is a union of cylinders and that each cylinder is open. So by the Borel-Lebesgue property we can find for any  $X$  a finite set of cylinders  $F_X$  such that  $\mathcal{B} \subseteq \bigcup_{\sigma \in F_X} [\sigma] \subseteq \mathcal{O}_X$ . In particular  $\mathcal{B} = \bigcap_{X \notin \mathcal{B}} \bigcup_{\sigma \in F_X} [\sigma]$ . Each union  $\bigcup_{\sigma \in F_X} [\sigma]$  is a closed class as a finite union of cylinders. As an arbitrary intersection of closed classes is closed, we deduce that  $\mathcal{B}$  is a closed class. ■

In practice, we will use the following consequence of compactness: any countable and decreasing intersection of non-empty closed classes is non-empty, which we prove here.

**Proposition 2.8.** Let  $\mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \dots$  be a decreasing sequence of non-empty closed classes. Then,  $\bigcap_n \mathcal{P}_n$  is non-empty. ★

PROOF. Let  $T_n$  be a tree such that  $[T_n] = \mathcal{P}_n$ . We can assume without loss of generality  $T_{n+1} \subseteq T_n$ . Let us show that  $\bigcap_n [T_n] = [\bigcap_n T_n]$ . We have  $X \in \bigcap_n [T_n]$  iff  $X \upharpoonright_m \in T_n$  for all  $m, n$  iff  $X \in [\bigcap_n T_n]$ . So  $\bigcap_n [T_n] = [\bigcap_n T_n]$ . Let  $T = \bigcap_n T_n$ . In particular  $[T] = \bigcap_n \mathcal{P}_n$ .

Suppose absurdly that  $[T] = \bigcap_n \mathcal{P}_n$  is empty. According to König's lemma there is therefore an integer  $a$  such that no string  $\sigma$  of size greater than or equal to  $a$  is in  $T$ . As the strings of size  $a$  are in finite quantity and the sequence  $(T_n)_{n \in \mathbb{N}}$  is decreasing by inclusion, there must therefore be integer  $m$  such that none of these strings belong to  $T_m$ . So  $[T_m]$  is empty, which contradicts the assumptions. ■

### 2.3. Continuity

Let's tackle another topological notion that we will mention here and there in the chapters to come. Continuity, another central notion of topology, unexpectedly hides in Computability Theory under the following idea.

A Turing functional  $\Phi$  can be seen as a partial function from  $2^{\mathbb{N}}$  to  $2^{\mathbb{N}}$ , whose input is an oracle  $X$ , and the result, which we denote by  $\Phi^X$  or  $\Phi(X)$ , is the set  $Y$  such that  $\Phi^X(n) \downarrow = Y(n)$ . This function from  $2^{\mathbb{N}}$  to  $2^{\mathbb{N}}$  is of course only defined for oracles  $X$  such that  $\forall n \Phi^X(n) \downarrow \in \{0, 1\}$ .

We have seen that when  $\Phi^X(n) \downarrow = v$ , by the use property (see Definition 4-4.2), only a finite initial segment  $\sigma$  of the oracle  $X$  is used. More generally, if  $\Phi^X \succeq \tau$  (which means  $\forall n < |\tau| \Phi^X(n) \downarrow = \tau(n)$ ), only a finite part  $\sigma$  of the oracle is used to realize this. It follows that for all  $X \in [\sigma]$ ,  $\Phi^X \succeq \tau$ , and therefore  $\{\Phi^X : X \in [\sigma]\} \subseteq [\tau]$ .

The reader having followed an introductory course in Topology will recognize in this idea the notion of continuity: for any open space of the image space as “small” as one wants — in practice a cylinder  $[\tau]$  — there is a “small” enough open class in the domain space — in practice a cylinder  $[\sigma]$  — such that every  $X \in [\sigma]$  is sent to  $[\tau]$ : concretely the string  $\sigma$  is “sent” to the string  $\tau$ .

**Definition 2.9.** A (possibly partial) function  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  is *continuous* in  $X \in \text{dom } f$  if for any cylinder  $[\tau]$  containing  $f(X)$ , there exists a cylinder  $[\sigma]$  containing  $X$  such that  $f([\sigma]) \subseteq [\tau]$ . We will say that a function is *continuous* if it is continuous in  $X$  for all  $X \in \text{dom } f$ .  $\diamond$

In general, we will consider continuous functions over their entire domain of definition, which then admit an equivalent characterization in terms of open pre-image:

**Proposition 2.10.** A (possibly partial) function  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  is continuous iff for any open class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$ , there exists an open class  $\mathcal{V} \subseteq 2^{\mathbb{N}}$  such that  $f^{-1}(\mathcal{U}) = \mathcal{V} \cap \text{dom } f$ . If the function is total we then have  $f^{-1}(\mathcal{U})$  open for any open class  $\mathcal{U}$ .  $\star$

**PROOF.** Let  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  be a continuous function and  $\mathcal{U} \subseteq 2^{\mathbb{N}}$  an open class. As  $\mathcal{U}$  is open, for all  $X \in f^{-1}(\mathcal{U})$  there exists a cylinder  $[\tau_X]$  containing  $f(X)$  such that  $[\tau_X] \subseteq \mathcal{U}$ . By continuity of  $f$ , for all  $X$ , there exists a cylinder  $[\sigma_X]$  containing  $X$  such that  $f([\sigma_X]) \subseteq [\tau_X]$ . Then,  $\text{dom } f \cap \bigcup_{X \in f^{-1}(\mathcal{U})} [\sigma_X] = f^{-1}(\mathcal{U})$ .

Conversely, suppose that for any open class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$ , there exists an open class  $\mathcal{V}$  such that  $\text{dom } f \cap \mathcal{V} = f^{-1}(\mathcal{U})$ . Let  $Y \in \text{Im } f$  and  $[\tau]$  be a cylinder containing  $Y$ . Let  $\mathcal{V}$  be an open class such that  $\text{dom } f \cap \mathcal{V} = f^{-1}([\tau])$ . Since  $\mathcal{V}$  is open, there is  $W \subseteq 2^{<\mathbb{N}}$  such that  $\bigcup_{\sigma \in W} [\sigma] = \mathcal{V}$ . Note that  $W \neq \emptyset$  because  $Y \in \text{Im } f \cap [\tau]$ , so there exists a string  $\sigma \in W$ . We have  $f([\sigma]) = f(\text{dom } f \cap [\sigma]) \subseteq [\tau]$ .  $\blacksquare$

A computable functional  $\Phi$  is therefore always also a continuous function on its domain of definition, i.e., on the space of  $X$  such that  $\Phi(X, n) \downarrow \in \{0, 1\}$  for all  $n$ . On the other hand, a continuous function has a priori no reason to be computable: it is possible that any finite piece of the output of the function can be determined by a finite piece of the input, but that this “determinism” is not computable. As an example let us consider the function  $\Phi$  which on any set  $X$  associates  $X \oplus \emptyset'$ . Such a function  $\Phi$  is continuous, but not computable. On the other hand, it can be computed with the help of  $\emptyset'$ .

In Computability Theory, the non-continuous function par excellence is that which to  $X$  associates  $X'$ : indeed to know whether  $n \in X'$  it is necessary to know if  $\Phi_n(X, n) \downarrow$  and for that potentially to know an infinity of bits of  $X$  (especially if  $\Phi_n(X, n) \uparrow$ ). We will see actual versions of some well-known theorems of analysis, which state that any function that is not continuous, but not too complex, for example  $X \mapsto X'$ , is nevertheless continuous over a “large” set of points, in particular on a co-meager class (see Theorem 10-3.20) and on a class of arbitrarily large measure (see Theorem 19-3.8).

## 2.4. Perfect classes

A last topological notion that we will use is that of perfect classes. These are the classes which are the image of a continuous injection from  $2^{\mathbb{N}}$  to  $2^{\mathbb{N}}$ , that is to say exactly the classes of the form  $[T]$  for an f-tree  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  (see Section 7-5). These classes are therefore always closed and can be represented by a tree. By extension we will therefore also speak of a perfect tree.

**Definition 2.11.** A non-empty tree  $T \subseteq 2^{<\mathbb{N}}$  is perfect if any node in  $T$  has two incompatible extensions in  $T$ . ◇

We have seen with Exercize 1.7 that given a closed class  $\mathcal{F}$  represented by a tree  $T$ , we can consider without loss of generality — if we do not deal with the effectiveness — that  $T$  contains only extendible nodes. On the other hand, an extendible node does not necessarily have two incompatible extensions in the general case. When this happens, it means that there is exactly one infinite path passing through this node. We call such paths *isolated points*. For an arbitrary class  $\mathcal{A}$  the corresponding definition is as follows.

**Definition 2.12.** Let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$ . An element  $X \in \mathcal{A}$  is a *isolated point* if there is a  $\sigma \prec X$  prefix such that  $[\sigma] \cap \mathcal{A} = \{X\}$ . ◇

The usual definition of perfect class then follows from that of isolated point:

**Definition 2.13.** A non-empty class  $\mathcal{F} \subseteq 2^{\mathbb{N}}$  is *perfect* if it is closed and has no isolated point. Equivalently  $\mathcal{F} = [T]$  for a perfect tree  $T \subseteq 2^{<\mathbb{N}}$ .  $\diamond$

Perfect classes are of great importance, in particular because they allow the construction of cardinality arguments: any perfect class is by definition in continuous bijection with  $2^{\mathbb{N}}$ , and therefore has the same cardinality as  $2^{\mathbb{N}}$ . Moreover if a class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  contains a perfect class, then we have an injection of  $2^{\mathbb{N}}$  into  $\mathcal{A}$ . Identity injection of  $\mathcal{A}$  into  $2^{\mathbb{N}}$  then gives us  $|\mathcal{A}| = |2^{\mathbb{N}}|$ . It is in fact roughly speaking *the only way* to show that a class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  has the power of continuum. We will talk about it again in Section 9-4 as well as in Section 30-4. Here is a simple application of the notion of perfect class to the study of cardinality.

**Proposition 2.14.** Any non-empty countable closed class  $\mathcal{F}$  has isolated points. We can inject  $2^{\mathbb{N}}$  in any non-empty closed class with no isolated point.  $\star$

PROOF. Suppose that  $\mathcal{F}$  does not contain any isolated point. Let  $\mathcal{F} = [T]$  for a tree  $T$  having only extendible nodes. Since  $\mathcal{F}$  does not contain an isolated point then all nodes of  $T$  have two incompatible extensions. We then have an injection of  $2^{\mathbb{N}}$  into  $\mathcal{F}$ . If a closed class  $\mathcal{F}$  is countable, we cannot inject  $2^{\mathbb{N}}$  into  $\mathcal{F}$  and therefore by contraposition, it contains isolated points.  $\blacksquare$

**Corollary 2.15 (Cantor)**

*The continuum hypothesis is true for closed classes of  $2^{\mathbb{N}}$ : they are either countable, or of cardinality  $|2^{\mathbb{N}}|$ .*

PROOF. Let  $\mathcal{F}$  be closed. Let  $W \subseteq 2^{<\mathbb{N}}$  be the set of strings  $\sigma$  such that  $\mathcal{F} \cap [\sigma]$  is countable. Let  $\mathcal{F}' = \mathcal{F} \setminus \bigcup_{\sigma \in W} [\sigma]$ . Note that  $\mathcal{F}' \subseteq \mathcal{F}$  is always a closed class. If  $\mathcal{F}'$  is empty, then according to König's lemma (or compactness) it is only necessary to remove from  $\mathcal{F}$  a finite number of strings  $\sigma_0, \dots, \sigma_n \in W$  so that  $\mathcal{F}' = \mathcal{F} \setminus [\sigma_0] \cup \dots \cup [\sigma_n]$  is empty. As each class  $\mathcal{F} \cap [\sigma_i]$  is countable, we have emptied  $\mathcal{F}$  by removing a countable quantity of points. So  $\mathcal{F}$  is countable. Otherwise  $\mathcal{F}'$  is not empty, and in addition to that it cannot contain isolated points (because if  $\mathcal{F}' \cap [\sigma]$  contains only one element then  $\mathcal{F} \cap [\sigma]$  is countable). By Proposition 2.14 we have an injection from  $2^{\mathbb{N}}$  to  $\mathcal{F}'$ .  $\blacksquare$

This result will be extended with Corollary 30-3.3.

### 3. $\Pi_1^0$ classes

We are now interested in the *effective* version of the open and closed classes. We often use the term *effective* rather than *computable* for this kind of objects, because they are not necessarily computable in the sense that we can precisely know the smallest details, but they still admit a certain tangible, effective description provided by an algorithm.

**Definition 3.1.** A class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$  is called  $\Sigma_1^0$  if there exists a c.e. set  $W \subseteq 2^{<\mathbb{N}}$  such that  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$ . A class  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  is said to be  $\Pi_1^0$  if its complement is a  $\Sigma_1^0$  class.  $\diamond$

The  $\Sigma_1^0$  and  $\Pi_1^0$  are respectively the *effectively* open and closed classes of Cantor space. We will call *code* of a  $\Sigma_1^0$  class  $\mathcal{U}$  an integer  $e$  such that  $\mathcal{U} = \bigcup_{\sigma \in W_e} [\sigma]$ . Likewise, a *code* of a  $\Pi_1^0$  class  $\mathcal{P}$  is a code of the  $\Sigma_1^0$  class  $\mathcal{U} = 2^{\mathbb{N}} \setminus \mathcal{P}$ . This allows us to speak of uniform computability on the sequence of  $\Sigma_1^0$  or  $\Pi_1^0$  classes by considering the computability of their sequence of codes. The  $\Pi_1^0$  classes are an important and well-studied notion in Computability Theory.

#### Remark

It is important to distinguish well the  $\Sigma_1^0$  or  $\Pi_1^0$  sets of integers which are the first levels of the arithmetic hierarchy (see Chapter 5) from the  $\Sigma_1^0$  and  $\Pi_1^0$  classes which are the respective effectively open and closed classes of Cantor space. However, there are links between these concepts.

Let's start with the fact that Proposition 2.3 which states that a finite intersection of open classes is an open one, also works with the  $\Sigma_1^0$  classes:

**Proposition 3.2.**  $\Sigma_1^0$  classes are closed under finite intersection. By passing to the complement, the  $\Pi_1^0$  classes are closed under finite union.  $\star$

PROOF. Let  $\mathcal{U}_0 = \bigcup_{\sigma \in W_0} [\sigma]$  and  $\mathcal{U}_1 = \bigcup_{\sigma \in W_1} [\sigma]$  be two  $\Sigma_1^0$  classes. Then, the open class  $\mathcal{U}_0 \cap \mathcal{U}_1$  is described by the c.e. set which lists the longest string among  $\sigma_0, \sigma_1$  for any  $\sigma_0 \in W_0$  and  $\sigma_1 \in W_1$  such that  $\sigma_0 \preceq \sigma_1$  or such that  $\sigma_1 \preceq \sigma_0$ . The string thus enumerated corresponds to  $[\sigma_0] \cap [\sigma_1]$ . ■

The first step to better understand the nature of the  $\Pi_1^0$  classes is undoubtedly to prove the effective version of Proposition 2.4: the  $\Pi_1^0$  are exactly the infinite paths of computable trees.

**Proposition 3.3.** A class  $\mathcal{P}$  is  $\Pi_1^0$  iff there is a computable tree  $T \subseteq 2^{<\mathbb{N}}$  such that  $[T] = \mathcal{P}$ .  $\star$

PROOF. Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable tree. The class  $[T] = \{X : \forall n \ X \upharpoonright_n \in T\}$  is  $\Pi_1^0$ . Indeed its complement is the  $\Sigma_1^0$  class described by the union of cylinders  $[\sigma]$  such that  $\sigma \notin T$ .

Suppose that  $\mathcal{P}$  is  $\Pi_1^0$ . Let  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$  be its complement. We compute the following tree  $T \subseteq 2^{<\mathbb{N}}$ : in the computation step  $t$ , for any string  $\sigma \in 2^{<\mathbb{N}}$  of size  $t$ , we decide  $\sigma \in T$  iff for any prefix  $\tau \preceq \sigma$  we have  $\tau \notin W[t]$ .

It is clear that  $T$  is closed under prefix: if  $\sigma$  of size  $t$  is in  $T$  then no prefix of  $\sigma$  is in  $W$  at the computation step  $t$ , so for  $s \leq t$ , also no prefix of  $\sigma \upharpoonright_s$  is in  $W$  at the computation step  $s$ . Now if  $X \in \mathcal{P}$  then no  $\sigma$  prefix of  $X$  is in  $W$  and therefore each of those prefixes will be in  $T$ . Conversely if  $X \notin \mathcal{P}$  then a prefix  $\sigma$  of  $X$  goes into  $W$  at a certain stage  $t$ . By construction no strings  $\tau \succeq \sigma$  larger than  $t$  will be in  $T$ . So  $\mathcal{P} = [T]$ . ■

A code of a computable tree  $T \subseteq 2^{<\mathbb{N}}$  is an integer  $e$  such that  $\Phi_e = T$ . Note that the proof of Proposition 3.3 is uniform, and allows to pass computably from a code of a  $\Pi_1^0$  class to a code of the corresponding tree, and vice versa. We can therefore consider without distinction the code of  $\Pi_1^0$  classes and computable trees in the proofs to come. The following proposition establishes a link with the  $\Sigma_1^0$  and  $\Pi_1^0$  classes and the arithmetic hierarchy.

**Proposition 3.4.** Let  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  be a class.

- (1)  $\mathcal{P}$  is  $\Sigma_1^0$  iff  $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \exists n \ R(X \upharpoonright_n)\}$  for a computable predicate  $R \subseteq 2^{<\mathbb{N}}$
- (2)  $\mathcal{P}$  is  $\Pi_1^0$  iff  $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall n \ R(X \upharpoonright_n)\}$  for a computable predicate  $R \subseteq 2^{<\mathbb{N}}$  ★

PROOF. For (2), given a  $\Pi_1^0$  class  $\mathcal{P}$ , it suffices to consider the computable tree  $T$  such that  $[T] = \mathcal{P}$ . The computable predicate is simply  $T$ . Conversely if  $\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall n \ R(X \upharpoonright_n)\}$  then the computable tree given by  $\sigma \in T$  iff  $\forall \tau \preceq \sigma \ R(\tau)$  is such that  $[T] = \mathcal{P}$ .

We obtain (1) by passing to the complement. ■

Let us now see some generalities, first of all the proof that König's lemma does not belong to computable mathematics: some non-empty  $\Pi_1^0$  classes — and therefore some infinite computable trees — do not contain any computable point. We will see throughout the following chapters many examples of  $\Pi_1^0$  classes not containing any computable point. We anticipate in particular for the following proposition on the simplest example to define: the class of  $\text{DNC}_2$  sets of Proposition 6.4.

**Proposition 3.5.** There are non-empty  $\Pi_1^0$  classes that do not contain any computable set. ★

PROOF. We define the class

$$\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall e \forall t \Phi_e(e)[t] \uparrow \vee \Phi_e(e)[t] \downarrow \neq X(e)\}.$$

The class  $\mathcal{P}$  contains all the sets  $X$  such that  $X(n)$  can take any value if  $\Phi_n(n) \uparrow$ , and which are always different from  $\Phi_n(n)$  if  $\Phi_n(n) \downarrow$ . It is clear that this class is not empty (the halting problem for example easily computes an element of  $\mathcal{P}$ ). By Proposition 3.4,  $\mathcal{P}$  is a  $\Pi_1^0$  class. Moreover, the class  $\mathcal{P}$  does not contain any computable set: if  $X$  is computable then there must be some  $e$  such that  $\Phi_e(e) \downarrow = X(e)$ . ■

Let us now continue on another key property of  $\Pi_1^0$  classes: the non-empty  $\Pi_1^0$  classes containing no computable points are necessarily uncountable. They cannot contain *isolated points*, that is to say sets  $X$  such that for a certain  $n$ , no other set than  $X$  and extending  $X \upharpoonright_n$  does not belong to  $\Pi_1^0$ .

**Proposition 3.6.** Let  $\mathcal{P}$  be a  $\Pi_1^0$  class containing exactly one  $X$  element. Then,  $X$  is computable. ★

PROOF. Let  $T \subseteq 2^{<\mathbb{N}}$  be the computable tree such that  $[T] = \mathcal{P}$ . Consider the following algorithm: search for the smallest  $t$  such that either for any string  $\sigma \succeq 0$  of size  $t$  we have  $\sigma \notin T$ , or for any string  $\sigma \succeq 1$  of size  $t$  we have  $\sigma \notin T$ . Note that exactly one of the two events must necessarily happen: if both events happen, the class is empty. If neither happens there is an infinity of strings in  $T$  which extend 0 and also an infinity which extend 1. According to König's lemma  $T$  therefore contains at least two infinite paths: one which extends 0 and one which extends 1, which contradicts the hypotheses on  $\mathcal{P}$ .

Once one of the two events has arrived, we therefore know whether  $X$  begins with 0 or 1. We can easily see how to continue by induction: once  $X \upharpoonright_n$  has been computed, we look for the smallest  $t$  such that for any string  $\sigma \succeq X \upharpoonright_n 0$  of size  $t$  we have  $\sigma \notin T$ , or for any string  $\sigma \succeq X \upharpoonright_n 1$  of size  $t$  we have  $\sigma \notin T$ . Once one of the two events has occurred, the value of  $X(n)$  is known. ■

### Corollary 3.7

*Isolated points of any  $\Pi_1^0$  class are computable.*

PROOF. Let  $\mathcal{P}$  be a  $\Pi_1^0$  class and  $X \in \mathcal{P}$  an isolated point. By definition, there is a prefix  $\sigma \prec X$  such that  $[\sigma] \cap \mathcal{P} = \{X\}$ . In particular  $[\sigma] \cap$

$\mathcal{P}$  is a  $\Pi_1^0$  class containing exactly one element, this element is therefore computable. ■

### Corollary 3.8

*Any countable  $\Pi_1^0$  class contains a computable set.*

PROOF. By Proposition 2.14 and Corollary 3.7. ■

## 4. Basis theorems

Members of a  $\Pi_1^0$  class can be of very different Turing degrees. For example, Cantor space  $2^{\mathbb{N}}$  is a  $\Pi_1^0$  class containing sets of each Turing degree. Given a non-empty  $\Pi_1^0$  class, we are mainly interested in the degree of difficulty of computing one of its members.

**Definition 4.1.** A *basis* for  $\Pi_1^0$  classes is a class of sets  $\mathcal{C}$  such that any non-empty  $\Pi_1^0$  class contains an element of  $\mathcal{C}$ . ◇

In this section, we will prove a number of theorems which, given a weakness property  $P$ , are of the form “Any non-empty  $\Pi_1^0$  class contains a member satisfying  $P$ .” These theorems are called “basis theorems”, because they state that the members of  $P$  form a basis for the  $\Pi_1^0$  classes. Conversely, the “anti-basis theorems” state the existence of a non-empty  $\Pi_1^0$  class containing no member satisfying a weakness property. The very first basis theorem is due to Kreisel [123], and is left as an exercise.

**Exercise 4.2. (★)** Show that any non-empty  $\Pi_1^0$  class contains a  $\emptyset'$ -computable element. ◇

We can do even better than Exercise 4.2 via the central theorem called “low basis theorem”, which states that any non-empty  $\Pi_1^0$  class contains a low set. This theorem has a fundamental importance in Computability Theory and in Reverse Mathematics, in particular to provide a number of examples and counter-examples.

### Theorem 4.3 (Jockusch et Soare [103])

*Any non-empty  $\Pi_1^0$  class contains a low set.*

PROOF. Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class. We are going to use  $\emptyset'$  to compute an element  $Z \in \mathcal{P}$ , while computing its Turing jump  $Z'$ . For that, let us define a uniformly  $\emptyset'$ -computable decreasing sequence of non-empty  $\Pi_1^0$

classes  $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$  as follows: Let  $\mathcal{P}_0 = \mathcal{P}$ . Suppose  $\mathcal{P}_n$  defined and consider the class

$$\mathcal{B}_n = \{X \in 2^{\mathbb{N}} : \forall t \Phi_n(X, n)[t] \uparrow\} \cap \mathcal{P}_n.$$

Note that  $\mathcal{B}_n$  is also a  $\Pi_1^0$  class, and that the code of a computable tree  $T_n$  such that  $\mathcal{B}_n = [T_n]$  is uniformly computable in  $n$ .

We ask  $\emptyset'$  the question whether  $\mathcal{B}_n$  is empty: according to König's lemma this is the case iff there exists  $m$  such that no string  $\sigma$  of size  $m$  belongs  $T_n$ , which is indeed a  $\Sigma_1^0$  event. If  $\emptyset'$  responds positively, we let  $Y(n) = 1$  and we define  $\mathcal{P}_{n+1} = \mathcal{P}_n$ . Note that in this case all the elements  $X \in \mathcal{P}_{n+1}$  are such that  $\Phi_n(X, n) \downarrow$ . In the opposite case we let  $Y(n) = 0$  and we define  $\mathcal{P}_{n+1} = \mathcal{B}_n$ . Note that in this case all the elements of  $X \in \mathcal{P}_{n+1}$  are such that  $\Phi_n(X, n) \uparrow$ .

For each  $n$ ,  $\mathcal{P}_n$  is a non-empty closed classe and therefore  $\bigcap_n \mathcal{P}_n$  is non-empty. By construction the element  $Y$  computed by  $\emptyset'$  corresponds to the Turing jump of any element of  $\bigcap_n \mathcal{P}_n$  (which happens to be a singleton). ■

We have already seen with Proposition 4-9.1 the existence of low and non-computable sets. We now have an alternative proof by combining Theorem 4.3 and Proposition 3.5.

We now tackle the second main basis theorem for  $\Pi_1^0$  classes: computably dominated sets. For this, we need a lemma which also has its own interest.

**Lemma 4.4.** Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class. Suppose that a functional  $\Phi$  is total on all the members of  $\mathcal{P}$ . Then, we can define uniformly in a code of  $\mathcal{P}$  a computable function  $g$  which dominates  $n \mapsto \Phi(X, n)$  for all  $X \in \mathcal{P}$ . ★

PROOF. Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable tree such that  $[T] = \mathcal{P}$ . Let us show that for any  $n$ , there exists  $t$  such that  $\Phi(\sigma, n)[|\sigma|] \downarrow$  for any string  $\sigma \in T$  of size  $t$ . Indeed, in the opposite case, there exists  $n$  such that the set  $\{\sigma \in T : \Phi(\sigma, n)[|\sigma|] \uparrow\}$  contains for all  $t$  a string of size  $t$  and is therefore an infinite subtree of  $T$ , which therefore contains by König's lemma an infinite path  $X$ . We thus have  $\forall t \Phi(X, n)[t] \uparrow$  which contradicts the totality of  $\Phi$  on all the oracles of  $[T]$ .

We can therefore compute the function  $g$  which for  $n$  searches for the smallest  $t$  such that  $\Phi(\sigma, n)[t] \downarrow = v_\sigma$  for any string  $\sigma \in T$  of size  $t$ . Once  $t$  is found we define  $g(n) = \sum_{|\sigma|=t} v_\sigma + 1$ . It is clear that  $g$  dominates all functions computable via  $\Phi$  by an oracle of  $[T]$ . ■

The following theorem is known as “computably dominated basis theorem”. With the existence of a non-empty  $\Pi_1^0$  class having no computable member, this theorem gives us an alternative proof of the existence of computably dominated sets that are non-computable.

**Theorem 4.5 (Jockusch et Soare [103])**

*Any non-empty  $\Pi_1^0$  class contains a computably dominated set.*

PROOF. Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class. We will define an infinite decreasing sequence of non-empty  $\Pi_1^0$  classes  $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$  such that  $\bigcap_n \mathcal{P}_n$  contains only computably dominated sets. Let  $\mathcal{P}_0 = \mathcal{P}$ . Suppose  $\mathcal{P}_n$  defined. Let  $\mathcal{B}_{n,m} = \{X : \Phi_n(X, m) \uparrow\}$ . Note that each class  $\mathcal{B}_{n,m}$  is  $\Pi_1^0$ . Suppose that there exists  $m$  such that  $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$ . Then, we define  $\mathcal{P}_{n+1} = \mathcal{P}_n \cap \mathcal{B}_{n,m}$ . Note that for all  $X \in \mathcal{P}_{n+1}$  we have  $\Phi_n(X, m) \uparrow$ . Suppose now that for all  $m$  we have  $\mathcal{P}_n \cap \mathcal{B}_{n,m} = \emptyset$ . This implies that the functional  $\Phi_n$  is total for all  $X \in \mathcal{P}_n$ . We then define  $\mathcal{P}_{n+1} = \mathcal{P}_n$ . According to Lemma 4.4, there exists a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  which dominates  $m \mapsto \Phi_n(X, m)$  for all  $X \in \mathcal{P}_{n+1}$ .

As a decreasing intersection of non-empty closed classes, the class  $\bigcap_n \mathcal{P}_n$  is non-empty. Let  $X \in \bigcap_n \mathcal{P}_n$ . By construction, for all  $n$ , if  $\Phi_n$  is total on the oracle  $X$  then  $m \mapsto \Phi_n(X, m)$  is bounded by a computable function. So  $X$  is computably dominated. ■

We now see a last basis theorem called “cone avoidance”: given a set  $X$ , we call *upper cone* of  $X$  the class  $\mathcal{C}_X = \{Y \in 2^{\mathbb{N}} : Y \geq_T X\}$ . Jockusch and Soare [103] proved that for each non-computable set  $X$ , the class  $2^{\mathbb{N}} \setminus \mathcal{C}_X$  is a basis for the  $\Pi_1^0$  classes. In other words, if  $X$  is a non-computable set, any non-empty  $\Pi_1^0$  class has an element which does not compute  $X$ . The more natural contraposition states that if a set is computable by all the members of a non-empty  $\Pi_1^0$  class, it is necessarily computable. Note that if a  $\Pi_1^0$  class has a computable member the result is obvious, and it becomes interesting only for non-empty  $\Pi_1^0$  classes which do not have any. As with the computably dominated basis theorem, we need a lemma to solve the case of a fixed functional.

**Lemma 4.6.** Let  $X$  be a set,  $\mathcal{P}$  a non-empty  $\Pi_1^0$  class and  $\Phi$  a functional. If  $\Phi^Y = X$  for all  $Y \in \mathcal{P}$ , then  $X$  is computable. ★

PROOF. Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable tree such that  $[T] = \mathcal{P}$ . Suppose that for all  $Y \in [T]$ ,  $\Phi^Y = X$ . Let us show that for any  $n$ , there exists a  $t \in \mathbb{N}$  such that  $\Phi(\sigma, n)[|\sigma|] \downarrow = X(n)$  for every string  $\sigma \in T$  of size  $t$ . Indeed, otherwise, the set  $S = \{\sigma \in T : \Phi(\sigma, n)[|\sigma|] \neq X(n)\}$  is a subtree of  $T$  which contains elements of each length, so by König’s lemma, there exists a path  $Y \in [S] \subseteq [T]$  such that  $\Phi^Y(n) \neq X(n)$ , contradicting our hypothesis.

Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be the computable function which on input  $n$  looks for  $t, v_n \in \mathbb{N}$  such that  $\Phi(\sigma, n)[|\sigma|] \downarrow = v_n$  for every string  $\sigma \in T$  of size  $t$ , and returns  $v_n$ . We have shown that this function is total. We also necessarily

have  $v_n = X(n)$  for all  $n$  because otherwise each element of  $\mathcal{P}$  computes something other than  $X$  on the bit  $n$ . So  $X$  is computed by  $g$  and is therefore computable. ■

**Theorem 4.7 (Jockusch et Soare [103])**

*Let  $X$  be a non-computable set and  $\mathcal{P}$  a non-empty  $\Pi_1^0$  class. Then, there exists an element of  $\mathcal{P}$  which does not compute  $X$ .*

PROOF. Let  $X$  be a non-computable set and  $\mathcal{P}$  a non-empty  $\Pi_1^0$  class. We will define an infinite decreasing sequence of non-empty  $\Pi_1^0$  classes  $\mathcal{P} = \mathcal{P}_0 \supseteq \mathcal{P}_1 \supseteq \mathcal{P}_2 \supseteq \dots$  so that no element of  $\bigcap_n \mathcal{P}_n$  computes  $X$ . Let  $\mathcal{P}_0 = \mathcal{P}$ . Suppose  $\mathcal{P}_n$  defined. Let  $\mathcal{B}_{n,m} = \{Y : \Phi_n(Y, m) \uparrow \vee \Phi_n(Y, m) \neq X(m)\}$ . Note that each class  $\mathcal{B}_{n,m}$  is  $\Pi_1^0$  (not uniformly of course because we do not know  $X$ ). Let us show that there exists  $m$  such that  $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$ . If this was not the case, then we would have  $\Phi_n^Y = X$  for all  $Y \in \mathcal{P}_n$ , contradicting Lemma 4.6. So there exists  $m$  such that  $\mathcal{P}_n \cap \mathcal{B}_{n,m} \neq \emptyset$ . We then define  $\mathcal{P}_{n+1} = \mathcal{P}_n \cap \mathcal{B}_{n,m}$  for such an integer  $m$  which gives us  $\Phi_n(X, m) \uparrow$  or  $\Phi_n(X, m) \downarrow \neq X(m)$  for all  $X \in \mathcal{P}_{n+1}$ .

As a decreasing intersection of non-empty closed classes, the class  $\bigcap_n \mathcal{P}_n$  is non-empty. Let  $Y \in \bigcap_n \mathcal{P}_n$ . By construction for all  $n$ ,  $\Phi_n^Y \neq X$  because  $Y \in \mathcal{P}_n$ . So  $X \not\leq_T Y$ . ■

Hirschfeldt [86] gave an elegant alternative proof of the cone avoidance basis theorem, as a simple consequence of the low basis theorem (Theorem 4.3) and of the computably dominated basis theorem (Theorem 4.5).

ALTERNATIVE PROOF OF THEOREM 4.7. Two cases arise:

- Case 1:  $X$  is  $\Delta_2^0$ . In particular, by Proposition 7-4.7,  $X$  is hyperimmune. By the computably dominated basis theorem (Theorem 4.5),  $\mathcal{P}$  contains a computably dominated set  $P$ . In particular,  $P$  does not compute  $X$ .
- Case 2:  $X$  is not  $\Delta_2^0$ . By the low basis theorem (Theorem 4.3),  $\mathcal{P}$  contains a low set, so  $\Delta_2^0$ . In particular,  $P$  does not compute  $X$ . In each case,  $\mathcal{P}$  contains an element which does not compute  $X$ . ■

We will see in the chapters to come many other theorems concerning the  $\Pi_1^0$  classes.

## 5. Basis for perfect $\Pi_1^0$ classes

We have seen that the non-empty  $\Pi_1^0$  classes with no computable element are necessarily perfect. These classes admit reinforced basis theorems, and

one can in particular construct perfect subclasses all of whose elements have a weakness property fixed in advance. Here, we see an example with computably dominated sets.

The idea is to start again the proof of the computably dominated basis theorem, but by duplicating the construction step by step.

**Theorem 5.1**

*Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class containing no computable element. There exists a perfect class  $\mathcal{B} \subseteq \mathcal{P}$  which contains only computably dominated sets.*

PROOF. Let  $P_\epsilon = \mathcal{P}$ . Suppose that for  $n$  and each  $\sigma \in 2^{<\mathbb{N}}$  of size  $n$ , we have defined pairwise disjoint non-empty  $\Pi_1^0$  classes  $\mathcal{P}_\sigma \subseteq \mathcal{P}$ . We repeat the construction of Theorem 4.5 to define for each  $\sigma$  a non-empty  $\Pi_1^0$  class  $\mathcal{Q}_\sigma \subseteq \mathcal{P}_\sigma$  such that either there is an  $m$  such that  $\Phi_n(X, m) \uparrow$  for all  $X \in \mathcal{Q}_\sigma$ , or there is a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\Phi_n(X, m) < g(m)$  for all  $m$  and for all  $X \in \mathcal{Q}_\sigma$ . As  $\mathcal{P}$  does not contain any computable point then for all  $\sigma$ , neither does  $\mathcal{Q}_\sigma \subseteq \mathcal{P}$ . So according to Corollary 3.7 there must be  $\tau_0, \tau_1$  incomparable such that  $\mathcal{Q}_\sigma \cap [\tau_0]$  and  $\mathcal{Q}_\sigma \cap [\tau_1]$  are both non-empty. We define  $\mathcal{P}_{\sigma 0} = \mathcal{Q}_\sigma \cap [\tau_0]$  and  $\mathcal{P}_{\sigma 1} = \mathcal{Q}_\sigma \cap [\tau_1]$ .

For each  $X \in 2^\mathbb{N}$ , the class  $\bigcap_n \mathcal{P}_{X \upharpoonright n} \subseteq \mathcal{P}$  contains exactly one element  $G_X$ , this element is computably dominated, and by construction  $X \neq Y$  implies  $G_X \neq G_Y$ . The class of  $G_X$  for  $X \in 2^\mathbb{N}$  in fact forms a perfect tree, whose nodes are determined by the choice of incomparable extensions  $\tau_0, \tau_1$ . ■

The duplication technique of the previous theorem can also be applied to the cone avoidance basis theorem, but of course it cannot be used with the low basis theorem, because the class of low sets is countable. The reader can try to apply it anyway, in order to see what goes wrong.

Finally, note that it is of course not necessary to go through  $\Pi_1^0$  classes to build a perfect class of computably dominated sets, and we can apply the same idea of construction duplication to the proof of f-trees:

**Exercise 5.2. (★)** Construct a perfect class of computably dominated sets via f-trees. ◇

**Exercise 5.3. (★)** Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class with no computable point. Mix the above construction with the proof of Lemma 4.6 to build a perfect subclass of  $\mathcal{P}$  whose elements are computably dominated, and whose Turing degrees are pairwise incomparable. ◇

**Exercise 5.4. (★★)** Let  $\mathcal{P}$  be a perfect class. Construct a perfect subclass of  $\mathcal{P}$  whose elements are pairwise incomparable in terms of Turing degrees.  
 $\diamond$

**Exercise 5.5. (★)** Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class without isolated point. Show that  $\emptyset'$  computes a non-computable element of  $\mathcal{P}$ .  
 $\diamond$

Note that the last exercise necessarily uses the fact that  $\mathcal{P}$  does not contain any isolated point. We will see with Proposition 30-3.5 a simple technique, but very powerful, allowing to build  $\Pi_1^0$  classes — with isolated points — whose elements are either finite sets, or sets of “very high” computational complexity.

## 6. PA degrees

We take here a little advance on Chapter 9, in which we expose the notions of first-order logical theory, of the formal system of Peano arithmetic, as well as of the first incompleteness theorem of Gödel: the notion of PA degree was born in direct link with these notions. We will however quickly abstract from this historical aspect to give with Theorem 6.2 a characterization of PA degree involving only computability-theoretic notions already seen.

The study of PA degrees — acronym of “Peano Arithmetic” — goes back to the work of Gödel and his famous incompleteness theorem: there is no computable, complete and consistent extension of the axioms of Peano arithmetic<sup>1</sup>. The study of Turing degrees developing, the question of the power necessary to compute such an extension arose quite naturally. We are going to see that the developments around this question have lead to one of the richest concepts of Computability Theory, which probably found its climax through the study of Reverse Mathematics.

In order to speak about the computational power of a theory, we first need to cast the related notions to the setting of Computability Theory, and in particular to represent theories as sets of integers. In what follows, let us fix a computable enumeration  $\psi_0, \psi_1, \psi_2, \dots$  of all the formulas of arithmetic. Suppose also that there exists a computable function  $\text{neg} : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\psi_{\text{neg}(n)} = \neg\psi_n$ . For the following theorem (Theorem 6.2), we will call *theory* a set  $T \subseteq \mathbb{N}$  such that for all  $m$ , if  $\{\psi_n : n \in T\} \vdash \psi_m$ , then  $m \in T$ . In other words, a theory is a set of arithmetic formulas closed under logical consequence. A theory  $T$  is *consistent* if the code of the formula “ $0 = 1$ ” does not belong to  $T$ . A theory  $T$  is *complete* if for all  $n$ , either  $n \in T$

---

<sup>1</sup>This version of the theorem is in fact a reinforcement of that of Gödel, which was proved by Rosser.

or  $\text{neg}(n) \in T$ . The reader who approaches these notions for the first time will find more details in Chapter 9.

**Definition 6.1.** A *completion of Peano arithmetic* is a complete theory  $T$  containing  $\{n \in \mathbb{N} : PA \vdash \psi_n\}$ . A Turing degree is *PA* if it contains a consistent completion of Peano arithmetic.  $\diamond$

The PA degrees being upward-closed, it is equivalent for a degree to be PA and to contain a set which computes a consistent completion of Peano arithmetic. We now show an equivalence which will serve as a characterization for the PA degrees.

**Theorem 6.2 (Jockusch et Soare [95], Solovay (non publié))**

*Let  $X$  be a set. The following statements are equivalent:*

- (1)  $X$  is of degree PA.
- (2)  $X$  is of  $\text{DNC}_2$  degree, i.e., the set  $X$  computes a function  $f : \mathbb{N} \rightarrow \{0, 1\}$  such that  $f(n) \neq \Phi_n(n)$  for all  $n$ .

Before going to the proof, we refer the reader to Definition 7-2.7 who introduced the notion of degree  $\text{DNC}_f$  for a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $2 \leq f(n) \leq f(n+1)$ . The notion of degree  $\text{DNC}_2$  is the strongest possible of this order: the computed function  $f$  has only two possibilities (0 or 1) to differ from each  $\Phi_n(n)$ . We will see with the corollaries 18-4.3 and 19-1.8 that many sets of DNC degree are not  $\text{DNC}_2$ .

PROOF. The equivalence shown by Jockusch and Soare uses Scott's basis theorem [193] for PA degrees, which states that any PA degree computes an infinite path in any non-empty  $\Pi_1^0$  class. Here we show the equivalence directly.

The implication (1)  $\rightarrow$  (2) is essentially the Gödel-Rosser theorem, which extends Gödel's first incompleteness theorem, and which will be formally proved with Theorem 9-3.10 and Corollary 9-3.11.

Let us show (2)  $\rightarrow$  (1). Let  $f \leq_T X$  be a  $\{0, 1\}$ -valued function such that  $f(n) \neq \Phi_n(n)$  for all  $n$ . We are going to define a uniformly  $f$ -computable increasing sequence of consistent theories  $PA = T_0 \subseteq T_1 \subseteq \dots$  such that  $T = \bigcup_n T_n$  is complete. Let  $T_0 = PA$ . Suppose  $T_n$  is consistent. We consider the arithmetic formula  $\psi_n$  of code  $n$  and we define the machine code  $e_n$  such that  $\Phi_{e_n}(e_n) = 1$  si  $T + \psi_n \vdash 0 = 1$  and  $\Phi_{e_n}(e_n) = 0$  si  $T + \neg\psi_n \vdash 0 = 1$ . If  $\Phi_{e_n}(e_n) \downarrow = 0$  then  $T + \neg\psi_n$  is inconsistent and therefore  $T + \psi_n$  is consistent. If  $\Phi_{e_n}(e_n) \downarrow = 1$  then  $T + \psi_n$  is inconsistent and therefore  $T + \neg\psi_n$  is consistent. If  $\Phi_{e_n}(e_n) \uparrow$  then  $T + \psi_n$  and  $T + \neg\psi_n$  are both consistent. Now as  $f(e_n) \neq \Phi_{e_n}(e_n)$ , we can define  $T_{n+1} = T_n + \psi_n$

if  $f(e_n) = 1$  and  $T_{n+1} = T_n + \neg\psi_n$  if  $f(e_n) = 0$ . In all cases we will have a consistent theory.

The theory  $T = \bigcup_n T_n$  is therefore consistent and by construction it is also complete. ■

### Remark

Note that the direction (2)  $\rightarrow$  (1) of Theorem 6.2 works for any consistent theory  $T_0$  whose axioms are computable. Thus, any  $\text{DNC}_2$  function is able to compute a completion of any consistent theory whose axioms are computable. Direction (1)  $\rightarrow$  (2) is more specific to Peano arithmetic, as it requires a sufficiently expressive theory to encode computations by formulas.

Note that  $\emptyset'$  can compute a  $\text{DNC}_2$  function and is therefore of PA degree. The following proposition implies that it is not at all necessary to be Turing complete to compute a complete and consistent extension of Peano arithmetic.

**Definition 6.3.** The *degree spectrum* of a class  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  is the set

$$\deg \mathcal{P} = \{\deg_T X : X \in [P]\}$$

◇

**Proposition 6.4.** There is a  $\Pi_1^0$  class whose degree spectrum corresponds to the PA degrees. ★

PROOF. This is a simple observation, which was already used for the proof of Proposition 3.5. The class of  $\text{DNC}_2$  sets is described as follows.

$$\mathcal{P} = \{X \in 2^{\mathbb{N}} : \forall e \forall t \Phi_e(e)[t] \uparrow \vee \Phi_e(e)[t] \downarrow \neq X(e)\}$$

■

### Corollary 6.5

*There are low PA degrees.*

PROOF. According to Proposition 6.4 and Theorem 4.3. ■

### Corollary 6.6

*There are computably dominated PA degrees.*

PROOF. According to Proposition 6.4 and Theorem 4.5. ■

**Corollary 6.7**

*Let  $A$  be a non-computable set. Then, there exists a PA degree which does not compute  $A$ .*

PROOF. According to Proposition 6.4 and Theorem 4.7. ■

Let us now see another important characterization of the PA degrees, which states that they capture the necessary and sufficient computational power for weak König's lemma.

**Theorem 6.8**

*Let  $X \subseteq \mathbb{N}$ . The following statements are equivalent:*

- (1)  *$X$  is of PA degree.*
- (2)  *$X$  computes a set in each non-empty  $\Pi_1^0$  class.*

*Moreover for (2) the computation is uniform in a code of the  $\Pi_1^0$  class.*

PROOF. For (2)  $\rightarrow$  (1) it suffices to notice that there exists a non-empty  $\Pi_1^0$  class containing only sets of PA degrees (see Proposition 6.4). Let us now show (1)  $\rightarrow$  (2). Let  $f \leq_T X$  be a  $\{0, 1\}$ -valued DNC function, that is, such that  $f(n) \neq \Phi_n(n)$  for all  $n$ . Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class and  $T$  a computable tree such that  $[T] = \mathcal{P}$ .

Let  $\sigma_0 = \epsilon$ . Given  $\sigma_n$  defined such that  $[\sigma_n] \cap [T]$  is not empty, we compute  $\sigma_{n+1} = \sigma_n i$  for  $i \in \{0, 1\}$  as follows: we first compute the code  $e_n$  of a program which on any input  $m$  searches for the smallest  $t$  such that for  $i = 0$  or  $i = 1$  no string  $\sigma$  of size  $t$  with  $\sigma \succeq \sigma_n i$  does not belong to  $T$ . If found, the program halts and outputs  $i$ . According to König's lemma this condition is equivalent to the fact that  $[\sigma_n i] \cap [T]$  is empty. We simply define  $\sigma_{n+1} = \sigma_n f(n)$ . As  $f(e_n) \neq \Phi_{e_n}(e_n)$  we have the guarantee that  $[\sigma_{n+1}] \cap [T]$  is non-empty. ■

### Classe universelle

Note that according to Proposition 6.4, there exists a non-empty  $\Pi_1^0$  class whose members are of PA degree, and that according to Theorem 6.8, any PA degree computes a member of each non-empty  $\Pi_1^0$  class. Such a class is therefore “maximal” in terms of computational complexity, in the sense that if we know how to compute a member of this class, then we know how to compute a member of any non-empty  $\Pi_1^0$  class. We call *universal  $\Pi_1^0$  class* a non-empty  $\Pi_1^0$  class all of whose members are of PA degree.

In the same vein as Theorem 7-7.1, we end with a characterization which now combines the fact of being of high or PA degree. Note the difference with (1)  $\leftrightarrow$  (3) of Theorem 7-6.2 within which we consider a sequence  $(X_n)_{n \in \mathbb{N}}$  containing exactly the computable sets, whereas here we only consider that it contains the computable sets.

**Theorem 6.9 (Jockusch [97])**

*Let  $X \subseteq \mathbb{N}$ . The following statements are equivalent:*

- (1)  *$X$  is of high or PA degree.*
- (2)  *$X$  computes a sequence  $(X_n)_{n \in \mathbb{N}}$  containing all the computable sets.*

PROOF. Let us first show (1) implies (2). If  $X$  is high then the implication is clear from Theorem 7-6.2. Suppose now that  $X$  is of PA degree. Let  $g \leq_T X$  be such that  $g(n) \neq \Phi_n(n)$  for all  $n$ . Note that  $X$  also computes the function  $f(x) = 1 - g(x)$ . In particular,  $\Phi_n(n) \downarrow \in \{0, 1\}$  implies  $f(n) = \Phi_n(n)$ . Given a computable function  $\Phi_e$  and an integer  $n$ , we can compute the code  $a_n$  such that  $\Phi_{a_n}(a_n) = \Phi_e(n)$ . By using this process and the fact that  $\Phi_{a_n}(a_n) \downarrow \in \{0, 1\}$  implies  $f(a_n) = \Phi_{a_n}(a_n) = \Phi_e(n)$  we easily compute a set  $X_e$  such that if  $n \mapsto \Phi_e(n)$  is total and has value in  $\{0, 1\}$  then  $X_e(n) = \Phi_e(n)$  for all  $n$ . We can therefore compute our sequence  $(X_e)_{e \in \mathbb{N}}$  containing all the computable sets.

Let us now show (2) implies (1). Let  $(X_n)_{n \in \mathbb{N}}$  be an  $X$ -computable sequence containing all the computable sets. The idea is to proceed initially as in the proof of (3)  $\rightarrow$  (1) of Theorem 7-6.2. Given a  $\Pi_2^0$  predicate of the form

$$P = \{e : \forall x_1 \exists x_2 R(e, x_1, x_2)\},$$

the idea was to define uniformly in  $e$  a partial computable function  $f_e$  such that:

- (a)  $e \in P$  implies that  $f_e$  is a total computable function.
- (b)  $e \notin P$  implies that  $f_e$  is a partial function which has no computable completion.

It suffices to notice that in Theorem 7-6.2, the definition of  $f_e$  which is given is such that in case (b), not only no completion of  $f_e$  is computable, but in addition such a completion is necessarily of PA degree. The definition was as follows: Let  $e$  fixed. At the stage of computation  $t$ , for any value  $n$  smaller than  $t$  and such that  $f_e$  does not halt for the moment on  $n$ , we proceed as follows : if  $\Phi_n(n)[t] \downarrow \neq 0$  we define  $f_e(n) = 0$ . If  $\Phi_n(n)[t] \downarrow \neq 1$  we define  $f_e(n) = 1$ . Otherwise, if for all  $k \leq n$  there exists  $m_k \leq t$  such that  $R(e, k, m_k)$  then we define  $f_e(n) = 0$ .

As in the proof of Theorem 7-6.2, if  $e \in P$  then  $f_e$  is a total function. Otherwise we notice that for almost all the values of  $n$  such that  $\Phi_n(n) \downarrow$  we have  $f_e(n) \neq \Phi_n(n)$ . Any completion of  $f_e$  is therefore a DNC<sub>2</sub> function, modulo a finite number of values, and is therefore of PA degree.

There are now two possibilities: either  $(X_n)_{n \in \mathbb{N}}$  contains a set of PA degree, in which case we have (1). Either this is not the case, in which case we can give a  $\Sigma_2^0(X)$  definition of  $P$  as in the proof of Theorem 7-6.2, which implies, applied to  $P = \mathbb{N} \setminus \emptyset''$  that  $\emptyset''$  is  $\Delta_2^0(X)$  and therefore  $X$  is high. ■

We end this section with an exercise which constitutes an alternative and well-known characterization of the PA degrees.

**Exercise 6.10.** (★) Show that  $X$  is PA iff for all c.e. sets  $A, B$  with  $A \cap B = \emptyset$ , there exists an  $X$ -computable set  $C$  such that  $A \subseteq C$  and  $C \cap B = \emptyset$ .  
◇

## 7. Finitely-branching trees

We introduce here the *Baire space*: the class  $\mathbb{N}^{\mathbb{N}}$  of all the infinite valued sequences in  $\mathbb{N}$ , or in other words the class of all the functions from  $\mathbb{N}$  to  $\mathbb{N}$ . Just as Cantor space has its set of strings  $2^{<\mathbb{N}}$ , Baire space has its set of strings  $\mathbb{N}^{<\mathbb{N}}$ : finite sequences with values in  $\mathbb{N}$ . The different operations that we have seen on binary strings (prefix, concatenation, length, ...) extend without problem to strings in Baire space. In particular, given a string  $\sigma \in \mathbb{N}^{<\mathbb{N}}$ , we denote by  $[\sigma]$  the class of sequences  $P \in \mathbb{N}^{\mathbb{N}}$  such that  $\sigma \prec P$ . The notion of tree also extends to subsets of  $\mathbb{N}^{<\mathbb{N}}$  as follows:

**Definition 7.1.** A set  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  is a *tree* if  $T$  is closed under prefix, that is to say for all  $\sigma \in T$  and  $\tau \preceq \sigma$ , then  $\tau \in T$ . ◇

Unlike binary trees, nodes can have an infinite number of successors. A node  $\sigma \in T$  is *branching* if it has at least two successors. A *path* of  $T$  is a sequence  $P \in \mathbb{N}^{\mathbb{N}}$  whose initial segments are all in  $T$ . We denote by  $[T]$  the class of paths of  $T$ . König's lemma no longer works on trees in Baire space, as the following counterexample shows.

**Example 7.2.** Let  $T = \{\sigma \in \mathbb{N}^{<\mathbb{N}} : \forall n < |\sigma| \sigma(n) \geq |\sigma|\}$ . The tree  $T$  contains nodes of arbitrary length and is infinite, but  $[T] = \emptyset$ .

The computational power of the paths of an arbitrary computable tree of Baire space will be studied in Part IV on Higher Computability Theory. In this section, we will restrict ourselves to a sub-category of trees falling

into the realm of König's lemma: the trees  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  which are *finitely-branching*, i.e., within which each node has a finite number of successors.

**Lemma 7.3 (König's lemma).** Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be a finitely-branching tree such that  $|T| = \infty$ . Then,  $[T]$  is non-empty. ★

PROOF. We construct a path  $X$  by induction on  $n$ . As  $T$  is infinite, but the root  $\epsilon$  has only a finite number of successors, by the pigeonhole principle, there exists  $i \in \mathbb{N}$  and an infinity of nodes  $\sigma \in T$  which extend  $i$  (ie with  $i \prec \sigma$ ). We define  $X(0) = i$ . Suppose that  $\tau = X(0)X(1)\dots X(n)$  is defined with  $\tau \in T$  and such that there is an infinity of nodes  $\sigma \in T$  for which  $\tau \preceq \sigma$ . The node  $\sigma$  having only a finite number of successors, by the pigeonhole principle, there exists  $i \in \mathbb{N}$  and an infinity of nodes  $\sigma \in T$  which extend  $\tau i$ . We define  $X(n+1) = i$ .

By induction on  $n$ , we thus define in this way a set  $X$  such that  $X \upharpoonright_n \in T$  for all  $n$ . ■

### Baire space

As for Cantor space, the open classes of Baire space are the classes  $\mathcal{O} \subseteq \mathbb{N}^{\mathbb{N}}$  of the form  $\mathcal{O} = \bigcup_{\sigma \in W} [\sigma]$  for a set  $W \subseteq \mathbb{N}^{<\mathbb{N}}$ , and the closed class  $\mathcal{P} \subseteq \mathbb{N}^{\mathbb{N}}$  are of the form  $[T]$  for a tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ . On the other hand, unlike Cantor space, the closed classes of Baire space are not compact in general. The compacts of Baire space are precisely the closed classes  $\mathcal{P}$  of the form  $[T]$  for a finitely-branching tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ .

The proof of König's lemma is almost the same as that of its weak version, and one might expect at first glance that the computational power necessary to compute a path from a computable finitely-branching tree is that of PA degrees. This is not the case, however, as shown by Proposition 7.4.

**Proposition 7.4.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a  $\Delta_2^0$  binary tree. There exists a finitely-branching computable tree  $S$  such that  $\deg([T]) = \deg([S])$ . ★

PROOF. Let  $(T_n)_{n \in \mathbb{N}}$  be a  $\Delta_2^0$  approximation of  $T$ . We can assume without loss of generality that for any  $n$ ,  $T_n$  is closed under prefix and  $T_n \subseteq 2^{\leq n}$  (the set of strings of size less than or equal to  $n$ ). We easily show that any union of trees is a tree, which implies that  $\bigcup_n T_n$  is a tree.

Let us show that  $[\bigcup_n T_n] = [T]$ . Clearly,  $T \subseteq \bigcup_n T_n$ , so  $[T] \subseteq [\bigcup_n T_n]$ . Let  $P \in [\bigcup_n T_n]$ . Let  $s \in \mathbb{N}$  and show that  $P \upharpoonright_s \in T$ .  $(T_n)_{n \in \mathbb{N}}$  being a  $\Delta_2^0$  approximation of  $T$ ,  $P \upharpoonright_s \in T$  iff  $\forall t \exists n \geq t \ P \upharpoonright_s \in T_n$ . Let  $t \geq s$ . As  $P \upharpoonright_t \in \bigcup_n T_n$  and as by hypothesis  $\bigcup_{n < t} T_n \subseteq 2^{<t}$ , then  $P \upharpoonright_t \in T_n$  for  $n \geq t$ . By downward-closure of  $T_n$  we have  $P \upharpoonright_s \in T_n$ . So  $\forall t \exists n \geq t \ P \upharpoonright_s \in T_n$ . So  $P \upharpoonright_s \in T$ .

Let  $\sigma, \tau \in \mathbb{N}^{<\mathbb{N}}$  have the same length. We denote by  $\langle \sigma, \tau \rangle$  the string  $\rho$  of length  $|\sigma|$  such that for all  $n < |\rho|$ ,  $\rho(n) = \langle \sigma(n), \tau(n) \rangle$ . The operation naturally extends to infinite sequences  $P, Q$  for which we will write  $\langle P, Q \rangle$ . We are going to build a computable finitely-branching tree  $S \subseteq \mathbb{N}^{<\mathbb{N}}$  whose paths will be of the form  $\langle P, Q \rangle$  with  $P \in [\bigcup_n T_n] = [T]$  and  $Q$  a “witness” of  $P \in [\bigcup_n T_n]$ , in the sense where for all  $s$ ,  $P \upharpoonright_s \in T_{Q(s)}$ .

Define a partial computable function  $f : \bigcup_n T_n \rightarrow \mathbb{N}^{<\mathbb{N}}$  which sends strings to strings of the same length inductively as follows:  $f(\epsilon) = \epsilon$ . If  $\sigma i \in \bigcup_n T_n$  then  $f(\sigma i) = f(\sigma) \frown s$  where  $s$  is the smallest integer such that  $\sigma i \in T_s$ . By continuity, the function  $f$  extends to infinite sequences of  $[\bigcup_n T_n] = [T]$ . Let  $S = \{\langle \sigma, f(\sigma) \rangle : \sigma \in \bigcup_n T_n\}$ . Note that  $\bigcup_n T_n$  is not computable in general, but that  $S$  is because for any  $\rho = \langle \sigma, \mu \rangle$ , it is easy to verify that  $f(\sigma) = \mu$ . The set  $S$  is closed under prefix, because  $\bigcup_n T_n$  is also closed and  $f(\sigma i) \upharpoonright_{|\sigma|} = f(\sigma)$  for all  $\sigma \in \bigcup_n T_n$ . Thus,  $S$  is a computable tree. Note also that  $S$  is 2-branching, therefore finitely-branching.

Let us show that  $\deg([T]) = \deg([S])$ . Let  $P \in [T]$ . Then,  $\langle P, f(P) \rangle \in [S]$  and  $P \equiv_T \langle P, f(P) \rangle$ . Let  $R \in [S]$ . Then,  $R = P \oplus f(P)$  for a  $P \in [\bigcup_n T_n] = [T]$ . Likewise,  $R \equiv_T P$ . This concludes the proof of Proposition 7.4. ■

### Corollary 7.5

*There exists a finitely-branching computable tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  and a PA degree  $P$  which does not compute a path through  $T$ .*

PROOF. Let  $S = \{\emptyset' \upharpoonright_n : n \in \mathbb{N}\}$  be the  $\Delta_2^0$  binary tree having  $\emptyset'$  for unique infinite path. By Proposition 7.4, there exists a computable finitely-branching tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  such that  $\deg([T]) = \deg([S])$ . In particular, any path of  $T$  computes  $\emptyset'$ . By Corollary 6.5, there is a degree both PA and low. In particular, this degree does not compute a path through  $T$ . ■

We can relativize the notion of being  $\text{DNC}_2$  relative to an oracle  $X$ : we ask for the computation of a function  $f : \mathbb{N} \rightarrow \{0, 1\}$  such that  $f(n) \neq \Phi_n(X, n)$  for all  $n$ . Theorem 6.8 is relativized well in the sense that the  $\text{DNC}_2$  degrees relative to  $X$ , which one will also call PA degrees relative to  $X$  or PA( $X$ ), coincide with those allowing to compute a path in any non-empty  $\Pi_1^0(X)$  class.

**Exercise 7.6.** Let  $Y$  be a PA( $X$ ) set. Show that  $Y \geq_T X$ . ◇

We deduce that a PA degree relative to  $\emptyset'$  is necessary to compute a path in any infinite computable finitely-branching tree. Note that the situation of Exercise 7.6 is different when we consider DNC degrees instead of  $\text{DNC}_2$

degrees. More precisely, if  $X$  is a non-computable set, there exists a set  $Y$  of DNC degree relative to  $X$  which does not compute  $X$  (see Corollary 18-4.4). This result brings into play notions of the Algorithmic Randomness that we will discuss in chapter Chapter 18.

### Binary tree vs 2-branching tree

The tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  built in the proof of Proposition 7.4 is 2-branching, in the sense that each node has at most two successors. From a purely structural point of view, it is therefore isomorphic to a binary tree  $S \subseteq 2^{<\mathbb{N}}$ . However, there are PA degrees that do not compute a path in this tree. The difference between the computational power of this tree and that of a binary tree does not therefore come from a combinatorial difference, but simply stems from a lack of information on the successors of a node: given a computable finitely-branching tree, one cannot limit in a computable and uniform way the maximum value of the successor of a node.

The preceding remark leads us to the following definition.

**Definition 7.7.** A tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  is *computably bounded* if there exists a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $\sigma \in T$  and  $n < |\sigma|$ ,  $\sigma(n) < g(n)$ .  $\diamond$

It is clear that any computably bounded tree is finitely-branching. The following proposition makes it possible to reconcile the idea according to which combinatorially similar objects should have the same computational power, by showing that as soon as the finitely-branching tree is accompanied by a computable bound on its branching, then the computational power necessary for compute a path is exactly that of PA degrees. Given a function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , we will denote by  $f^{<\mathbb{N}}$  the set of strings  $\sigma \in \mathbb{N}^{<\mathbb{N}}$  such that for all  $n < |\sigma|$ ,  $\sigma(n) < f(n)$ .

**Proposition 7.8.** For any computable, computably bounded tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ , there exists a binary tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $\deg([T]) = \deg([S])$ .  $\star$

**PROOF.** The idea of the proof is quite simply to define a binary encoding of the strings, using the computational bound to know how many bits to allocate at each level. To remove any ambiguity, we will denote by  $2^{=n}$  the set of binary strings of length  $n$ , instead of  $2^n$ , which will denote the  $n$ -th power of 2. Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be a computable function such that  $T \subseteq g^{<\mathbb{N}}$ . Without loss of generality, we can assume that  $g(n) = 2^{h(n)}$  for all  $n$ , with  $h : \mathbb{N} \rightarrow \mathbb{N}$  a computable function.

For all  $n$ , let  $e_n : 2^n \rightarrow 2^{=n}$  be the canonical bijection. For example,  $e_2 : 4 \rightarrow \{00, 01, 10, 11\}$  is defined by  $e_2(0) = 00, e_2(1) = 01, e_2(2) = 10$

and  $e_3(3) = 11$ . This coding extends into a computable bijection  $e : g^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  defined by

$$e(\sigma) = e_{h(0)}(\sigma(0)) \frown e_{h(1)}(\sigma(1)) \frown \dots \frown e_{h(|\sigma|-1)}(\sigma(|\sigma|-1)),$$

where  $\frown$  denotes here for more clarity the concatenation by  $\frown$ . For example, if  $h(n) = n + 1$ , then  $g(n) = 2^{h(n)} = 2^{n+1}$ , and  $e(032) = e_1(0) \frown e_2(3) \frown e_3(2) = 0 \frown 11 \frown 010 = 011010$ .

Note that the set  $\hat{S} = \{e(\sigma) : \sigma \in T\}$  is not a tree, because it is closed under prefix only for the initial segments of length exactly  $\text{Im } g$ . We must therefore define the tree  $S$  as the prefix closure of  $\hat{S}$ , in other words  $S = \{e(\sigma) \upharpoonright_n : \sigma \in T \wedge n \in \mathbb{N}\}$ . The coding function  $e$  being monotonic on the lengths, and the set  $T$  being closed under prefix,  $\rho \in S$  if and only if there exists a string  $\sigma \in g^{<\mathbb{N}}$  of length at most  $|\rho|$  such that  $\rho \prec e(\sigma)$ . Thus,  $S$  is an infinite computable binary tree, whose paths are exactly infinite sequences of the form  $e_{h(0)}(P(0)) \frown e_{h(1)}(P(1)) \frown \dots$  for  $P \in [T]$ . Thus,  $\deg([T]) = \deg([S])$ . ■

### Corollary 7.9

*Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be a computable, computably bounded infinite tree. Any PA degree computes a path of  $T$ .*

PROOF. By Proposition 7.8, there exists a binary tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $\deg([T]) = \deg([S])$ . By Theorem 6.8, any PA degree computes a path of  $S$ , so any PA degree computes a path of  $T$ . ■

We now see the converse of Proposition 7.4, which shows that the PA degrees relative to  $\emptyset'$  are exactly those able to compute a path in a computable finitely-branching tree.

**Proposition 7.10.** Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be an infinite computable finitely-branching tree. There is a  $\Delta_2^0$  binary tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $\deg([T]) = \deg([S])$ . ★

PROOF. Note first that any infinite computable finitely-branching tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  is  $\emptyset'$ -computably bounded, that is to say that there exists a  $\emptyset'$ -computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $T \subseteq g^{<\mathbb{N}}$ . The proof of Proposition 7.8 is relativized to  $\emptyset'$ , and allows to define a  $\Delta_2^0$  binary tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $\deg([T]) = \deg([S])$ . ■

Let's finish this section with a few exercises. The *prefix closure* of a set  $S \subseteq \mathbb{N}^{<\mathbb{N}}$  is the set

$$\hat{S} = \{\tau \in \mathbb{N}^{<\mathbb{N}} : \exists \sigma \in S \ \tau \preceq \sigma\}.$$

**Exercise 7.11.** Show that for any infinite set  $S \subseteq 2^{<\mathbb{N}}$ , its prefix closure admits a path.  $\diamond$

**Exercise 7.12.** Show that there exists a computable infinite set  $S \subseteq 2^{<\mathbb{N}}$  such that  $[\hat{S}] = \{\emptyset'\}$ .  $\diamond$

**Exercise 7.13. (\*\*)** Show that for any infinite  $\emptyset'$ -computable tree  $T \subseteq 2^{<\mathbb{N}}$ , there exists an infinite set  $S \subseteq 2^{<\mathbb{N}}$  containing exactly one string of each length, such that  $[T] = [\hat{S}]$ .  $\diamond$

## Summary Diagram

Here is a figure which summarizes the various concepts approached so far.

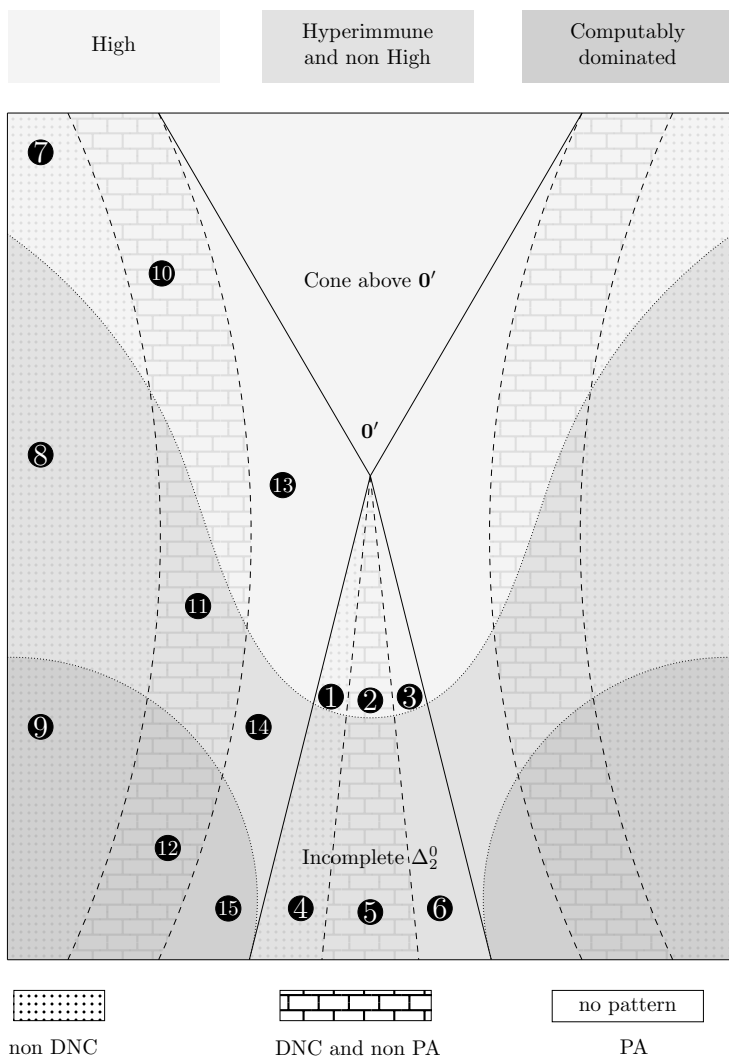


Figure 7.14: Summary on the Turing degrees seen so far. Points (1) to (6) refer to examples of sets of each type. Points (7) to (15) refer to the existence of a perfect class of sets of each type (hence according to Exercise 5.4) to the existence of a perfect class of Turing degrees of each type.)

The points from (1) to (15) which follow use for some of the concepts which will be seen only in the following chapters.

- (1) There exists a  $\Delta_2^0$  Turing degree not DNC and high: It suffices to mix the proof of Exercize 4-10.6 with that of Exercize 7-2.9 to obtain a high set which in addition to be incomplete, is not DNC.
- (2) There exists a  $\Delta_2^0$  Turing degree which is DNC, non PA and high: We start from a DNC, non PA and low set  $X$  (by Theorem 18-4.1 and Corollary 18-2.3 on can take for example a low random in the sense of Martin-Löf). We can then embroider on the construction of Exercize 4-10.6 and on that of Exercize 7-2.9 to build a set  $Y$   $\Delta_2^0$  high such that  $X \oplus Y$  is not PA (using in particular the fact that  $X' \leq_T \emptyset'$ ).
- (3) There exists an incomplete  $\Delta_2^0$  Turing degree which is PA and high: We start from a PA and low set  $X$  (see Corollary 6.5). We then embroider on the construction of Exercize 4-10.6 to build a  $\Delta_2^0$  high set  $Y$  such that  $X \oplus Y$  is incomplete.
- (4) There exists a  $\Delta_2^0$  Turing degree not DNC, hyperimmune and not high: It suffices to build a non DNC and low set, by mixing Proposition 4-9.1 and Exercize 7-2.9.
- (5) There exists a  $\Delta_2^0$  Turing degree which is DNC, non PA, hyperimmune and not high: It suffices to consider a random set in the sense of Martin-Löf and low. According to Theorem 18-4.1 such a set is DNC. According to Theorem 19-1.7 it is not PA. According to Proposition 7-4.7 it is hyperimmune. Finally, since it is low, it cannot be high.
- (6) There exists a  $\Delta_2^0$  PA, hyperimmune and not high Turing degree: According to Proposition 7-4.7 it suffices to consider a low PA degree given by Corollary 6.5.
- (7) There is a perfect class of high and non-DNC sets. Just use Theorem 10-3.21 and embroider on Posner/Robinson (see Corollary 10-3.34) to build a perfect class of 1-generic and high sets.
- (8) There is a perfect class of hyperimmune, non-high and non-DNC sets. According to Theorem 10-3.2, Proposition 10-3.38 and Corollary 10-3.34 any sufficiently generic set will be in this case there.
- (9) There exists a perfect class of computably dominated and non-DNC sets. We have to take again the construction of computably dominated sets via f-trees, and modify it to produce non-DNC sets as it is done in Exercize 7-2.9.

- (10) There is a perfect class of high, DNC and non PA sets. We can apply Theorem 18-3.4 of Kučera/Gács relativized to  $\emptyset'$  on the 2-random tree to build high and 2-random sets. By Theorem 18-4.1 such sets are DNC. By Theorem 19-1.7 they are not PA.
- (11) There is a perfect class of hyperimmune, non-high, DNC and non-PA sets. According to Theorem 18-4.1, Corollary 19-3.9 and Corollary 19-1.8 this is the case for any sufficiently random set.
- (12) There exists a perfect class of computably dominated, DNC and not PA sets. This is Theorem 5.1 applied to a  $\Pi_1^0$  class containing only random numbers within the meaning of Martin-Löf. According to Theorem 19-1.7 MLR and computably dominated sets cannot be PA.
- (13) There exists a perfect class of incomplete sets, high and PA. We fix a high and incomplete set  $X$ . We then develop on Theorem 4.7 the cone avoidance basis relativized to  $X$ , to build a perfect class of sets PA  $Y$  such that  $X \oplus Y$  is incomplete.
- (14) There is a perfect class of hyperimmune, non-high and PA sets. Let  $X$  be a non-high hyperimmune set. We use the relativization to  $X$  of Theorem 5.1, applied to the  $\Pi_1^0$  class of  $\text{DNC}_2$  sets, to construct a perfect class of sets  $Y$  such that any function computed by  $X \oplus Y$  is dominated by a function computed by  $X$ .
- (15) There exists a perfect class of computably dominated PA sets. This is Theorem 5.1 applied to the  $\Pi_1^0$  class of  $\text{DNC}_2$  sets.

# Chapter 9

## Formal interlude

### 1. A little history: the crisis of foundations

Mathematics have developed naturally over the centuries as a tool at the service of an abstract representation of reality. This state of affairs is, for example, flagrant in physical sciences for which mathematics accurately accounts for a variety of phenomena. Over time, the concepts studied have become more and more complex, more and more abstract, and the connections between mathematics and the real world have become more and more uncertain. For a long time, however, the discipline has been able to rely on the innate logical sense of the human mind to talk about things that we do not “see anymore” while keeping a rigorous framework. Complex numbers constitute a striking example: the negative square numbers, which only exist a priori in the imagination of the mathematician, are baptized in 1545 by Cardan “sophisticated quantities”. Sophistication was undoubtedly needed to accept this UFO as a serious object of study. However these “sophisticated quantities” find their utility in the resolution of very concrete problems. Raphaël Bombelli gives a first formalization in 1572, and shows how to use these numbers to solve certain third degree equations. Over the centuries, they will find many uses in mathematics, as well as in physics, where they are used successfully in equations representing the real world.

It took a certain conceptual leap to accept the development of a rigorous and consistent mathematical framework around complex numbers. Despite everything, let’s say that this concept, however surprising it may be, still remains “relatively simple”. The real problems arise with Cantor’s work

on cardinality and transfinite numbers. Cantor opens the door wide to a bottomless abyss, that takes us far beyond what the human mind can confidently apprehend: the study of the infinite. Of course, infinity has been present in mathematics since antiquity in the first place, through the consideration that there is no greater whole number. But Cantor's epistemological revolution consists in considering the infinite as an object of study in its own right. This consideration will lead to the beginnings of what will become a century later Set Theory with a capital "S".

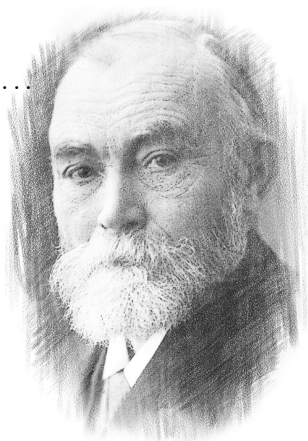
With Cantor's work, the question of knowing *what is* mathematical activity has become more and more pressing: can we really reason about everything, and even about the infinite, a concept beyond us? But if we accept, as is the case today, that we can reason about infinity, we certainly cannot do it just any old way. Basically what is *doing* mathematics? Especially when we start to manipulate objects about which we no longer have so much intuition, how can we be sure that what we are talking about really has a meaning? These considerations found their apogee during the famous "crisis of foundations" which goes from the end of the 19th century to the beginning of the 20th. It is then a question of defining rules to frame the mathematical activity. It is in fact a question of precisely defining the mathematical study, not of objects such as integers, reals or even functions, but of defining the mathematical study *of mathematics itself*. It is remarkable a posteriori to note the success of this enterprise: mathematics is a sufficiently powerful tool to be able to define and study itself, with the rigor inherent in the discipline! It was obviously not an easy path. In this enterprise, the three musketeers — who as we know are four — are called Frege, Russel, Zermelo and Hilbert.

## Frege

German philosopher and mathematician from the end of the 19th century, Gottlob Frege was moved by one certainty: logic precedes mathematics. But the logic of the time is still very poor, and is essentially confined to the work of George Boole, on what is called today *propositional calculus*: the manipulation of propositions, true or false, that one can connect between them via "and" and "or" logic, well known to computer scientists. This system is too small for Frege's ambition to put mathematics on a logical foundation. In particular, nothing in propositional calculus allows us to speak of specific objects through the relations they maintain with one another. He then formalized in his work "Begriffsschrift" a new language in order to overcome the shortcomings of the logic of the time, a language which will evolve to become what we call today *predicate calculus*, ubiquitous in mathematics.

Frege is considered today as the father of modern logic, notably through his concept of quantified variables  $\forall x \dots, \exists x \dots$ .

He uses his formalism to tackle in his following works “Foundations of arithmetic” (1884) and “Fundamental laws of arithmetic I and II” (1893 and 1903) the foundation of arithmetic on logic. For this, he proposed a definition of natural integers which can be seen today as based on the concept of *set* and that of *comprehension scheme*: if  $\Psi(x)$  is a mathematical formula which can be true or false according to  $x$ , then the set of elements which satisfy this formula:  $\{x : \Psi(x)\}$ , is well defined.



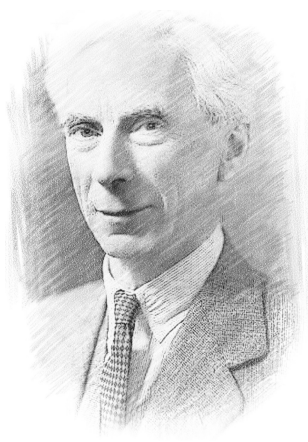
Gottlob Frege, 1848–1925

## Russell

In 1902, Russell in a letter to Frege expressed doubts about his work. Let  $y$  be the set of sets which do not belong to each other:  $y = \{x : x \notin x\}$ . Do we then have  $y \in y$  or  $y \notin y$ ? We can easily understand the paradoxical situation we have reached. This example will be famous as *Russell's paradox*. At 53, Frege realizes that Russell's paradox implies the collapse of the system he took years to build. A hard blow from which he will have great difficulty recovering.

Despite this paradox, Russell welcomes Frege's work with great enthusiasm, and goes a long way in promoting its value. Like Frege, Russell feels the need to put mathematics on a solid foundation.

Much like Frege, Russell has this intuition that logic precedes mathematics. He will tackle ten years during, with his former professor Alfred Whitehead, this search for logical foundations, which will lead to their famous work “*Principia Mathematica*”: a titanic work which extends over more than 2,000 pages, whose ambition is to describe a set of logical axioms and inference rules from which any mathematical truth could be demonstrated.



Bertrand Russell, 1872–1970

These works lay the foundations of what we call today *Type Theory*, a system still studied today, presenting very strong links with programming languages. In parallel, the Set Theory developed, the axiomatization of which was initiated by Zermelo, and completed later by Fraenkel and Skolem independently, to give the axiomatic system ZF, named after Zermelo-Fraenkel.

## Zermelo

An absolutely remarkable fact is that the ZF system, to which it is sometimes necessary to add the axiom of choice, gives a framework within which can be formalized *the totality of modern mathematics*, if we exclude recent developments in Set Theory, the objective of which is precisely to get out of this system. Zermelo finds a way to avoid Russell's famous paradox — like Russell himself with his theory of types — by limiting the axiom of comprehension. The set  $\{x : \Psi(x)\}$  is no longer valid, it is necessary to start from an existing set  $y$ , in which case we can now define the set of elements of  $y$  which satisfy  $\Psi$ :  $\{x \in y : \Psi(x)\}$ . However, this theory does not immediately reach consensus. Poincaré, if he never actively took part in the crisis of foundations, followed its developments with interest. Like all protagonists of the time, he is keenly aware of the danger behind Russell's paradox. He even theorizes the problem as *impredicativity*<sup>1</sup>: An impredicative definition is in substance a circular definition in which the object that is defined is itself likely to be used in the definition. This is what happens when we define  $y = \{x : x \notin x\}$ : the set  $y$  defined to the left of the equality is also concerned to the right of the equal sign, since we potentially consider all the sets. If Zermelo's axiomatic avoids Russell's paradox, it nevertheless remains indirectly impredicative, as Poincaré will notice, who will write [175]:



Ernst Zermelo, 1871–1953

*“By assuming in advance its set  $M$  [Poincaré then speaks of the bound used by Zermelo in the axiom of comprehension, which makes it possible to define for an existing set  $M$   $\{x \in M : \Psi(x)\}$ ], he [Mr. Zermelo] has erected a wall which stops any disturbers who might come from outside. But he does not ask himself if there may be hindrances from within that he has*

---

<sup>1</sup>Previously used by Russell in a slightly different sense.

*locked up with him in his wall. If the set  $M$  has an infinity of elements, this does not mean that these elements can be conceived as existing in advance all at the same time, but that new ones can constantly be born; they will be born inside the wall, instead of being born outside, that's all."*

Zermelo's system considers that if a set  $A$  exists, then the set of its parts  $\mathcal{P}(A)$  also exists. This axiom combined with the axiom of restricted comprehension allows circular definitions to be made. Thus, in the definition  $A = \{n \in \mathbb{N} : \forall S \in \mathcal{P}(\mathbb{N}) \ n \notin S\}$ , the quantifier  $\forall S$  will take for value all the subsets of  $\mathbb{N}$ , and in particular the set  $A$  itself. The set  $A$  is therefore defined as a function of itself. Poincaré then adds at the end of his argument:

*"But if he [Mr. Zermelo] has closed his sheepfold well, I'm not sure the wolf hasn't locked up there. I would only be reassured if he had shown that he is immune to contradiction."*

We can hardly prove Poincaré wrong: how can we be sure that a Russell's paradox will not appear out of nowhere, at the bend of a hidden circular definition? Zermelo himself is aware of the problem, and will seek to demonstrate without success that his axiomatic system is *consistent*, that is to say free from paradox. This search for proof of the consistency of mathematics reached its peak around 1920, under the leadership of David Hilbert.

## Hilbert

Hilbert is certainly — just like Poincaré — one of the last mathematicians to have an in-depth knowledge of all the mathematics of his time. His work is considerable, and he profoundly influenced the developments of the discipline during the 20th century. He takes an active part in the crisis of the foundations by opposing to Russell a *formalist* vision of mathematics, rather than a *logicist* vision. For Hilbert, mathematics must be able to be reduced to a set of rules, which we must be applicable in a purely mechanical way and disconnected from any psychology of the mathematician. He thus imagines proof systems, within which he differentiates *axioms*, which are the

mathematical sentences assumed to be true, for example the Zermelo-Fraenkel axioms, and *deduction rules*, which allow the axioms to be combined together to deduce theorems. It is Hilbert's vision that will eventually prevail, even if it does not happen without a stir. The ultimate objective for Hilbert is to show via deduction systems considered to be



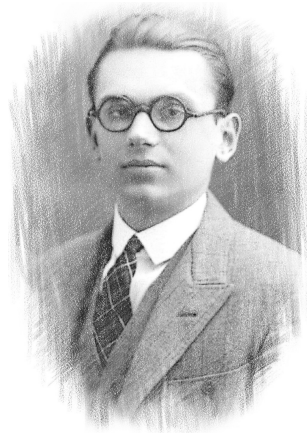
reliable — in particular via finite reasoning on finite objects — that the set of mathematics, which they call on infinite objects whose relevance is subject to caution, forms a consistent system, that is to say free from paradox: this is what we will call *Hilbert's program*, of which the Entscheidungsproblem mentioned in Section 6-1 constitutes one of the aspects. This program will come to a sudden stop with the work of Gödel, who demonstrates ten years later his famous incompleteness theorem: arithmetic itself is powerless to demonstrate that it is free from paradox.

### The Fifth Musketeer

Gödel's work brings a conclusion as masterful as it is unexpected to the crisis of foundations. Gödel shows two things: even the simplest and best understood systems, like arithmetic, which speaks only of finite objects, contain unprovable truths. In particular, and supposing it to be true that the axioms of arithmetic form a system free from paradoxes, then the consistency of arithmetic is itself one of those unprovable truths. Gödel finally shows — with the help of Rosser — that the addition of axioms changes nothing: any consistent system of axioms, containing arithmetic, and “whose axioms can be known” cannot demonstrate its own consistency. Gödel developed for this the first versions of what would later be Computability Theory: “whose axioms can be known” means computable, in a similar sense to the modern one.

The repercussions in the mathematical world are colossal. Hilbert's program is down, and mathematics will never have a fully satisfactory foundation.

Even today, we do not know whether the ZF system, which axiomatizes all mathematics, is consistent: and for good reason, if, as we hope, it is indeed free from paradox, we will never be able to demonstrate it mathematically, or by anyway as long as one confines oneself to the axioms of ZF. The epistemological impact is considerable. Mathematics, mother of the exact sciences, is not only dependent on a *belief*, but is also able to demonstrate that it will always be so!



Kurt Gödel, 1906–1978

## 2. First-order logic

From Frege and Russell, we will retain the modern logical language used in mathematics, from Hilbert we will retain a proof system based on axioms and deduction rules applicable to mathematical statements, and from Zermelo, we will retain the axiomatic system ZF or ZFC, sufficient to formalize all of traditional mathematics. We now present without going into too much detail the basic principles of first-order logic, our objective being to present in a more precise way Gödel's theorem and its consequences. For this reason, the common thread of our presentation will be the specific example of Peano arithmetic.

### 2.1. Arithmetic language

The first step to formalize our mathematical demonstrations is to fix the language. We will therefore define the language of Peano arithmetic.

**Definition 2.1.** The language  $\mathcal{L}_{\text{PA}}$  of Peano arithmetic includes symbols specific to predicate calculus:

- (1) *Variable* symbols  $x, y, z, \dots$ : they represent natural numbers.
- (2) Parentheses  $()$  and symbols of *logical connectors*:  $\wedge, \vee, \rightarrow, \neg$ .
- (3) *Quantifier* symbols:  $\forall, \exists$ .

And those specific to Peano arithmetic:

- (1) The following symbols of *binary functions*:  $+, \times$ .
- (2) The following symbols of *binary relations*:  $=, <$ .
- (3) *Constant* symbols  $\dot{0}, \dot{1}$ . ◇

A language is nothing but a list of symbols. However, these symbols are intended to be used with a specific meaning. Regarding predicate calculus, this is the usual meaning: for example “ $\wedge$ ” is the logical *and*, while “ $\exists$ ” is the existential quantification. Regarding the symbols specific to Peano arithmetic, there are first the functions  $+$  and  $\times$  which respectively represent addition and multiplication, the symbols of equality and inequality, which have their usual meanings on integers, and finally the constants  $\dot{0}, \dot{1}$  which each represents the corresponding respective integer.

### First-order languages

It is easy to see how to generalize the previous definition to obtain other languages. The symbols specific to predicate calculus are the same for all first-order languages, to which we add an arbitrary number of function symbols ( $n$ -ary for arbitrary integers  $n$ ), an arbitrary number of symbols of relations (also  $n$ -ary for arbitrary integers  $n$ ) and an arbitrary number of constant symbols.

The function symbols are subject to arrangement rules to form what are called the *terms* of the language:

**Definition 2.2.** The *terms* of predicate calculus for arithmetic are inductively defined as follows.

- (1) A variable or a constant symbol is a term.
- (2) If  $t_1, t_2$  are terms, then  $(t_1 + t_2)$  and  $(t_1 \times t_2)$  are terms. ◇

**Example 2.3.** The following expressions are terms:  $x$ ,  $((((x + \dot{1}) + \dot{1}) + \dot{1}) + \dot{1} + \dot{1})$ ,  $(\dot{1} + \dot{0})$ ,  $(x + (y \times z))$

As we can see, the language of arithmetic is quite minimalist, and the valid expressions are very structured to remove any ambiguity. In practice, a number of notation shortcuts will be used to improve readability, as long as the translation in valid terms is unambiguous. For example,  $t_0 + t_1 + t_2$  is a shorthand for  $((t_0 + t_1) + t_2)$ . Likewise,  $x + \dot{3}$  is a shorthand for  $(x + ((\dot{1} + \dot{1}) + \dot{1}))$ .

**Definition 2.4.** A term is *closed* if it does not contain any variable and therefore only constants and the operations  $+$  and  $\times$ . ◇

Intuitively, a closed term in arithmetic is a way of representing a natural number. For example,  $(\dot{1} + \dot{1}) \times \dot{0}$  is a name for the integer 0. The integers each have an infinite number of names.

**Example 2.5.** The term  $x + \dot{1}$  is not closed, unlike  $(\dot{1} + \dot{1}) \times \dot{0}$ .

If the function symbols are used to create the terms of the language, the relation symbols are used to create the *formulas* of the language:

**Definition 2.6.** The *arithmetic formulas* are defined as follows:

- (1) For all terms  $t_1, t_2$ , then  $t_1 = t_2$  and  $t_1 < t_2$  are formulas. These formulas are called *atomic formulas*. The atomic formulas and their

negations, here,  $\neg t_1 = t_2$  and  $\neg t_1 < t_2$ , are called *literals*.

- (2) For all formulas  $F_1, F_2$ , then  $(F_1 \wedge F_2), (F_1 \vee F_2), (F_1 \rightarrow F_2)$  and  $\neg F_1$  are formulas.
- (3) For any formula  $F$ , then  $\forall x F$  and  $\exists x F$  are formulas.  $\diamond$

There again, we will resort to syntactic sugar by writing  $t_1 \leq t_2$  for the formula  $(t_1 < t_2) \vee (t_1 = t_2)$  and  $F_1 \leftrightarrow F_2$  for the formula  $(F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$ .

### First-order formulas

Here as well, we can easily generalize the formation of formulas and terms in any language: the function symbols of the language are used to create the terms, which then serve with the help of the relation symbols to create atomic formulas, which can then be composed between them with the help of the symbols of predicate calculus as in (2) and (3) of the previous definition.

With the quantifiers appear the notions of *bounded* and *free* variables: the *bounded* variables are unsurprisingly those which are bound to a quantifier, and the free variables are those which are not. The formal definition is quite heavy, but a few examples are enough to create an intuition.

**Example 2.7.** In the following formula: “ $\forall x \exists y y = x + 1$ ” the variables  $x$  and  $y$  are bounded while in the following formula: “ $\exists y y = x + 1$ ” only the variable  $y$  is bounded, unlike the variable  $x$  which is free.

**Definition 2.8.** A *closed formula* or a *statement* is a formula in which no variable is free.  $\diamond$

### Notation

Given a formula  $F$  having for free variables  $x_1, \dots, x_n$ , we will write  $F(x_1, \dots, x_n)$  to signify that the free variables of  $F$  are  $x_1, \dots, x_n$ .

Intuitively, a closed formula is an affirmation which will have a truth value (true or false) when evaluated on integers. Formulas with free variables define predicates on integers.

## 2.2. Hilbert-style deduction systems

In order to mathematically formalize the notion of proof, Hilbert imagined a very precise system of rules, which suffice to show “everything that is demonstrable”. How do we know? This idea will be made precise with

Gödel's completeness theorem to come. Subsequently, many other demonstration systems were developed, all equivalent and more or less suited to certain objectives.

### 2.2.1. Axioms and rules

In a Hilbert-style system, a proof is a finite list of mathematical sentences  $F_0, F_1, F_2, \dots, F_n$ —formulas in the considered language—satisfying the following rules: for any  $i \leq n$ , either  $F_i$  is an axiom, or  $F_i$  is produced from inference rules applied to formulas  $F_{j_1}, \dots, F_{j_m}$  for  $j_1, \dots, j_m < i$ . Each sentence  $F_i$  in this list will then be demonstrated, the objective being normally to obtain  $F_n$ , the last of them. Let us now see a specific example of a system à la Hilbert powerful enough to demonstrate all that is demonstrable.

**Axioms:** The axioms that we can always use are the tautologies of first-order logic. Thus, for example  $A \vee \neg A$  could be used as an axiom. There are three types:

1. The tautologies of propositional logic. For example  $(F \rightarrow G) \rightarrow (\neg G \rightarrow \neg F)$  is a tautology of propositional logic: it will be true for any formula  $F$  or  $G$  regardless of their truth value.
2. The tautologies of predicate calculus. In practice, only four axiom schemes are necessary:
  - (a)  $\forall x(F \rightarrow G) \rightarrow (F \rightarrow \forall xG)$  for any formula  $F$  not containing the variable  $x$ , and any formula  $G$ .
  - (b)  $\exists x(F \rightarrow G) \rightarrow (\exists xF \rightarrow G)$  for any formula  $F$ , and any formula  $G$  not containing the variable  $x$ .
  - (c)  $\forall xF \rightarrow F_{t/x}$  for any term  $t$  and any formula  $F$  containing no variable of  $t$ .
  - (d)  $F_{t/x} \rightarrow \exists xF$  for any term  $t$  and any formula  $F$  containing no variable of  $t$ .

Above  $F_{t/x}$  denotes the formula  $F$  for which each occurrence of  $x$  is replaced by the term  $t$ .

3. The axioms of equality:
  - (e)  $t = t$  for any term  $t$ .
  - (f)  $t_1 = q_1 \wedge \dots \wedge t_n = q_n \rightarrow f(t_1, \dots, t_n) = f(q_1, \dots, q_n)$  for all  $n$ , all terms  $(t_i)_{1 \leq i \leq n}, (q_i)_{1 \leq i \leq n}$  and any  $n$ -ary function symbol  $f$ .
  - (g)  $t = q \rightarrow (F(t/z) \rightarrow F(q/z))$  for any terms  $t, q$  and any formula  $F(z)$  not involving variables of  $t$  or  $q$ .

Above,  $F(t/z)$  and  $F(q/z)$  denote the formula  $F$  in which each occurrence of  $z$  is replaced by  $t$  and  $q$ , respectively.

Note that these axiom schemes depend on the language considered, each language using symbols of functions and relations which are specific to them, to construct the atomic terms and formulas respectively.

### Equality symbol

We consider here that the symbol of equality is necessarily part of the language that we use, and will always have its usual meaning, which justifies the axioms of equality mentioned above.

**Inference rules.** Inference rules allow us to combine sentences already demonstrated in our list, to obtain new ones. The following two rules are sufficient.

1. *Rule 1: Modus Ponens* - the basis of all deductive reasoning. If  $A \rightarrow B$  is proved and if  $A$  is proved, then we can deduce  $B$ .
2. *Rule 2: Generalization*. If  $F(x)$  is proved for a variable  $x$  free in  $F$ , then can deduce  $\forall xF(x)$ . This rule is widely used in mathematics: if we want to prove for example that for all rationals  $x < y$ , there exists a rational  $z$  such that  $x < z < y$ , we start by fixing rational variables  $x, y$  on which we assume nothing other than  $x < y$ . If we manage to deduce the existence of a rational  $z$  such that  $x < z < y$ , without using any specific property of  $x, y$ , we deduce by the generalization rule that for all rational  $x < y$ , there exists a rational  $z$  such that  $x < z < y$ .

This concludes the description of our system à la Hilbert. Let's see an example of a demonstration right away.

**Example 2.9.** Let us show  $\forall xF(x) \rightarrow \exists xF(x)$ . For more readability we will note  $A \equiv \forall xF(x)$ ,  $B \equiv F(y)$  and  $C \equiv \exists xF(x)$ .

- (1)  $A \rightarrow B$  (axiom (c)).
- (2)  $B \rightarrow C$  (axiom (d)).
- (3)  $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow ((A \rightarrow B) \wedge (B \rightarrow C)))$  (tautology).
- (4)  $(B \rightarrow C) \rightarrow ((A \rightarrow B) \wedge (B \rightarrow C))$  (Modus Ponens on (1) and (3)).
- (5)  $(A \rightarrow B) \wedge (B \rightarrow C)$  (Modus Ponens on (2) and (4)).
- (6)  $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$  (tautology).

(7)  $A \rightarrow C$  (Modus Ponens on (5) and (6)).

### — Quid of empty universes? —

The reader may be surprised by the sentence  $\forall x F(x) \rightarrow \exists x F(x)$ —that we have demonstrated. What happens if we place ourselves in an empty universe? At this time,  $\forall x F(x)$  is correct, but not  $\exists x F(x)$ . The axiom (d) above effectively implies that the demonstrated formulas will only be valid if there is at least one element in our universe. The notion of universe will be made precise in Section 2.4 to come. It is in fact necessary to have such a restriction if one wants to demonstrate that any formula is equivalent to a formula in prenex form (see Definition 2.12).

#### 2.2.2. First tools

We claim that the Hilbert-style system described above allows us to show everything that is demonstrable, and we will see this formally with the completeness theorem to come. The system is intentionally minimalist, and difficult to handle as it is. The reader can for example try to demonstrate  $\neg \forall x F(x) \rightarrow \exists x \neg F(x)$  to realize the difficulty of using the system as it is. The mathematician who wants to do this kind of proof will proceed naturally as for any proof: assuming that  $\neg \forall x F(x)$  is true, and trying to deduce  $\exists x \neg F(x)$ . The problem is that if we want to respect the formalism of a Hilbert system,  $\neg \forall x F(x)$  is not necessarily an axiom that we can assume to be true in order to derive a conclusion. We then see our first fundamental tool, which will allow us to proceed as we are used to.

**Lemma 2.10 (Deduction lemma).** Let  $F$  be a closed formula. If we can prove  $G$  using  $F$  as an axiom, then there exists a proof of  $F \rightarrow G$  (which does not use  $F$  as an axiom). ★

PROOF. Let  $G_1, \dots, G_n$  be a proof of  $G_n$  using  $F$  as an axiom. Let us show by induction on the size of a proof that we can do some insertions in the sequence  $F \rightarrow G_1, \dots, F \rightarrow G_n$  in order to make a valid proof of it not using  $F$  as an axiom. If  $G_i$  is the statement  $F$ , then  $F \rightarrow F$  is an axiom of propositional logic. If  $G_i$  is an axiom of propositional logic, then it is also the case for  $F \rightarrow G_i$ . If  $G_i$  is one of the axioms (a), (b), (c), (d) of predicate calculus, then  $G_i \rightarrow (F \rightarrow G_i)$  is an axiom of propositional logic. By using the Modus Ponens on  $G_i$  and  $G_i \rightarrow (F \rightarrow G_i)$ , we get  $F \rightarrow G_i$ . If  $G_i = \forall x G_j$  for  $j < i$  is obtained by the generalization rule, then  $\forall x (F \rightarrow G_j)$  is obtained from  $F \rightarrow G_j$  (which we have by induction hypothesis) by the generalization rule. We obtain  $F \rightarrow \forall x G_j$  by Modus Ponens and by axiom (a)

$$\forall x (F \rightarrow G_j) \rightarrow (F \rightarrow \forall x G_j).$$

Finally, if  $G_i = G_b$  is obtained by Modus Ponens on  $G_a$ , then  $G_a \rightarrow G_b$

for  $a, b < i$ . Then, we have  $F \rightarrow G_a$  and  $F \rightarrow (G_a \rightarrow G_b)$ , by induction hypothesis. The formula

$$(F \rightarrow (G_a \rightarrow G_b)) \rightarrow ((F \rightarrow G_a) \rightarrow (F \rightarrow G_b))$$

is a tautology of predicate calculus. By Modus Ponens, we deduce  $(F \rightarrow G_a) \rightarrow (F \rightarrow G_b)$  and, by a second application of Modus Ponens, we deduce  $F \rightarrow G_b$ . ■

Let us see immediately an example of application of the deduction lemma to prove  $\neg\forall x F(x) \rightarrow \exists x \neg F(x)$ .

**Example 2.11.** Let us show  $\neg\exists x F(x) \rightarrow \forall x \neg F(x)$ . From the deduction lemma, we can assume  $\neg\exists x F(x)$  as an axiom.

- (1)  $\neg\exists x F(x)$  (axiom).
- (2)  $F(x) \rightarrow \exists x F(x)$  (axiom (d)).
- (3)  $(F(x) \rightarrow \exists x F(x)) \rightarrow (\neg\exists x F(x) \rightarrow \neg F(x))$  (tautology).
- (4)  $\neg\exists x F(x) \rightarrow \neg F(x)$  (Modus Ponens on (2) and (3)).
- (5)  $\neg F(x)$  (Modus Ponens on (1) and (4)).
- (6)  $\forall x \neg F(x)$  (generalization on (5)).

Let us now show  $\forall x \neg\neg F(x) \rightarrow \forall x F(x)$ .

- (1)  $\forall x \neg\neg F(x)$  (axiom).
- (2)  $\forall x \neg\neg F(x) \rightarrow \neg\neg F(x)$  (axiom (c)).
- (3)  $\neg\neg F(x)$  (Modus Ponens on (1) and (2)).
- (4)  $\neg\neg F(x) \rightarrow F(x)$  (tautology).
- (5)  $F(x)$  (Modus Ponens on (3) and (4)).
- (6)  $\forall x F(x)$  (generalization on (5)).

We leave it to the reader to use the contrapositive to deduce

$$\neg\forall x F(x) \rightarrow \exists x \neg F(x).$$

### 2.2.3. Prenex form

This proof system allows us to show that any formula — in any language — is provably equivalent to a formula in *prenex form*.

**Definition 2.12.** A formula is in *prenex form* if it is of the form

$$Q_1 x_1 \dots Q_n x_n F(x_1, \dots, x_n, y_1, \dots, y_m)$$

where each  $Q_i$  is a  $\forall$  or  $\exists$  quantifier and  $F(x_1, \dots, x_n, y_1, \dots, y_m)$  is a quantifier-free formula.  $\diamond$

We leave it to the reader to show the following equivalences:

- $\forall x F \wedge G \equiv \forall x (F \wedge G)$ ;
- $\forall x F \vee G \equiv \forall x (F \vee G)$ ;
- $\exists x F \wedge G \equiv \exists x (F \wedge G)$ ;
- $\exists x F \vee G \equiv \exists x (F \vee G)$ .

These equivalences, coupled with Example 2.11 allow to transform any formula into a provably equivalent prenex formula in our system of deduction, by gradually shifting the quantifiers to the left.

Note that the equivalences above only hold if one places oneself in a universe having at least one element. Then, for example, we will have  $(\forall x x = x) \wedge (\exists y y \neq y)$  false in the empty universe, but  $\forall x (x = x \wedge (\exists y y \neq y))$  always true.

### 2.3. Logical and arithmetic theories of Peano

Once a language fixed — in our case, that of arithmetic — and the proof system specified, we can then consider a *mathematical theory* in this language, and use it to prove theorems concerning the structure described by this theory.

**Definition 2.13.** A *theory*  $T$  in a language  $\mathcal{L}$  is a collection of closed formulas of that language. We also often use the term *axiomatic system* or more simply *system* to denote a theory.  $\diamond$

The theory is then seen as a list of axioms, which we can use in our proofs, in addition to the axioms present in the proof system. Let us see immediately the axioms of arithmetic which were developed by Peano towards the end of the 19th century.

#### 2.3.1. Axioms of Peano arithmetic

Peano axioms allow us to specify the behavior of natural numbers. The first series of axioms defines the behavior of integers with respect to the successor.

- (1)  $\forall x \neg(x + \dot{1} = \dot{0})$ : 0 has no predecessor.

- (2)  $\forall x (x = \dot{0} \vee \exists y (x = y + \dot{1}))$ : Any integer other than 0 has a predecessor.
- (3)  $\forall x \forall y (x + \dot{1} = y + \dot{1} \rightarrow x = y)$ : The successor function for integers is injective.

The following axioms give rules for computing addition and multiplication:

- (4)  $\forall x (x + \dot{0} = x)$ ;
- (5)  $\forall x \forall y (x + (y + \dot{1}) = (x + y) + \dot{1})$ ;
- (6)  $\forall x (x \times \dot{0} = \dot{0})$ ;
- (7)  $\forall x \forall y (x \times (y + \dot{1}) = (x \times y) + x)$ .

Finally, we define the behavior of integers with respect to order:

- (8)  $\forall x \forall y (x < y \leftrightarrow (\exists z (z \neq \dot{0} \wedge x + z = y)))$

### Notation

We denote by **Q** the theory composed of axioms (1) - (8), which form what we call *Robinson arithmetic*.

To obtain Peano arithmetic, we add the following axiom, for any arithmetic formula  $F(x)$ :

- (9)  $(F(\dot{0}) \wedge (\forall x (F(x) \rightarrow F(x + \dot{1})))) \rightarrow \forall x F(x)$

Note that axiom (9) is not a unique axiom. As for axioms (a), (b), (c), (d) of our proof system, it is an *axiom scheme*, that is to say of an infinity of axioms parameterized by a formula, here  $F(x)$ .

Statement (9) is the well-known axiom of induction on integers: if a formula  $F$  is true for the integer 0, and if the fact that it is true for  $n$  implies that it is true for  $n + 1$ , then it is true for all integer  $n$ .

### Notation

We denote by **PA** the theory composed of **Q** and the axiom scheme (9) for any formula of arithmetic. It is known as *Peano arithmetic*.

We will see in Chapter 23 how to use the axioms of PA to show some elementary facts about natural numbers. We will see in particular that the induction scheme is equivalent to the following scheme: for any formula  $F$  true for at least one integer, there exists a smaller integer  $x$  such that  $F(x)$  is true. This may of course seem perfectly obvious, because we have in mind the structure of the natural numbers  $\mathbb{N}$  that we know well, but a proof does not use this structure: it uses only the axioms, and in the case of the

arithmetic, these are precisely made so that any mathematical structure verifying them behaves like the natural numbers. This will bring us to the notion of *model*, in the next section.

### 2.3.2. Proofs in a theory

Once we have fixed a theory, we can use its axioms within a Hilbert system to prove mathematical statements.

#### Notation

We will note  $T \vdash F$  to signify that there is a proof of the formula  $F$  from the axioms of  $T$  via the Hilbert system exposed in Section 2.2. If there is no such proof, then we write  $T \nvdash F$ .

Among the tautologies of propositional logic, we find for all formulas  $F, G$  the formula  $(F \wedge \neg F) \rightarrow G$ , called “ex falso quodlibet”, meaning that from a contradiction  $(F \wedge \neg F)$  we can deduce anything. It follows that if a theory proves a formula and its opposite, any statement is provable in this theory, which removes all interest from it. We will therefore expect above all from a theory that it be *consistent*.

**Definition 2.14.** A theory  $T$  is *consistent* if there is no formula  $F$  such that  $T \vdash F \wedge \neg F$ . ◇

#### Notation

We will write  $T \vdash \perp$  to mean  $T \vdash F \wedge \neg F$  for a certain formula  $F$ . The notation  $T \nvdash \perp$  then logically signifying that for any formula  $F$  we have  $T \nvdash F \wedge \neg F$ . As the equality symbol is part of our language, we can consider without loss of generality  $\perp$  as being  $\neg x = x$ . Since  $x = x$  is an axiom, if  $T \vdash \neg x = x$  then  $T \vdash x = x \wedge \neg x = x$ .

We saw in the introduction to our interlude that the inconsistency Russell found in Frege’s work was a cornerstone in the crisis of foundations. So mathematicians would like as much as possible to be certain that they are working only with consistent theories. But how can we verify the consistency of a theory? Without even talking about theory, what about the consistency of the Hilbert system itself and its axioms of logic, presented in Section 2.2? The notion of *model* answers these questions.

## 2.4. Structures, models and consistency theorem

The notion of structure can be defined very generally for any fixed language. Let us start with the example which interests us more specifically, namely the language of arithmetic.

**Definition 2.15.** A *structure*  $\mathcal{M} = (M, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, =^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  in  $\mathcal{L}_{\text{PA}}$  is given by:

- A non-empty set  $M$
- $+^{\mathcal{M}}, \times^{\mathcal{M}} : M \times M \rightarrow M$  functions corresponding to  $+$ ,  $\times$  function symbols.
- A relation  $<^{\mathcal{M}} \subseteq M \times M$  corresponding to the relation symbol  $<$ .
- A relation  $=^{\mathcal{M}} \subseteq M \times M$  corresponding to the relation symbol  $=$ , and which corresponds to “true equality”, i.e., such that  $(x, y) \in =^{\mathcal{M}} \leftrightarrow x = y$ .
- Elements  $0^{\mathcal{M}}, 1^{\mathcal{M}} \in M$  corresponding to the constant symbols  $\dot{0}$  and  $\dot{1}$ . ◇

A structure is therefore a set, as well as functions, relations and constants on this set, constituting an interpretation of the symbols of language.

By abuse of notation, we will sometimes identify  $\mathcal{M}$  with its *underlying set*  $M$  (we write for example  $x \in \mathcal{M}$ ). To simplify the notations, we will sometimes remove the exponents  $^{\mathcal{M}}$  when it is clear that we are talking about the functions and relations of the structure and not of symbols of the language.

### First-order structure

It is easy to see how to generalize the previous definition to obtain structures for any first-order language. We always have a non-empty set  $M$ . Each  $n$ -ary function symbol  $f$  corresponds to a function of  $f^{\mathcal{M}} : M^n \rightarrow M$ , each  $n$ -ary relation symbol  $R$  corresponds to a relation  $R^{\mathcal{M}} \subseteq M^n$  and each constant symbol  $c$  corresponds to a constant  $c^{\mathcal{M}} \in M$ . The equality relation will always be present and will always correspond to “the true equality”.

A formula, like for example  $F(x) = \exists y \, y \times (\dot{1} + \dot{1}) = x$  is only a sequence of symbols. Once a structure  $\mathcal{M}$  has been fixed, each symbol is intended to be interpreted by the object which corresponds to it in  $\mathcal{M}$ ; moreover, the free variables can also be replaced by various *parameters* —that is to say various elements— of the structure.

**Definition 2.16 (Parametric formulas).** Let  $\mathcal{L}$  be a language and  $\mathcal{M}$  a structure for  $\mathcal{L}$ . Given a formula  $F(x_1, \dots, x_n)$  of  $\mathcal{L}$  having  $x_1, \dots, x_n$  as free variables, and given  $a_1, \dots, a_n \in \mathcal{M}$ , the expression  $F(a_1, \dots, a_n)$  denotes a formula *parameterized* by  $a_1, \dots, a_n$ : this is the formula  $F$

in which each free occurrence of  $x_i$  is replaced by  $a_i$  for  $1 \leq i \leq n$ . A parametric formula without free variable will be a *closed parametric formula*.  $\diamond$

A closed parametric formula is no longer a simple sequence of symbols, but a statement which will be *true* or *false* in the considered structure.

### Remark

Note that the notion of parametric formula induces that of parametric term. Thus, in the usual structure of integers for the language of arithmetic,  $(5 + 4) \times 2$  will be a term parameterized by the elements 5, 4 and 2 of our structure. This should not be confused with the closed term  $(\dot{5} + \dot{4}) \times \dot{2}$  with no parameter.

We now define satisfaction in a structure, for closed formulas or else parameterized in this structure.

**Definition 2.17.** Let  $\mathcal{L}$  be a language and  $\mathcal{M} = (M, \dots)$  a structure for  $\mathcal{L}$ . We say that a formula  $F(x_1, \dots, x_n)$  of  $\mathcal{L}$  is *true* in  $\mathcal{M}$  for parameters  $a_1, \dots, a_n \in M$ , and we write  $\mathcal{M} \models F(a_1, \dots, a_n)$ , if  $F(a_1, \dots, a_n)$  is actually satisfied in the structure. The definition is formally done by induction on the formulas (in what follows  $\bar{x}$  and  $\bar{a}$  are shortcuts for  $x_1, \dots, x_n$  and  $a_1, \dots, a_n$ ):

- Base case:  $\mathcal{M} \models R(t_1(\bar{a}), \dots, t_m(\bar{a}))$  where  $R$  is an  $m$ -ary relation symbol of the language corresponding to the relation  $R^{\mathcal{M}} \subseteq M^m$  and each  $t_i(\bar{x})$  is a term, iff  $(t_1^{\mathcal{M}}(\bar{a}), \dots, t_m^{\mathcal{M}}(\bar{a})) \in R^{\mathcal{M}}$  where each  $t_i^{\mathcal{M}}(\bar{a})$  is the element of  $M$  obtained by applying the functions corresponding to each function symbol used in  $t_i(\bar{a})$ .
- Universal quantification:  $\mathcal{M} \models \forall y G(y, \bar{a})$  iff  $\mathcal{M} \models G(b, \bar{a})$  for all  $b \in M$ .
- Existential quantification:  $\mathcal{M} \models \exists y G(y, \bar{a})$  iff there is  $b \in M$  such that  $\mathcal{M} \models G(b, \bar{a})$ .
- Negation:  $\mathcal{M} \models \neg G(\bar{a})$  iff  $\mathcal{M} \not\models G(\bar{a})$ .
- Conjunction:  $\mathcal{M} \models G_1(\bar{a}) \wedge G_2(\bar{a})$  iff  $\mathcal{M} \models G_1(\bar{a})$  and  $\mathcal{M} \models G_2(\bar{a})$ .
- Disjunction:  $\mathcal{M} \models G_1(\bar{a}) \vee G_2(\bar{a})$  iff  $\mathcal{M} \models G_1(\bar{a})$  or  $\mathcal{M} \models G_2(\bar{a})$ .

The satisfaction of implication is deduced from that of negation and disjunction.  $\diamond$

Once a theory is fixed, we can consider models of this theory, that is, structures within which each axiom of the theory will be true.

**Definition 2.18.** Let  $T$  be a theory in a language  $\mathcal{L}$ . A structure  $\mathcal{M}$  in  $\mathcal{L}$  is a *model* of  $T$  if every axiom of  $T$  is true in  $\mathcal{M}$ .  $\diamond$

**Example 2.19.**

- The set  $\mathbb{Z}$  equipped with the usual operations is not a model of Peano arithmetic because it does not satisfy axiom (1): 0 has no predecessor.
- The set  $2\mathbb{N}$  of even integers where  $\dot{0}$  is interpreted by 0 and  $\dot{1}$  is interpreted by 2 is also not a model of the arithmetic of Peano because it does not verify axiom (7):  $2 \times (2 + 2) \neq (2 \times 2) + 2$ .
- The model par excellence of Peano arithmetic is of course that of natural numbers  $\mathbb{N}$ , equipped with the usual operations and relations.

While theories form the *syntactic* aspect of mathematics, the models form the *semantic* aspect. There are many advantages to thinking about models of a theory.

In the first place, models are what mathematicians have naturally worked with since the beginning. Today mathematics is so advanced in abstraction that it is an aspect of things that we sometimes lose sight of, but this science is not initially that far from physics, in the sense that it is first of all an *observation* work of the reality of certain phenomena in order to extract the logical laws which govern them. Every mathematician knows from experience that he does not decide the truth, which sometimes resides well hidden in the abstract structures studied, in other words in the models. This methodology of observation and research based on logic gives its universal and transcendent character to mathematical truth, constituting in a way the glue which binds the community of mathematicians. If syntax is of course important, because it constitutes the language allowing mathematics to be communicated, *semantics precedes syntax*<sup>2</sup>: it is from there that intuitions start and about it that the theorems bear.

Finally, there is a more prosaic advantage to the study of models: in essence, if a closed formula  $F$  is true in a model, then its negation  $\neg F$  cannot be true there. A model is always a consistent and complete structure: each closed formula is either true or false, and no formula can be true at the same time as its negation. We can use this to show the consistency theorem.

---

<sup>2</sup>Aphorism dear to Professor René Cori, who taught the authors of this book the principles of present chapter.

**Theorem 2.20 (Soundness theorem)**

*Let  $T$  be a theory and  $F(x_1, \dots, x_n)$  a formula in the language of that theory. If  $T \vdash F(x_1, \dots, x_n)$  then any model of  $T$  is also a model of  $\forall x_1 \dots \forall x_n F(x_1, \dots, x_n)$ .*

PROOF. The proof is easily done by induction on the size of a proof. Suppose this is the case for any proof  $G_1, \dots, G_n$  in  $T$ . Let  $G_1, \dots, G_{n+1}$  be a proof in  $T$ . By induction hypothesis any model of  $T$  is a model of each formula  $\forall \bar{x} G_i$  for  $i < n+1$  where the notation  $\forall \bar{x} G_i$  means that we universally quantify on each free variable of  $G_i$  (when there is). Let  $\mathcal{M}$  be a model of  $T$ . If  $G_{n+1}$  is an axiom of  $T$ , then it is a closed formula and  $\forall \bar{x} G_{n+1}$  is obviously true in  $\mathcal{M}$ . If  $G_{n+1}$  is an axiom of equality (of type (e) (f) or (g) above), then  $\forall \bar{x} G_{n+1}$  is true in  $\mathcal{M}$  by the fact that the equality of  $\mathcal{M}$  is always true equality.

If  $G_{n+1}$  is a tautology of propositional logic or an axiom of type (a) (b) (c) or (d) of predicate calculus, then  $\forall \bar{x} G_{n+1}$  is true by definition of the satisfaction in a model (the details are left to the reader, note that for (d) we use the fact that the model is not empty).

If  $G_{n+1}$  is obtained by generalization on  $G_i$  for  $i < n+1$  — in particular  $G_{n+1}$  is of the form  $\forall y G_i$  — then  $\mathcal{M}$  satisfying  $\forall \bar{x} G_i$ , it also satisfies  $\forall \bar{x} G_{n+1}$  (which is in fact the same formula). Finally if  $G_{n+1}$  is obtained by Modus Ponens via  $G_i \rightarrow G_{n+1}$  and  $G_i$  for  $i < n+1$  then  $\mathcal{M}$  satisfies  $\forall \bar{x} G_i$  and  $\forall \bar{x} (G_i \rightarrow G_{n+1})$ . By definition of satisfaction,  $\mathcal{M}$  thus satisfies  $\forall \bar{x} G_{n+1}$ . ■

The previous theorem can be summarized as follows: “one can only prove true things”. This is good news, from which we can deduce our consistency theorem:

**Corollary 2.21 (Consistency theorem)**

*If an axiomatic system admits a model, then it is consistent.*

PROOF. This is shown by contraposition. If  $T \vdash F \wedge \neg F$  for a closed formula  $F$ , then any model of  $T$  is a model of  $F \wedge \neg F$ . Since there is no model for  $F \wedge \neg F$  then  $T$  has no model. ■

We easily verify the existence of a model of the axioms of logic: the set  $\{1\}$  with the relation of equality  $1 = 1$ . This shows that the axioms of logic are consistent and cannot prove  $\perp$ .

We also easily verify the existence of a model for Peano arithmetic, namely  $\mathbb{N}$  equipped with the usual functions of addition and multiplication, as well

as the usual relations  $<$  and  $=$  on integers. This is the second good news, the axioms of Peano arithmetic are also consistent. We will examine the meaning of this statement a little further on, notably in the light of Gödel's second incompleteness theorem and its implications.

## 2.5. Models and completeness theorem

If the system of deduction à la Hilbert that we have given, with its axioms of logic and its rules of deduction is indeed sound, how on the other hand know that it is sufficiently powerful? After all, the two inference rules seem to form a very poor working tool. Can we really demonstrate everything with this system? We will see that this is the case, via a theorem demonstrated by Gödel in his doctoral thesis which can be seen as the converse of the consistency theorem: everything that is universally true is demonstrable.

### **Theorem 2.22 (Théorème de complétude de Gödel)**

*Let  $T$  be a theory in a countable language. If  $T$  is consistent, then  $T$  has a model.*

Note that the completeness theorem for uncountable languages can also be demonstrated using the axiom of choice. We only show a countable version which will be more than enough for us. Gödel's completeness theorem is more difficult to prove than the consistency theorem which is a simple routine check. It is a question here of building a model of a theory  $T$ , from the simple fact that  $T \not\vdash \perp$ . The idea of the proof passes by the creation of a *complete* theory.

**Definition 2.23.** A theory  $T$  is called *complete* if  $T \vdash F$  or  $T \vdash \neg F$  for any closed formula  $F$  in the language of  $T$ . ◇

**Proposition 2.24.** Let  $\mathcal{L}$  be a countable language. Any consistent theory of  $\mathcal{L}$  can be extended into a complete and consistent theory. ★

The proof of the above proposition uses Lemma 2.10 which we reformulate here with the notation  $\vdash$  introduced since:

**Lemma (2.10).** Let  $T \cup \{F\}$  be a theory and  $G$  a formula. Suppose  $T \cup \{F\} \vdash G$ . Then,  $T \vdash F \rightarrow G$ . ★

**PREUVE DE PROPOSITION 2.24.** Given a consistent theory  $T$  in a countable language, we inductively construct a complete and consistent extension  $T'$ . Let  $T_0 = T$ . At step  $n$ , suppose that a consistent theory  $T_n$  is defined. Let  $F_n$  be the  $n$ -th closed formula of our language. If  $T_n \vdash F_n$  we define  $T_{n+1} = T_n \cup \{F_n\}$ . If  $T_n \vdash \neg F_n$  we define  $T_{n+1} = T_n \cup \{\neg F_n\}$ . In

these first two cases, the consistency of  $T_{n+1}$  follows from the consistency of  $T_n$  and from the deduction lemma.

If ever  $T_n$  proves neither  $F_n$  nor  $\neg F_n$ , then we define  $T_{n+1} = T_n \cup \{F_n\}$ . Let us assume by the absurdity  $T_n \cup \{F_n\} \vdash \perp$ . Then, according to the deduction lemma  $T_n \vdash F_n \rightarrow \perp$  and therefore  $T_n \vdash \neg F_n$  by contraposition and Modus Ponens, which contradicts the fact that  $T$  does not prove  $\neg F_n$ . Note that we could just as well define  $T_{n+1} = T_n \cup \{\neg F_n\}$ .

Finally, we let  $T' = \bigcup_n T_n$ . The consistency of  $T'$  then comes from the fact that a proof in a theory uses only a finite number of its axioms: if  $T' \vdash \perp$  then there necessarily exists  $n$  such that  $T_n \vdash \perp$ . Since each theory  $T_n$  is consistent, then  $T'$  must be consistent. ■

**PROOF OF THE COMPLETENESS THEOREM.** The proof that we give, is due to Leon Henkin [84], and rests on the creation of a complete, consistent theory, and having what one calls *Henkin witnesses*. We will first show the following statement.

“Let  $T$  be a theory in a language  $\mathcal{L}$  and  $c$  a constant symbol which does not appear in  $\mathcal{L}$ . Suppose  $T \cup \{\exists x F(x) \rightarrow F(c)\} \vdash \perp$ . Then,  $T \vdash \perp$ .”

As  $T \cup \{\exists x F(x) \rightarrow F(c)\} \vdash \perp$  then by the deduction lemma, contraposition and Modus Ponens we have  $T \vdash \exists x F(x) \wedge \neg F(c)$ . In particular  $T \vdash \neg F(c)$ . Let now be a variable  $z$  which does not intervene in the proof of  $\neg F(c)$ . As the constant  $c$  does not appear in the theory  $T$  it can only be introduced into the proof by an axiom of logic. Each of these axioms remains valid by replacing  $c$  by  $z$ . We leave it to the reader to check that if we replace  $c$  by  $z$  for each step of the proof, we obtain a valid proof of  $\neg F(z)$ . By the generalization rule we finally get  $T \vdash \forall x \neg F(x)$ . As also  $T \vdash \exists x F(x)$  then  $T \vdash \perp$ .

Let us now proceed to the proof of the completeness theorem. Let  $T$  be a theory in a countable language  $\mathcal{L}$ . Let's show how to build a model. We define  $T_0 = T$  and  $\mathcal{L}_0 = \mathcal{L}$ . At step  $n \in \mathbb{N}$ , suppose that we have defined a consistent theory  $T_{2n}$  in a language  $\mathcal{L}_{2n}$ . Let  $\mathcal{L}_{2n+1}$  be the language  $\mathcal{L}_{2n}$  to which we add a new constant symbol  $c_G$  for any formula  $G$  of  $T_{2n}$  of the form  $\exists x F(x)$ . Let  $T_{2n+1}$  be the theory  $T_{2n}$  to which we add the statements  $\exists x F(x) \rightarrow F(c_G)$  for any statement  $G$  of  $T_{2n}$  of the form  $\exists x F(x)$ . Note that by the above statement, as  $T_{2n}$  is consistent, each time an axiom of the form  $\exists x F(x) \rightarrow F(c_G)$  is added, the theory remains consistent. So  $T_{2n+1}$  is consistent. Finally let  $\mathcal{L}_{2n+2} = \mathcal{L}_{2n+1}$  and using Proposition 2.24, let  $T_{2n+2}$  be the completion of  $T_{2n+1}$  for the language  $\mathcal{L}_{2n+2}$ . Let  $\mathcal{T}_\omega = \bigcup_n T_n$  and  $\mathcal{L}_\omega = \bigcup_n \mathcal{L}_n$ . Note that  $\mathcal{T}_\omega$  is a complete and consistent theory in the language  $\mathcal{L}_\omega$ , which also contains a statement of the form  $\exists x F(x) \rightarrow F(c)$  for each of the statements of the form  $\exists x F(x)$  of  $\mathcal{T}_\omega$ ,

where  $c$  is a constant symbol of  $\mathcal{L}_\omega$ . The famous *Henkin's witnesses* are the new constant symbols thus introduced.

The underlying set  $M$  of our model  $\mathcal{M}$  is the set of closed terms of  $\mathcal{L}_\omega$ , quotiented by the equality relation. Formally let  $(t_n)_{n \in \mathbb{N}}$  be the list of closed terms of  $\mathcal{L}_\omega$ . Then,  $M = \{t_n : \forall i < n \ (\neg t_i = t_n) \in T_\omega\}$ .

Note that the function symbols of  $\mathcal{L}$  have a clear interpretation in  $M$ . For example if  $f$  is a unary function symbol of  $\mathcal{L}$  then its corresponding function  $f^{\mathcal{M}} : M \rightarrow M$  is defined by  $f^{\mathcal{M}}(t) = q$  for  $q$  the element of  $M$  which is equal to closed term  $f(t) \in \mathcal{L}_\omega$ , that is to say such that  $(f(t) = q) \in T_\omega$ .

The theory  $T_\omega$  being complete and consistent, for any symbol of  $m$ -ary relation  $R$  of  $\mathcal{L}$  and any element  $t_1, \dots, t_m \in M$ , exactly one of the statements among  $R(t_1, \dots, t_m)$  or  $\neg R(t_1, \dots, t_m)$  is in  $T_\omega$ . This induces an interpretation  $R^{\mathcal{M}}$  of the relation symbol  $R$  of  $\mathcal{L}$  in  $\mathcal{M}$ .

It remains to show by induction on the formulas that all the statements of  $T_\omega$  are satisfied in  $\mathcal{M}$  (and therefore also those of  $T$ ). Without loss of generality, only the formulas in prenex form are treated, and we can assume that the negation symbol only appears in front of the atomic formulas. By definition of relations in  $\mathcal{M}$  this is indeed the case for atomic formulas and their negations. If  $F_1 \wedge F_2 \in T_\omega$  then as  $T_\omega$  is complete  $F_1, F_2 \in T_\omega$ . By induction hypothesis  $\mathcal{M} \models F_1$  and  $\mathcal{M} \models F_2$  therefore  $\mathcal{M} \models F_1 \wedge F_2$ . If  $F_1 \vee F_2 \in T_\omega$  then as  $T$  is complete  $F_1 \in T_\omega$  or  $F_2 \in T_\omega$  (otherwise by completeness  $\neg F_1, \neg F_2 \in T_\omega$  which contradicts  $F_1 \vee F_2$ ). By induction hypothesis  $\mathcal{M} \models F_1 \vee F_2$ . If  $\forall x F(x) \in T_\omega$  then as  $T_\omega$  is complete  $F(t)$  is in  $T_\omega$  for any closed term  $t$  of  $\mathcal{L}_\omega$  and therefore any element of  $M$ . By induction hypothesis  $\mathcal{M} \models F(t)$  for all  $t \in M$  and therefore  $\mathcal{M} \models \forall x F(x)$ . If  $\exists x F(x) \in T_\omega$  then  $\exists x F(x) \rightarrow F(c) \in T_\omega$  for a symbol with constant  $c \in \mathcal{L}_\omega$ . In particular  $F(c) \in T_\omega$  by Modus Ponens. Let  $t \in M$  be a closed term of  $\mathcal{L}_\omega$  such that  $(t = c) \in T_\omega$ . By the axioms of equality we have  $F(t)$ . By induction hypothesis  $\mathcal{M} \models F(t)$ . So  $\mathcal{M} \models \exists x F(x)$ . ■

The completeness theorem is often used in the form of the following corollary, which can be seen as a reverse of Theorem 2.20:

**Corollary 2.26**

*If every model of a theory  $T$  is a model of a formula  $F$ , then  $T \vdash F$ .*

PROOF. If we have  $T \cup \{\neg F\} \vdash \perp$  then  $T \vdash \neg F \rightarrow \perp$  by the deduction lemma and therefore  $T \vdash F$ .

Suppose now  $T \not\vdash F$ , then by the line above  $T \cup \{\neg F\} \not\vdash \perp$ . According to the completeness theorem there exists a model of  $T \cup \{\neg F\}$ , that is to say a model of  $T$  which is not a model of  $F$ . ■

The completeness theorem implies in particular that one cannot do better than the Hilbert system that we have presented: suppose that in this system the axioms of logic alone are not sufficient to prove a formula  $F$ , in other words  $\not\vdash F$  (from an empty theory). Suppose that a more powerful and consistent proof system exists such that  $\vdash^* F$ , where  $\vdash^*$  is the notion of proof in this system. So also,  $\vdash^* \neg F \rightarrow \perp$  and therefore  $\neg F \vdash^* \perp$ . Now as  $\not\vdash F$ , according to the completeness theorem there is a model of  $\neg F$  and according to the consistency theorem for  $\vdash^*$  our model is therefore a model of  $\perp$  which is impossible.

### 3. Incompleteness theorems of Gödel

We now get to the heart of the matter, via Gödel's incompleteness theorems, which are based, among other things, on a coding of computably enumerable sets by arithmetic formulas.

#### 3.1. Peano arithmetic formulas

We have defined in Chapter 5 a complexity hierarchy on sets, called *arithmetic hierarchy*. We are now going to give all its meaning to this name by defining a syntactic hierarchy of arithmetic formulas, which coincides with the arithmetic hierarchy, in the sense that a set  $A$  is  $\Sigma_n^0$  iff it is definable by a  $\Sigma_n$  formula of Peano arithmetic.

**Definition 3.1.** A formula of Peano arithmetic is  $\Delta_0$  if the quantifications it comprises are all bounded, that is to say of the form  $\exists x < t$  and  $\forall x < t$ , with  $t$  a term where the variable  $x$  does not appear freely. Note that the formulas  $\exists x < t F(x)$  and  $\forall x < t F(x)$  translate respectively to  $\exists x(x < t \wedge F(x))$  and  $\forall x(x < t \rightarrow F(x))$ .  $\diamond$

Given a  $\Delta_0$  formula of Peano arithmetic  $F(x_1, \dots, x_n)$ , the restriction on quantifications makes the set  $\{(x_1, \dots, x_n) \in \mathbb{N} : F(x_1, \dots, x_n)\}$  computable. This follows for example directly from the closure of primitive recursive predicates by conjunction, disjunction and bounded quantification (see Example 6-3.16 and Exercise 6-3.19). We define a complexity hierarchy on the formulas of Peano arithmetic analogous to the arithmetic hierarchy of Definition 5-1.1.

**Definition 3.2.**

1. A formula  $F(x_1, \dots, x_m)$  of Peano arithmetic is  $\Sigma_n$  if

$$F(x_1, \dots, x_m) = \overbrace{\exists y_1 \forall y_2 \dots Q y_n}^{n \text{ quantifiers}} G(x_1, \dots, x_m, y_1, \dots, y_n)$$

for a  $\Delta_0$  formula  $G(x_1, \dots, x_m, y_1, \dots, y_n)$ , where  $Q$  is  $\exists$  if  $n$  is odd, and  $\forall$  if  $n$  is even.

2. A formula  $F(x_1, \dots, x_m)$  of Peano arithmetic is  $\Pi_n$  if

$$F(x_1, \dots, x_m) = \overbrace{\forall y_1 \exists y_2 \dots Q y_n}^{n \text{ quantifiers}} G(x_1, \dots, x_m, y_1, \dots, y_n)$$

for a  $\Delta_0$  formula  $G(x_1, \dots, x_m, y_1, \dots, y_n)$ , where  $Q$  is  $\forall$  if  $n$  is odd, and  $\exists$  if  $n$  is even. ◇

We will see that the sets definable by a  $\Sigma_n$  (resp.  $\Pi_n$ ) formula of Peano arithmetic coincide with the  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) sets of Definition 5-1.1. We start for that by showing some closure properties similar to those of propositions 5-1.6 to 5-1.9.

**Proposition 3.3.** Let  $F(\bar{a}, x), F_1(\bar{a}, x), F_2(\bar{a}, x)$  be  $\Sigma_n$  (resp.  $\Pi_n$ ) formulas. Then, each of the following formulas is provably equivalent (using the axioms of arithmetic) to a  $\Sigma_n$  (resp.  $\Pi_n$ ) formula:

- $F_1(\bar{a}, x) \wedge F_2(\bar{a}, x), F_1(\bar{a}, x) \vee F_2(\bar{a}, x)$
- $\exists x < b F(\bar{a}, x), \forall x < b F(\bar{a}, x),$
- $\exists x F(\bar{a}, x)$  (resp.  $\forall x F(\bar{a}, x)$ ) ★

PROOF. Let  $F(\bar{a}, x) \equiv \exists y G(\bar{a}, x, y), F_1(\bar{a}, x) \equiv \exists y G_1(\bar{a}, x, y)$  and  $F_2(\bar{a}, x) \equiv \exists y G_2(\bar{a}, x, y)$ . Then, we have the following equivalences:

$$\begin{aligned} F_1(\bar{a}, x) \wedge F_2(\bar{a}, x) &\leftrightarrow \exists y \exists y_1, y_2 < y (G_1(\bar{a}, x, y_1) \wedge G_2(\bar{a}, x, y_2)) \\ F_1(\bar{a}, x) \vee F_2(\bar{a}, x) &\leftrightarrow \exists y (G_1(\bar{a}, x, y) \vee G_2(\bar{a}, x, y)) \\ \exists x < b F(\bar{a}, x) &\leftrightarrow \exists y \exists x < b G(\bar{a}, x, y) \\ \forall x < b F(\bar{a}, x) &\leftrightarrow \exists z \forall x < b \exists y < z G(\bar{a}, x, y) \\ \exists x F(\bar{a}, x) &\leftrightarrow \exists z \exists x < z \exists y < z G(\bar{a}, x, y). \end{aligned}$$

Now, if  $F, F_1, F_2$  are  $\Sigma_1$  with  $G, G_1, G_2 \Delta_0$ , the above equivalences show the proposition for the  $\Sigma_1$  case. By passing to the negation, the equivalences also hold for the  $\Pi_1$  case. Suppose the proposition is true for the  $\Sigma_n$  and  $\Pi_n$  cases. Then, the above equivalences for  $F, F_1, F_2$  of the  $\Sigma_{n+1}$  formulas with  $G, G_1, G_2 \Pi_n$  imply —using the induction hypotheses on  $G, G_1, G_2$ — the proposition for the  $\Sigma_{n+1}$  case. By passing to negation, the proposition is true for the  $\Pi_{n+1}$  case. ■

Note that the fourth equivalence in the above proof (the equivalence  $\forall x < b \ F(\bar{a}, x) \leftrightarrow \exists z \forall x < b \ \exists y < z \ G(\bar{a}, x, y)$ ) is the least trivial of all: the other four simply use the fact that two integers always have an upper bound, while this one requires the use of induction. This is something we will study in detail in Section 23-3.

Let's now move on to the announced equivalence. Given a  $\Sigma_1$  formula of Peano arithmetic  $\exists y \ F(x_1, \dots, x_n, y)$  where  $F$  is  $\Delta_0$ , the set  $\{(x_1, \dots, x_n) \in \mathbb{N} : \exists y \ F(x_1, \dots, x_n, y)\}$  is computably enumerable: we test the formula  $F$  little by little on all  $(n+1)$ -tuples  $x_1, \dots, x_n, y$  and when we find one for which  $F$  is true, we enumerate  $(x_1, \dots, x_n)$ . Gödel showed that any computably enumerable set could in fact be represented in this form.

**Theorem 3.4 (Gödel)**

*A set of integers  $A \subseteq \mathbb{N}$  is c.e. iff there exists a  $\Sigma_1$  formula  $F(n)$  of  $\mathcal{L}_{PA}$  such that  $n \in A$  iff  $\mathbb{N} \models F(n)$ .*

We will show Theorem 3.4 based on the model of general recursive functions which coincide, as we saw in Chapter 6, with computable functions. We will show that any general recursive partial function  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  is *represented* by a  $\Sigma_1$  formula of arithmetic  $F(n_1, \dots, n_k)$ , that is, to say

$$\{(\bar{n}, r) \in \mathbb{N}^{k+1} : f(\bar{n}) \downarrow = r\} = \{(\bar{n}, r) \in \mathbb{N}^{k+1} : \mathbb{N} \models F(\bar{n}, r)\}.$$

As any c.e. set is the domain of a partial function, this proves the theorem. The main difficulty lies in the management of the primitive recursion scheme, for which we need to encode lists of integers by arithmetic formulas. Gödel resorts to a clever use of a result of modular arithmetic: the Chinese Remainder Theorem.

**Lemma 3.5 (Chinese Remainder Theorem).** Let  $(a_0, \dots, a_n)$  be an arbitrary sequence of integers. Let  $(p_0, \dots, p_n)$  be a sequence of pairwise prime integers with  $p_i \geq a_i$  for  $i \leq n$ . Then, there exists an integer  $b$  such that  $a_i$  is the remainder of the Euclidean division of  $b$  by  $p_i$  for all  $i$ . ★

The Chinese Remainder Theorem will allow Gödel to code lists of integers of arbitrary size. Note that this is not the only way to establish a system for coding/decoding lists by arithmetic formulas, and we will see another one in Section 23-4.

**Lemma 3.6 (Gödel's  $\beta$  function).** There exists a function  $\beta : \mathbb{N}^3 \rightarrow \mathbb{N}$  represented by a  $\Delta_0$  formula, such that for all  $n$  and any sequence of integers  $(a_0, \dots, a_n)$ , there exist integers  $a, b \in \mathbb{N}$  for which  $\beta(a, b, i) = a_i$  for all  $i \leq n$ . ★

PROOF. The formula  $B(a, b, i, r)$  which represents  $\beta$  is as follows: “ $r$  is the remainder of the Euclidean division of  $b$  by  $a(i+1)+1$ ”. The formula is

indeed  $\Delta_0$ :  $r < a \times (i + 1) + 1 \wedge \exists c < b \ c \times (a \times (i + 1) + 1) + r = b$ . Let  $(a_0, \dots, a_n)$  be a sequence of integers. Let us show the existence of integers  $a, b$  such that this formula defines the function  $(a, b, i) \mapsto a_i$ .

Let  $m$  be such that  $m > \max\{a_i : i \leq n\}$  and  $m > n$ . Let  $a = m!$ . We will use the Chinese Remainder Theorem with  $p_i = a(i+1)+1$ . Let us show that these numbers are pairwise prime. Suppose absurdly that a prime number  $p$  divides  $p_i$  and  $p_j$  with  $i < j$ . So  $p$  also divides  $p_j - p_i = a(j-i)$ . Since  $p$  is prime then  $p$  divides  $a$  or  $p$  divides  $j-i$ . As  $a = m!$  with  $m > n \geq j > i$  then  $j-i$  divides  $a$  and therefore in all cases  $p$  divides  $a$ . So  $p$  divides  $a(i+1)$ . Since  $p$  also divides  $a(i+1) + 1$ , then  $p$  divides  $a(i+1) + 1 - a(i+1) = 1$  which is a contradiction. So the numbers  $p_i$  are mutually prime.

We have  $p_i = a(i+1) + 1 > m > a_i$  for all  $i$ . According to the Chinese Remainder Theorem, there exists an integer  $b$  such that for all  $i$ , the integer  $a_i$  is the remainder of the Euclidean division of  $b$  by  $a(i+1) + 1$ . ■

**PROOF OF THEOREM 3.4.** We show that any general recursive partial function is represented by a  $\Sigma_1$  formula of arithmetic. We leave it to the reader to show that this is indeed the case for the basic functions (projections, constant functions and successor function). We use Proposition 3.3 without mentioning it for each of the three schemes to come, in order to obtain a  $\Sigma_1$  formula which represents our function.

*Composition scheme.* Let

$$f(\bar{x}) = g(h_1(\bar{x}), \dots, h_k(\bar{x}))$$

for functions  $g, h_1, \dots, h_k$  represented by formulas  $G, H_1, \dots, H_k$ . Then,  $f$  is represented by the formula

$$F(\bar{x}, r) \equiv \exists y_1, \dots, y_k \ H_1(\bar{x}, y_1) \wedge \dots \wedge H_k(\bar{x}, y_k) \wedge G(y_1, \dots, y_k, r).$$

*Minimization scheme.* Let

$$f(\bar{x}) = \min\{a \in \mathbb{N} : \forall i \leq a \ g(\bar{x}, i) \downarrow \wedge g(\bar{x}, a) = 0\}$$

for  $g$  represented by a formula  $G$ . Then,  $f$  is represented by the formula:

$$F(\bar{x}, a) \equiv G(\bar{x}, a, 0) \wedge \forall i < a \ \exists r \neq 0 \ G(\bar{x}, i, r).$$

*Primitive recursion scheme.* This is where we will need Gödel's  $\beta$  function, represented by the formula  $B$ . Let

$$\begin{aligned} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, n+1) &= h(\bar{x}, n, f(\bar{x}, n)) \end{aligned}$$

for functions  $g, h$  represented by formulas  $G, H$ . Then,  $f$  is represented by

the following formula  $F(\bar{x}, n, r)$ :

$$\begin{aligned} \exists a, b \quad & B(a, b, n, r) \wedge \exists a_0 (B(a, b, 0, a_0) \wedge G(\bar{x}, a_0)) \wedge \forall i < n \\ & \exists a_i, a_{i+1} (B(a, b, i, a_i) \wedge B(a, b, i+1, a_{i+1}) \wedge H(\bar{x}, i, a_i, a_{i+1})). \end{aligned}$$

In order to see that  $f$  is well represented by  $F$ , it should be noted that  $B$ , as defined in the previous lemma, is always a functional formula: whatever the values of  $a, b, i$ , there are always at most one element  $r$  such that  $B(a, b, i, r)$  is true. Thus, if the formula  $F(\bar{x}, n, r)$  holds, there does exist a sequence  $a_0, \dots, a_n$  such that  $g(\bar{x}) = a_0$  and  $h(\bar{x}, i, a_i) = a_{i+1}$ , with  $a_n = r$ , the result of  $f(\bar{x}, n)$ . According to the previous lemma, there are indeed, for all  $n$  integers  $a, b$  which code via the function  $\beta$ , the sequence of values  $(f(\bar{x}, 0), f(\bar{x}, 1), \dots, f(\bar{x}, n))$ . The formula  $F$  therefore represents the function  $f$ . ■

We easily show from Theorem 3.4 that a set of integers is  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) iff it is described by a  $\Sigma_n$  (resp.  $\Pi_n$ ) formula of arithmetic.

### Coding of finite sequences

There are many ways to encode finite sequences of integers using natural numbers. Most of these techniques use primitive recursive functions, which requires showing beforehand that they are representable by simple arithmetic formulas. The Chinese Remainder Theorem allows a simple encoding of finite sequences based on Euclidean division, which is expressed by an immediate  $\Delta_0$  predicate (see Lemma 3.6).

## 3.2. Proofs and computation

A proof in our system of deduction à la Hilbert relies on a very precise system of rules, and it is easy to create a computer program which takes as parameter a proof — via an appropriate coding — and which verifies in a finite time whether the demonstration is valid or not. Indeed, the inference rules are clear, as for the axioms of logic, we have the tautologies of propositional calculus, four schemes of axioms for predicate calculus, and three schemes of axioms for equality. It is easy to check whether a sentence of propositional calculus — for example of the form  $(F \rightarrow G) \leftrightarrow (\neg G \rightarrow \neg F)$  — is a tautology, by ensuring that the sentence is always true for any truth value for  $F$  and  $G$ . It is also easy to check whether a sentence matches one of the equality axiom schemes or predicate calculus. We deduce the following theorem:

### Theorem 3.7 (Gödel)

Given a computably enumerable theory  $T$  in the language  $\mathcal{L}_{PA}$ , the set

of formulas  $F$  such that  $T \vdash F$  is computably enumerable.

It suffices to do a search on all the possible proofs from the axioms of  $T$  and to list all the formulas they prove. Hilbert's goal was to show that any arithmetical truth was provable, Peano arithmetic supposedly being a sufficient system to do so. If this were the case, then we could create an algorithm allowing to decide whether a mathematical statement  $F$  is provable or refutable: it would suffice to list all the statements proved by Peano arithmetic until we find  $F$  or  $\neg F$ .

Gödel showed that this was not possible, neither for Peano arithmetic, nor for any computably enumerable and consistent theory containing Peano arithmetic (however Rosser's help was needed for this last step). We start by proving a lemma which will help us in the following:

**Lemma 3.8.** If  $\mathbb{N} \models F$  where  $F$  is a closed  $\Sigma_1$  formula, then  $\text{PA} \vdash F$ . ★

PROOF. The formula  $F$  is of the form  $\exists x_1 \dots \exists x_n G(x_1, \dots, x_n)$  for a  $\Delta_0$  formula  $G$ . If  $F$  is true in  $\mathbb{N}$  then there are integers  $a_1, \dots, a_n \in \mathbb{N}$  such that  $G(a_1, \dots, a_n)$  is true in  $\mathbb{N}$ . It is easily shown by induction that if a  $\Delta_0$  formula and parameterized in  $\mathbb{N}$  is true in  $\mathbb{N}$ , then it is provable in PA. Informally, for a bounded existential quantification, it suffices to take a witness integer for this quantification and to show the formula with this witness, and for a universal quantification bounded by an integer  $a$ , it suffices to show that the formula is true for each integer less than  $a$ . ■

We now have the necessary ingredients to prove the first incompleteness theorem.

**Theorem 3.9 (Gödel's first incompleteness theorem)**

Let  $T \supseteq \text{PA}$  be a consistent c.e. theory, such that if  $T$  proves a  $\Sigma_1$  formula, then  $\mathbb{N}$  is a model of this formula. Then, there exists a  $\Sigma_1$  formula  $F$  such that  $T \not\vdash F$  and  $T \not\vdash \neg F$ .

PROOF. Let  $(\Phi_e)_{e \in \mathbb{N}}$  be an enumeration of all computable functions. According to Theorem 3.4 there exists a  $\Sigma_1$  formula  $F(e)$  of  $\mathcal{L}_{\text{PA}}$  such that

$$\{e \in \mathbb{N} : \mathbb{N} \models F(e)\} = \{e \in \mathbb{N} : \exists t \Phi_e(e)[t] \downarrow\}.$$

Suppose that for all  $e$ , we have  $T \vdash F(e)$  or  $T \vdash \neg F(e)$ . Note that if  $\exists t \Phi_e(e)[t] \downarrow$  then  $\mathbb{N} \models F(e)$  and therefore  $\text{PA} \vdash F(e)$  according to Lemma 3.8. As  $\text{PA} \subseteq T$  we also have  $T \vdash F(e)$ .

Now, if  $\forall t \Phi_e(e)[t] \uparrow$  we have  $\mathbb{N} \models \neg F(e)$ . According to our hypothesis we cannot have  $T \vdash F(e)$  because we would then have  $\mathbb{N} \models F(e)$  which contradicts  $\mathbb{N} \models \neg F(e)$ . Since  $T$  is complete, we therefore have  $T \vdash \neg F(e)$ .

It follows that the c.e. set  $\{e \in \mathbb{N} : T \vdash \neg F(e)\}$  coincides with the set  $\{e \in \mathbb{N} : \forall t \Phi_e(e)[t] \uparrow\}$  which makes the complement of the halting problem a c.e. set. Contradiction. ■

We notice several things. In the first place we had to restrict ourselves to 1-consistent theories  $T$ , that is to say theories which do not prove  $\Sigma_1$  formulas false in  $\mathbb{N}$ . We will soon see that there are many other possible models than  $\mathbb{N}$  for PA, and as many non 1-consistent theories as we want. So this is a very annoying restriction.

Then the proof shows in substance that a complete and 1-consistent theory allows to compute  $\emptyset'$ . We have already seen that any PA degree allows to compute a complete and consistent extension of PA, and since there are PA degrees which do not compute  $\emptyset'$ , there is in fact no hope of showing Gödel's theorem by reducing the halting problem to any complete and consistent theory that extends PA. The trick to get out of this situation was found by Rosser, the idea being in essence to use the fact that if a theory proves  $\exists t \Phi_e(e)[t] \downarrow = 0$  (whether this is true in  $\mathbb{N}$  or not), then it cannot show at the same time  $\exists t \Phi_e(e)[t] \downarrow = 1$ , assuming of course that the  $\Sigma_1$  formulas allowing to talk about computable functions, integrate well the fact that a function has at most one value on its input, which is the case in practice.

**Theorem 3.10 (Gödel-Rosser's incompleteness theorem)**

*Let  $T \supseteq \text{PA}$  be a consistent c.e. theory. Then, there exists a  $\Sigma_1$  formula  $F$  such that  $T \not\vdash F$  and  $T \not\vdash \neg F$ .*

PROOF. Let  $(\Phi_e)_{e \in \mathbb{N}}$  be an enumeration of all computable functions. Let us assume absurdly that we have  $T \vdash F$  or  $T \vdash \neg F$  for any  $\Sigma_1$  formula  $F$ . We will then compute a total function  $f : \mathbb{N} \rightarrow \{0, 1\}$  which is DNC<sub>2</sub>, i.e., such that for any integer  $e$  we have  $\Phi_e(e) \downarrow$  implies  $f(e) \neq \Phi_e(e)$ . Note that this is then a contradiction: if  $f$  is computable then there exists a code  $e$  such that  $\Phi_e(n) \downarrow = f(n)$  for all  $n$  and therefore in particular such that  $\Phi_e(e) \downarrow = f(e)$ .

The sets  $\{\exists t \Phi_e(e)[t] \downarrow = 0\}$  and  $\{\exists t \Phi_e(e)[t] \downarrow = 1\}$  are c.e. and therefore according to Theorem 3.4, there exist  $\Sigma_1$  formulas  $F_0(e)$  and  $F_1(e)$  such that

$$\begin{aligned} \{e \in \mathbb{N} : \mathbb{N} \models F_0(e)\} &= \{e \in \mathbb{N} : \exists t \Phi_e(e)[t] \downarrow = 0\} \\ \{e \in \mathbb{N} : \mathbb{N} \models F_1(e)\} &= \{e \in \mathbb{N} : \exists t \Phi_e(e)[t] \downarrow = 1\}. \end{aligned}$$

Note that PA also proves  $F_0(e) \rightarrow \neg F_1(e)$  and  $F_1(e) \rightarrow \neg F_0(e)$ : in other words  $e \mapsto \Phi_e(e)$  is a partial function which cannot have both 0 and 1 as a value for the same element.

To compute the value of  $f(e)$ , we then enumerate using Theorem 3.7 all the formulas demonstrated by  $T$ , until we find  $F_0(e)$  or  $\neg F_0(e)$ . By hypothesis, one of these two eventualities necessarily occurs. If  $T \vdash F_0(e)$  then we define  $f(e) = 1$ . Otherwise we define  $f(e) = 0$ . Let us show that our function has the expected property. If  $\exists t \Phi_e(e)[t] \downarrow = 0$  then  $\mathbb{N} \models F_0(e)$  and therefore, according to Lemma 3.8,  $T \vdash F_0(e)$ . So  $f(e) = 1 \neq \Phi_e(e)$ . If now  $\exists t \Phi_e(e)[t] \downarrow = 1$  then  $\mathbb{N} \models F_1(e)$  and therefore, according to Lemma 3.8,  $T \vdash F_1(e)$ . We then have  $T \not\vdash F_0(e)$  and therefore  $T \vdash \neg F_0(e)$ . So  $f(e) = 0 \neq \Phi_e(e)$ . Finally if  $\forall t \Phi_e(e)[t] \uparrow$  the value of  $f$  does not matter. ■

### Corollary 3.11

*Let  $T \supseteq \text{PA}$  be a complete and consistent theory. Then,  $T$  computes a  $\text{DNC}_2$  set.*

PROOF. From the proof of the previous theorem. ■

Let us dwell for a moment on what the Gödel-Rosser theorem tells us: there exists a  $\Sigma_1$  formula  $F$  which is neither provable nor refutable in PA. According to Lemma 3.8 if a  $\Sigma_1$  formula is true in  $\mathbb{N}$  it is provable in PA. We deduce  $\mathbb{N} \not\models F$  and therefore  $\mathbb{N} \models \neg F$ : that makes  $\neg F$  a true formula, in the sense that it is true in  $\mathbb{N}$ , but that we cannot prove using axioms of PA.

Finally, the theorem tells us that adding axioms is hopeless: as long as we keep the theory computably enumerable, it will remain incomplete. Note that we have shown with Proposition 2.24 that  $T$  could quite be extended into a complete and consistent theory, but this will be at the cost of no longer knowing its axioms, and we would then struggle using it to demonstrate anything ...

Let us now move on to Gödel's second theorem, even more surprising than the first, and which put an abrupt halt to Hilbert's program.

### Notation

For a computably enumerable theory  $T$  in  $\mathcal{L}_{PA}$ , let  $\text{Coh}(T)$  the closed  $\Pi_1$  formula corresponding to “ $T$  is a consistent theory”, that is -to say “for any proof  $p$  in  $T$ ,  $p$  is not a proof of  $\perp$ ”.

The existence of such a formula follows from the correspondence between  $\Sigma_n^0/\Pi_n^0$  sets and  $\Sigma_n/\Pi_n$  formulas.

**Theorem 3.12 (Gödel's second incompleteness theorem)**

*Let  $T$  be a c.e. consistent theory containing PA. Then,  $T \not\vdash \text{Coh}(T)$ .*

PROOF. The first step is to see that the proof of Theorem 3.10 can be formalized, via an appropriate coding, in Peano arithmetic: there exists a  $\Sigma_1$  formula  $F$  such that

$$\text{PA} \vdash \text{Coh}(T) \rightarrow (\ulcorner T \not\vdash F \urcorner \wedge \ulcorner T \not\vdash \neg F \urcorner).$$

Notations  $\ulcorner \Psi \urcorner$  indicate the transformation of  $\Psi$  into a statement of arithmetic.

Let us assume by the absurdity that  $T \vdash \text{Coh}(T)$ . Then, by the Modus Ponens rule, we have  $T \vdash \ulcorner T \not\vdash F \urcorner \wedge \ulcorner T \not\vdash \neg F \urcorner$  and therefore  $T \vdash \ulcorner T \not\vdash \neg F \urcorner$  as well as  $T \vdash \ulcorner T \not\vdash F \urcorner$ .

It should then be noted that the proof of Lemma 3.8 can also be done in Peano arithmetic, that is to say that Peano arithmetic shows that if a fixed  $\Sigma_1$  formula is true, then it is demonstrable in Peano arithmetic - and therefore in  $T$ . This therefore gives formally with the formula  $F$

$$T \vdash F \rightarrow \ulcorner T \vdash F \urcorner.$$

We then have by contraposition (using the equivalence between the negation and the encoding of the negation):

$$T \vdash \ulcorner T \not\vdash F \urcorner \rightarrow \neg F.$$

As  $T \vdash \ulcorner T \not\vdash F \urcorner$  then by Modus Ponens we get

$$T \vdash \neg F.$$

Finally, it suffices to see that if  $T$  proves any formula, then PA — and therefore  $T$  — shows that  $T$  proves this formula. This is once again an application of the formalization of the proof of Lemma 3.8 in  $T$ , the sentence  $T \vdash F$  being  $\Sigma_1$ . So we finally have

$$T \vdash \ulcorner T \vdash \neg F \urcorner$$

which contradicts  $T \vdash \ulcorner T \not\vdash \neg F \urcorner$ . ■

### 3.3. Consequence of the incompleteness theorems

Let us remember Gödel's completeness theorem:  $T \vdash F$  iff any model of  $T$  is a model of  $F$ . According to the second incompleteness theorem, no consistent and computably enumerable theory  $T$  containing arithmetic can demonstrate its own consistency:  $T \not\vdash \text{Coh}(T)$ . We therefore deduce for

example that there are models of Peano arithmetic within which the sentence  $\text{Coh}(\text{PA})$  is false: in these models, there exists in particular a proof of  $0 = 1$ . We have however shown that PA had models and therefore according to the consistency theorem, that PA could not prove  $0 = 1$ . The situation which seems paradoxical is resolved with the following consideration: the proof of  $0 = 1$  in a model of  $\text{PA} \cup \{\neg\text{Coh}(\text{PA})\}$  is not a real proof. The sentence  $\neg\text{Coh}(\text{PA})$  when expressed in the language of arithmetic is of the form: “there is an integer which codes for a valid proof of  $0 = 1$  in PA”. Such a model will therefore contain such an integer, but this integer will not be a true integer - otherwise by unwinding the proof encoded by this integer we would have a true proof of  $0 = 1$ .

Such a PA model is called *non-standard*. Any PA model contains 0 and 1. By the axioms governing addition, we easily show that any PA model contains of course the standard integers:  $0, 1, 2, 3, 4, \dots$ . A non-standard model of PA will contain integers *larger* than all standard integers, and from the point of view of the model, there is nothing to distinguish these integers from others. These integers also called non-standard.

Let us take a non-standard integer  $a$  of such a model. Using the axioms of Peano arithmetic we see that the integers  $a+1, a+2, a+3, \dots$  also exist in the model. By the axiom which states that any integer different from 0 has a predecessor we also see that the integers  $a-1, a-2, a-3, \dots$  are in the model. As  $a > n$  for all  $n \in \mathbb{N}$  then also  $a+a > a+n$  for all  $n \in \mathbb{N}$ . There is therefore also a non-standard integer greater than all  $a+n$ . By playing with the axioms of PA in this way, we arrive at the following theorem:

**Theorem 3.13**

*For any countable non-standard models of arithmetic, the order  $<$  consists of a copy of  $\mathbb{N}$ , followed by  $\mathbb{Q}$  copies of  $\mathbb{Z}$ .*

We therefore know what the order of the elements looks like in a non-standard model. Unfortunately, however, we do not know what the addition or the multiplication looks like:

**Theorem 3.14 (Tennenbaum)**

*Let  $\mathcal{M}$  be a countable non-standard model of PA. Let  $+, \times : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  be functions which represent the addition and multiplication functions of  $\mathcal{M}$ , once established a bijection between  $\mathcal{M}$  and  $\mathbb{N}$ . Then, neither  $+$  nor  $\times$  is computable.*

We will talk a little more about non-standard models of arithmetic in the part on Reverse Mathematics, and in particular in Section 23-3.

## 4. ZFC system

We have given a proof of the consistency of Peano arithmetic, using the consistency theorem (see Corollary 2.21) and providing a model for this theory. According to Gödel's second incompleteness theorem, we necessarily used a theory more powerful than PA to create this model. What theory is it? We can give several answers to this question. The most natural theory to show the consistency of first-order arithmetic is undoubtedly the theory of *second-order* arithmetic, which will be discussed in more detail in Chapter 22.

### 4.1. Motivations

In second-order arithmetic, we allow ourselves not only to work with integers, but also with sets of integers. The existence of arbitrary sets of integers is questionable, at least more than the existence of the integers themselves: they are infinite objects, which are in uncountable quantity, and if there is a “legitimate” aspect of accepting the existence of computable sets of integers (after all we can write algorithms that produce them, their theoretical existence is therefore doubled by a certain form of practical existence), we can more easily question that of other sets. Some of them are still accessible, in the sense that they have a clear definition, for example the Turing jump. What makes  $\emptyset'$  more legitimate than another arbitrary non-computable set is the fact that it is *definable*, and what is more by a fairly simple formula -  $\Sigma_1$  in particular:  $\{e \in \mathbb{N} : \Phi_e(e) \downarrow\}$ . In second-order arithmetic, we allow ourselves the existence of all sets definable by an arbitrary arithmetic formula, in particular the  $\Sigma_n^0$  formulas for a certain  $n$ : this is called the *axiom of comprehension*. Once we have admitted the existence of a set  $X$ , it would be absurd not to accept the existence of computable sets with  $X$  as an oracle, and if we accept the axiom of comprehension, it would be just as absurd not to accept the existence of sets definable by a formula of arbitrary arithmetic, which could use  $X$  as oracle.

Second-order arithmetic therefore consists of Peano arithmetic, to which we add the axiom of comprehension which makes it possible to validate the existence of definable sets via a formula of arithmetic, possibly using a oracle — an already existing set —. A very large part of mathematics can already be formalized in this way, but not all of mathematics. In particular, nothing in second-order arithmetic allows us to speak of the set of all the subsets of integers, or even of arbitrary subsets of the latter. For that, we need another axiom: given a set  $X$ , the *power set*  $\mathcal{P}(X)$  of  $X$  — that is to say the set of all the subsets of  $X$  — is legitimate, it exists and we can use it. The power set of  $X$  is not of the same nature as  $X$ . One is a set

of integers and the other is a set of sets of integers. Set theory makes it possible to treat these two elements in a homogeneous way: every element will be a set, including the integers. Informally the integer 0 will be the empty set, the integer 1 will be the set which contains 0, the integer 2 will be the set which contains 0 and 1, and inductively the set  $n + 1$  will be the set which contains  $m$  for all  $m \leq n$ . We can of course imagine other ways of representing integers by sets, and with practice these do not matter, but we must choose one. This representation is the one that has become standard, under the impulse of the mathematician John von Neumann. We will discuss this again with the study of ordinals in Chapter 27.

#### 4.2. Zermelo system

Now that we only work with sets, we are no longer in the language of arithmetic. Our only relation symbols will be that of ownership  $\in$  and of course the symbol of equality  $=$ . We need some basic axioms in order to govern the elementary manipulations of sets:

- (1) *Axiom of empty set*: the empty set exists.
- (2) *Axiom of pairing*: if  $a$  and  $b$  are sets, then  $\{a, b\}$  is a set.
- (3) *Axiom of union*: the idea is that if  $(a_i)_{i \in I}$  is a collection of sets indexed by a set  $I$ , then the set  $\bigcup_{i \in I} a_i$  exists. The notion of “indexed set” is not formally defined in Set Theory, we will say instead that if  $b$  is a set whose elements are  $a_i$ , then the union of all these  $a_i$  exists. To be quite formal, the axiom is:  $\forall b \exists c \forall x (x \in c \leftrightarrow \exists a \in b \ x \in a)$ .

We also need an axiom which governs the equality between two sets.

- (4) *Axiom of extensionality*: two sets are equal iff they have the same elements.

Finally, there is our troublemaking axiom, which pushed us into this Set Theory allowing, for example, to deal with the power set of  $\mathbb{N}$ .

- (5) *Axiom of power set*: for any set  $a$ , the power set of  $a$ , denoted  $P(a)$  exists. Formally:  $\forall a \exists b \forall c (c \in b \leftrightarrow c \subseteq a)$  where  $c \subseteq a$  can be written as  $\forall x (x \in c \rightarrow x \in a)$ .

We can finally add our axiom of comprehension, allowing to build sets from first-order formulas, possibly using other sets as parameters. As there is an infinity of first-order formulas, it is not a question of a single axiom, but of an axiom scheme: one axiom per formula.

- (6) *Comprehension scheme*: for a formula  $F(y, x_1, \dots, x_n)$  fixed in the language of Set Theory, for all  $n$ -tuples of sets  $b_1, \dots, b_n$  and for any set  $a$ , the set  $\{y \in a : F(y, b_1, \dots, b_n)\}$  exists.

We can easily verify that the preceding axioms imply the existence of all hereditarily finite sets, that is to say finite sets, the elements of which are themselves finite sets, etc., until arriving, by scrolling down the tree describing the membership relations of a set with its elements, to the empty set for any leaf of this tree. Nothing allows us for the moment to speak of the set of all integers. In fact, we need an axiom for this.

- (7) *Axiom of infinity*: there exists an infinite set. Formally

$$\exists x (\emptyset \in x \wedge \forall y \in x \quad y \cup \{y\} \in x).$$

The axiom of infinity morally asserts the existence of  $\mathbb{N}$  as a set. Via the encoding that we have given of integers, we can check that the set encoding the integer  $n + 1$  is equal to the set  $n \cup \{n\}$ , where  $n$  is the set encoding the integer  $n$ . The axiom of infinity therefore tells us that there exists a set containing all integers. It could possibly contain other elements, but using the other axioms, we define  $\mathbb{N}$  as the smallest set  $x$  —smallest for inclusion— such that  $\emptyset \in x \wedge \forall y \in x \quad y \cup \{y\} \in x$ .

### 4.3. Replacement axiom and Borel games

The axioms from (1) to (7) form Zermelo's theory Z. They are sufficient to develop a large part of mathematics. Fraenkel and Skolem will introduce a new axiom, both intuitive and formally necessary to develop the theories of ordinals and hierarchies of infinities. It is more precisely a scheme of axioms, which essentially says that the image of any functional formula exists. A formula  $F(y, r)$  is *functional* if for all  $y$ , the formula  $F(y, r_y)$  is satisfied for exactly one set  $r_y$ . Formally:

- (8) *Replacement scheme*: for a functional formula  $F(y, x_1, \dots, x_n, z)$  fixed in the language of Set Theory, for any  $n$ -tuple of sets  $b_1, \dots, b_n$ , for any set  $a$ , the set

$$\{z : \exists y \in a \quad F(y, b_1, \dots, b_n, z)\}$$

exist.

Set Theory becomes strictly more powerful with the replacement scheme, which allows in particular to show the existence of the set  $\mathbb{N} \cup P(\mathbb{N}) \cup P(P(\mathbb{N})) \dots$  which it is impossible to build without this axiom.

It is remarkable to note that a joint use of the axiom of power set and of the replacement scheme, is essential to construct certain sets of integers - which

will be necessarily extreme complexity in terms of Turing degree. The emblematic example is Martin's determination theorem for Borel games. Let's see what it is: consider a class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  for the moment arbitrary, and consider the following two-player game: Player 1 chooses a bit  $x_0 \in \{0, 1\}$ , then Player 2 in turn chooses a bit  $x_1 \in \{0, 1\}$ , and so on, in step  $2n$ , Player 1 chooses bit  $x_{2n}$  and in step  $2n + 1$  Player 2 chooses bit  $x_{2n+1}$ . At the "end" of the game, we get a set  $X = x_0x_1x_2 \dots$ . Player 1 wins the game if  $X \in \mathcal{B}$ , otherwise Player 2 wins. The question then is: does one of the two players have a winning strategy? A *strategy* for Player 1 is a function  $f$  which takes as parameter a string  $\sigma$  of even size, corresponding to what has been played so far, the last move being the last bit of  $\sigma$  played by Player 2, and who returns the next move  $f(\sigma)$ . Such a strategy is winning if the set obtained by playing the moves given by the function  $f$  is always in  $\mathcal{B}$ , whatever the moves made by Player 2.

We will see in Chapter 17 that a large variety of classes  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  do not require the axiom of power set to be handled: we already have an example with the  $\Pi_1^0$  or  $\Sigma_1^0$  classes. We will carry out iterated constructions of increasingly complex classes, which can be encoded by a countable object, in a manner analogous to the encoding of  $\Pi_1^0$  classes by trees: these will be the so-called *Borel classes*. The mathematician Donald A. Martin, of whom we will speak again in Chapter 12, showed the following remarkable theorem:

**Theorem 4.1 (Martin [148])**

*Let  $\mathcal{B}$  be a Borel class. Then, for the game described above with the class  $\mathcal{B}$ , one of the two players has a winning strategy.*

The strategy of a Borel game is a function  $f : 2^{<\mathbb{N}} \rightarrow \{0, 1\}$  and can therefore be represented by a set of integers. Friedman [66] showed that for some Borel classes, of relatively simple complexity, such a strategy could not be constructed without the use of the axiom of power set, and even without arbitrary iteration of the application of this axiom. Thus, to show the existence of certain reals, we must appeal to the axiom of power set used jointly with the replacement scheme. More precisely, it is used to construct  $\mathbb{N} \cup P(\mathbb{N}) \cup P(P(\mathbb{N})) \dots$ , essential for the definition of our function - the winning strategy for a certain Borel class.

#### 4.4. Axiom of foundation and coding

Another axiom was added by Fraenkel and Skolem, as well as by von Neumann. Unlike the other axioms, the latter is almost useless for the construction of the mathematical universe, but we add it simply because it corresponds to our conception of things: the axiom says in essence that

given a set  $x$ , if we consider an element  $x_1 \in x$ , then an element  $x_2 \in x_1$ , thus continuing inductively with an element  $x_{n+1} \in x_n$ , we will necessarily eventually obtain the empty set for some  $n \in \mathbb{N}$ . In other words, there are no sets other than those that we can build inductively via the other axioms starting from the empty set. In particular, there is no set  $x$  such that  $x \in x$ . This may or may not seem obvious to everyone, but in any case reflects the conception generally adopted in the community on the nature of sets.

In order to support our point, let us recall that one of the interests of Set Theory is the ability to formalize the totality of mathematics in it. This formalization involves the coding of the usual mathematical structures by sets, and this coding is done anyway only with sets which respect the axiom of foundation - whether the latter is adopted or not. So if it is not contradictory to think that there exist other sets which do not respect this axiom, in practice we do not need them, and such objects do not correspond *a priori* to anything tangible.

Let us end by insisting on the fact that if we can code the rest of mathematics by sets, this is on the other hand not a reason for doing so, and we are obviously much more comfortable working with integers, reals or other. What is interesting is the *existence* of such an encoding, and the fact that if a statement is undecidable in Set Theory, it then becomes undecidable.

#### 4.5. Axiom of choice and cardinality

Zermelo's theory Z plus the replacement axiom and the foundation axiom gives ZF theory, from Zermelo/Fraenkel.

A final axiom, arguably the most famous, is the axiom of choice, which was originally part of Zermelo's theory. The need for this axiom should be easy to understand via the analogy that can be made of it in Computability Theory. We have for example seen that any non-empty and countable  $\Pi_1^0$  class contains a computable set. On the other hand it is not possible, given the code of a non-empty  $\Pi_1^0$  class, to find uniformly an algorithm allowing to compute an element of it. In particular, given a countable sequence  $(\mathcal{F}_n)_{n \in \mathbb{N}}$  of non-empty  $\Pi_1^0$  classes, there does not necessarily exist a computable function allowing to *choose* an element in each of these classes. The key question here is that of uniformity. Obviously, we can create this choice function using  $\emptyset'$ , but what happens if we consider more complex classes? Can we still build a choice function? The answer is no, without the help of a new axiom.

- (9) *Axiom of choice*: given a collection  $(A_i)_{i \in I}$  of non-empty sets, there exists a function  $f : I \rightarrow \bigcup_{i \in I} A_i$  such that  $f(i) \in A_i$  for all  $i$ .

The axiom of choice appears necessary to develop a complete theory of the cardinality of sets. In particular with the axiom of choice, we can show that for any set  $A, B$ , we have  $|A| \leq |B|$  or  $|B| \leq |A|$  (we will come back to this with the detailed study of ordinals in Chapter 27). This is no longer true without the axiom of choice, the example par excellence in Computability Theory being certainly that of Turing degrees. It is easy to construct an injection of  $2^{\mathbb{N}}$  in the Turing degrees, it is however impossible to construct an injection of Turing degrees in  $2^{\mathbb{N}}$  without the axiom of choice (in particular it is impossible to show the existence of a function  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  such that  $X \equiv_T Y \leftrightarrow f(X) = f(Y)$ ).

A first reaction is to simplify your life and use the axiom of choice if necessary. However, this approach is not in the spirit of Computability Theory, from which Set Theory is less distant than one might think: axioms other than the axiom of choice allow us to *construct* more and more complex objects starting from existing objects, a bit like the way one constructs more and more complex Turing degrees by iterating the jump. The axiom of choice is fundamentally non-constructive, and from this point of view is not as legitimate as the others. It also leads to theorems which seem paradoxical, the best known example being the *Banach/Tarski paradox*, a construction, which using the axiom of choice, shows how to cut a ball of the space  $\mathbb{R}^3$  into a finite number of pieces, and how to reassemble these pieces to obtain two balls strictly identical to the first.

By not accepting the axiom of choice, we are of course leaving this ideal world where the cardinality of any set is comparable, but it is in fact an “artificial” situation of which we have do not really need.

## 4.6. Independence results

The theory obtained with axioms (1) - (8) is the ZF theory, if we add the axiom of choice to it, we then have the so-called ZFC theory.

### 4.6.1. Axiom of choice

The question of knowing whether the axiom of choice is provable from the other axioms, or even that of knowing whether it does not risk introducing contradiction, has long remained open. Gödel’s completeness theorem allows the following technique: if we assume that ZF is consistent, and therefore has a model, and that using this model we can build a model of ZFC, we will have showed that the consistency of ZF implies that of ZFC, and in particular that ZF cannot prove that the axiom of choice is false (unless of course ZF is inconsistent). This is exactly what Gödel did a few years later [73], via his model called *constructible universe*. It was only later in 1962 with his famous forcing technique, some aspects of which we

will see in Chapter 11, that Cohen [36] succeeded in the feat of building a model of ZF in which the axiom of choice is false: the axiom of choice can therefore neither be proved nor refuted in ZF.

#### 4.6.2. Continuum hypothesis

The question that obsessed Cantor throughout his life (and many mathematicians for almost a century) is also independent of the other axioms of Set Theory (with or without the axiom of choice). Gödel's constructible universe also constitutes a ZFC model in which the continuum hypothesis is verified, i.e., there is no set  $A$  for which  $|\mathbb{N}| < |A| < |2^{\mathbb{N}}|$ . Still with his forcing technique, Cohen built models of ZFC in which we have an arbitrary number of infinities strictly between  $|\mathbb{N}|$  and  $|2^{\mathbb{N}}|$ .

Note here that we can give several versions of the continuum hypothesis. The one originally formulated by Cantor concerned sets of reals: does there exist a set  $\mathcal{A} \subseteq \mathbb{R}$  such that  $|\mathbb{N}| < |\mathcal{A}| < |\mathbb{R}|$ ? Later the question will be extended to any set, and in particular to ordinals, which we will see formally in Chapter 26. Regardless of the version of the continuum hypothesis, this is a question independent of the other axioms of Set Theory.

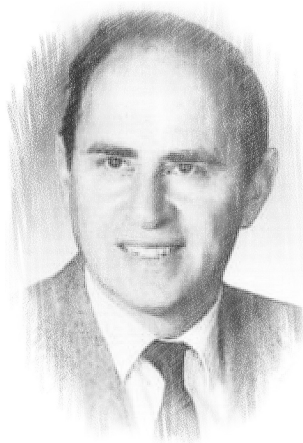
# Chapter 10

## Cohen forcing

Forcing is a technique invented by the mathematician Paul Cohen in the early 1960s, to show the independence of the continuum hypothesis and the axiom of choice from Zermelo/Fraenkel's theory. This technique has revolutionized Set Theory, and also plays a preponderant role in Computability Theory, where it has established itself as one of the two main techniques of set construction, alongside the priority method (see Chapter 13).

Historically, this technique was designed to extend a model  $\mathcal{M}$  of ZF theory by adding a new object  $G$  to form a new model  $\mathcal{M}[G]$ , while allowing the elements of the model  $\mathcal{M}$  to control the properties of the extended model  $\mathcal{M}[G]$ .

More generally, the forcing technique makes it possible to create mathematical objects using increasingly precise approximations, while being able to control certain properties of the final object before the construction is finished. A forcing notion is above all the definition of a partial order of approximations  $(\mathbb{P}, \leq)$ , where the relation  $d \leq c$  means that the approximation  $d$  is more precise than  $c$ . When a property (or requirement)  $\mathcal{R}$  on the final object is already determined by an approximation  $c \in \mathbb{P}$ , in other



Paul Joseph Cohen, 1934–2007

words when whatever the sequence of the construction, the final object will satisfy  $\mathcal{R}$ , we then say that  $c$  *forces*  $\mathcal{R}$ . The heart of the technique of forcing lies in the ability to force complex properties on the final object from simple approximations.

Forcing is often considered a difficult technique to grasp at first glance. Its application to Computability Theory, which is simpler, makes it possible on the one hand to approach it smoothly, and on the other hand to more easily understand the details of its underlying mechanisms. This simplicity is mainly due to two aspects:

- Like the finite extension method, we will mainly use forcing to create sets of integers, while controlling their computational powers. We will therefore be interested in forcing simple formulas, of the type “ $\Phi^G(n) \downarrow$ ” or “ $\Phi^G(n) \uparrow$ ”, where  $G$  denotes the final set. It is therefore a question of forcing  $\Sigma_1^0$  or  $\Pi_1^0$  formulas. We will see that several constructions, in particular those with the finite extension method, are simplified uses of forcing. We will also see how to extend the forcing to arbitrarily complexity formulas, which presents a first level of additional complexity.
- Contrary to Set Theory, we will only force properties whose quantifiers relate to integers. The creation of a set of integers by forcing does not modify the nature of the integers, and therefore the scope of the quantifiers. In Set Theory on the other hand, quantifiers relate to sets, and creating a new set adds a large amount of sets to the model, and therefore changes the scope of quantifiers. It is then necessary to use *names*, to talk about the objects of our extended model inside our base model. We will not need to use names in Computability Theory, which makes the presentation more accessible.

## 1. Formulas of second-order arithmetic

The statements that one “forces” in Computability Theory are always formulas of first-order arithmetic *with free set variables*. These formulas are a special case of second-order arithmetic, which we will talk about in more detail in Chapter 22. A formula of second-order arithmetic has two types of variables: *first-order* variables, representing integers and which will be noted in lowercase, and *second-order* variables, representing sets of integers and which will be denoted in upper case. The language is enriched with the membership symbol  $\in$ , and the atomic formula “ $x \in A$ ”. In second-order arithmetic, quantifications can be on integers, and on sets of integers. For example, the statement “ $\forall A \forall n \exists m (m > n \wedge m \in A)$ ” states that any set of integers is infinite.

First-order arithmetic with free set variables is a restriction of second-order arithmetic, where only the quantifications on the integers are allowed<sup>1</sup>. During the evaluation of a second-order arithmetic formula in a given model, its free set variables are replaced by parameters, i.e., elements of the model considered. Thus the statement  $F(G) = \forall n \exists m (m > n \wedge m \in G)$  is a formula of first-order arithmetic with  $G$  as free set variable, and any infinite set  $X \subseteq \mathbb{N}$  is a parameter for which  $F(X)$  is true.

We have seen in Section 9-3 the  $\Delta_0^0$  formulas of arithmetic: those containing only bounded quantifications, that is to say quantifications of the form  $\forall x \leq y$  and  $\exists x \leq y$ . We can define a hierarchy on arithmetic formulas with free set variables, similar to the hierarchy of Definition 9-3.2.

**Definition 1.1.**

1. A formula  $F(Y_1, \dots, Y_k, x_1, \dots, x_m)$  of second-order arithmetic is  $\Sigma_n^0$  if

$$F(Y_1, \dots, Y_k, x_1, \dots, x_m) = \overbrace{\exists y_1 \forall y_2 \dots Q y_n}^{n \text{ quantifiers}} G(Y_1, \dots, Y_k, x_1, \dots, x_m, y_1, \dots, y_n)$$

for a  $\Delta_0^0$  formula  $G(Y_1, \dots, Y_k, x_1, \dots, x_m, y_1, \dots, y_n)$ , where  $Q$  is  $\exists$  if  $n$  is odd, and  $\forall$  if  $n$  is even.

2. A formula  $F(Y_1, \dots, Y_k, x_1, \dots, x_m)$  of second-order arithmetic is  $\Pi_n^0$  if

$$F(Y_1, \dots, Y_k, x_1, \dots, x_m) = \overbrace{\forall y_1 \exists y_2 \dots Q y_n}^{n \text{ quantifiers}} G(Y_1, \dots, Y_k, x_1, \dots, x_m, y_1, \dots, y_n)$$

for a  $\Delta_0^0$  formula  $G(Y_1, \dots, Y_k, x_1, \dots, x_m, y_1, \dots, y_n)$ , where  $Q$  is  $\forall$  if  $n$  is odd, and  $\exists$  if  $n$  is even. ◇

We have so far implicitly used the formulas of second-order arithmetic, via the use of functionals. Theorem 9-3.4 comes in the following equivalence.

**Theorem 1.2**

Let  $A, Z \in 2^{\mathbb{N}}$ . The following statements are equivalent:

- (1)  $A \subseteq \mathbb{N}$  is  $Z$ -c.e.
- (2) There exists a  $\Sigma_1^0$  formula of second-order arithmetic  $F(X, n)$  such

<sup>1</sup>We will see only very late in this work, in Part IV the great computational complexity hidden behind second-order quantifications.

that  $A = \{n \in \mathbb{N} : \mathbb{N} \models F(Z, n)\}$ .

- (3) There exists a Turing functional  $\Phi(Z, n)$  such that  $A = \{n \in \mathbb{N} : \Phi(Z, n) \downarrow\}$ .

## 2. $\Sigma_1^0/\Pi_1^0$ forcing

We will now begin our immersion in the world of forcing by studying a specific forcing notion, namely Cohen forcing. As explained in the introduction, a forcing notion is specified by its partial order of approximations. In our case, it will be the partial order of the strings, equipped with the prefix relation. It is one of the simplest concepts of forcing conceptually, but which already contains the fundamental concepts of forcing.

There are several ways to approach forcing, with different levels of abstraction. It is possible to see it as an elaboration of the finite extension method, seeking to systematize the satisfaction of requirements, and to extract a general construction from it. We will present this approach in Section 2.1.

It is also possible to formulate forcing in a topological framework. Topology makes it possible to define a notion of “negligible” or *meager* class of sets. The construction of a set by forcing then consists in choosing an element “typical”, that is to say avoiding a negligible set of undesirable properties. We will see this approach in Section 2.2.

### 2.1. The finite extension approach

Let us reconsider the finite extension method developed in Section 4-8. The goal is to build a set  $G \in 2^{\mathbb{N}}$  satisfying an infinity of requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$ . Each requirement is treated independently, and must therefore be satisfied while leaving enough degrees of freedom in the construction to satisfy the other requirements.

#### Check-in on requirements

The requirements — which we have seen so far informally — can be seen as formulas in the language of first-order arithmetic with set variables, and in particular with a free second-order variable representing the set  $G$  which must fulfill the requirement. The finite extension method consists of building  $G \in 2^{\mathbb{N}}$  by specifying increasingly long initial segments, represented as strings  $\sigma \in 2^{<\mathbb{N}}$ . The  $\Sigma_1^0$  requirements are those corresponding to  $\Sigma_1^0$  formulas, or in an equivalent way those which one can always put in the form “ $\Phi_e(G, 0) \downarrow$ ” for a functional  $\Phi_e$ . The  $\Pi_1^0$  requirements are those corresponding to  $\Pi_1^0$  formulas or in an equivalent way those which

one can always put in the form “ $\Phi_e(G, 0) \uparrow$ ” for a functional  $\Phi_e$ .

**Satisfying a requirement.** The general procedure for satisfying a requirement  $\mathcal{R}_e$  is as follows: given a string  $\sigma \in 2^{<\mathbb{N}}$  representing an initial segment of  $G$  already specified to satisfy previous requirements, we must find a string  $\tau \succeq \sigma$  such that the requirement  $\mathcal{R}_e$  is satisfied. The final set  $G$  then not being known, the requirement must be satisfied whatever the rest of the construction, in other words it must be satisfied for all  $G \in [\tau] = \{X \in 2^{\mathbb{N}} : \tau \prec X\}$ .

**Partial order of strings.** Let's take some height, and consider the finite extension method from a more abstract point of view. We have a partial order  $(2^{<\mathbb{N}}, \preceq)$ , which corresponds to the set  $2^{<\mathbb{N}}$  of finite strings, equipped with the prefix relation. We also have an interpretation function  $[\cdot] : 2^{<\mathbb{N}} \rightarrow \mathcal{P}(2^{\mathbb{N}})$  defined by  $[\sigma] = \{X \in 2^{\mathbb{N}} : \sigma \prec X\}$ . Intuitively, the elements of  $2^{<\mathbb{N}}$  represent approximations of the set  $G$  under construction. Given an approximation  $\sigma$ ,  $[\sigma]$  is the class of sets that we could potentially obtain at the end of the construction. The further we advance in the construction, the more the approximation is refined and the class of candidate sets is restricted. Thus, we have the following property: if  $\sigma \preceq \tau$ , then  $[\tau] \subseteq [\sigma]$ . Let us see a first definition of the forcing relation.

**Definition 2.1.** Let  $\mathcal{R}$  be a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement. A string  $\sigma$  *forces*  $\mathcal{R}$ , denoted by  $\sigma \Vdash^* \mathcal{R}$ , if the requirement is satisfied for all  $G \in [\sigma]$ .  $\diamond$

**Density.** We can abstract ourselves from the notion of requirement by representing a requirement  $\mathcal{R}_e$  as the set  $P_e \subseteq 2^{<\mathbb{N}}$  of the strings which force it. Note that if  $\sigma$  forces  $\mathcal{R}_e$ , then every  $\tau \succeq \sigma$  also forces  $\mathcal{R}_e$  because  $[\tau] \subseteq [\sigma]$ . The set  $P_e$  is therefore closed under suffix. The procedure for satisfying the requirement  $\mathcal{R}_e$  consists in showing that for any string  $\sigma \in 2^{<\mathbb{N}}$ , there exists an extension  $\tau \succeq \sigma$  such that  $\tau \in P_e$ . If this is the case, we will say that the set  $P_e$  is dense:

**Definition 2.2.** A set  $W \subseteq 2^{<\mathbb{N}}$  is *dense* if for all  $\sigma \in 2^{<\mathbb{N}}$ , there exists an extension  $\tau \succeq \sigma$  such that  $\tau \in W$ .  $\diamond$

Intuitively, a set  $W \subseteq 2^{<\mathbb{N}}$  is dense if whatever the current construction  $\sigma_0 \preceq \sigma_1 \preceq \dots \preceq \sigma_n$  using the finite extension method, it is never too late to find an extension  $\sigma_{n+1} \succeq \sigma_n$  in  $W$ . It follows that if we have a countable set of requirements represented by dense sets  $(P_n)_{n \in \mathbb{N}}$ , since these sets are dense, there exists an infinite sequence  $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$  such that for all  $n$ , there exists an integer  $m$  for which  $\sigma_m \in P_n$ . In particular, let  $\{G\} = \bigcap_n [\sigma_n]$ , then  $G$  will have initial segments in each set  $P_e$ .

**Remark**

If the sets  $(P_n)_{n \in \mathbb{N}}$  are uniformly c.e, then the construction of the infinite sequence  $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$  can be done in a computable way, and the resulting set  $G$  is also computable.

**Genericity.** We can formalize the construction of the finite extension method into a trivial theorem in view of the previous intuitions.

**Definition 2.3.** We say that a set  $G \in 2^{\mathbb{N}}$  *meets* a set  $W \subseteq 2^{<\mathbb{N}}$  if  $G \upharpoonright_n \in W$  for a some  $n \in \mathbb{N}$ . Let  $\vec{D} = (D_n)_{n \in \mathbb{N}}$  be a sequence of sets of strings. A set  $G \in 2^{\mathbb{N}}$  is  $\vec{D}$ -*generic* if it meets every  $D_n$ .  $\diamond$

**Theorem 2.4**

Let  $\vec{D} = (D_n)_{n \in \mathbb{N}}$  be a countable sequence of dense sets of strings and let  $\sigma \in 2^{<\mathbb{N}}$ . There is a  $\vec{D}$ -generic set which extends  $\sigma$ .

PROOF. Let  $\sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$  be the strictly increasing infinite sequence of strings defined inductively as follows. Initially,  $\sigma_0 = \sigma$ . If  $\sigma_n$  is defined,  $\sigma_{n+1}$  is a strict extension of  $\sigma_n$  in  $D_n$ . Such an extension exists by density of  $D_n$ . Let  $\{G\} = \bigcap_n [\sigma_n]$ . Then,  $G$  is  $\vec{D}$ -generic and extends  $\sigma$ . ■

We will see the topological counterpart of the previous theorem with Lemma 2.14.

Theorem 2.4 says, among other things, that if  $(D_n)_{n \in \mathbb{N}}$  is a sequence of dense sets corresponding to the requirements  $(\mathcal{R}_n)_{n \in \mathbb{N}}$ , then there exist  $\vec{D}$ -generic sets, which therefore all satisfy the requirements simultaneously.

There is an uncountable amount of dense sets of strings, and a set  $G \in 2^{\mathbb{N}}$  cannot be generic for all of these sets simultaneously, simply because the set  $\{\sigma \in 2^{<\mathbb{N}} : \sigma \not\prec G\}$  is a dense set that  $G$  does not meet. The notion of genericity is therefore dependent on a countable collection  $\vec{D}$  of dense sets of strings.

**Sufficiently generic**

It is common to state results of the form “any *sufficiently generic* set satisfies such and such property”. This means that there exists a countable sequence  $\vec{D} = (D_n)_{n \in \mathbb{N}}$  of dense sets of strings such that any  $\vec{D}$ -generic set satisfies the property. Note that given any other countable sequence of dense sets of strings  $\vec{E} = (E_n)_{n \in \mathbb{N}}$ , the sequence  $\{\vec{D}, \vec{E}\}$  is again countable, and we can therefore construct a  $\{\vec{D}, \vec{E}\}$ -generic set. This is what justifies the name of *sufficiently generic*.

We will reformulate the proof of Proposition 4-8.2 in terms of density and genericity. For that, we need to extend the forcing relation to  $\Sigma_2^0$  requirements, which does not present any difficulty: a string  $\sigma$  *forces* such a requirement if the latter is satisfied for all  $G \in [\sigma]$ . We will see in Section 4 that this idea no longer works for  $\Pi_2^0$  requirements.

**Proposition (4-8.2).** For any non-computable set  $A$ , there exists a set  $B$  such that  $B \not\leq_T A$  and  $A \not\leq_T B$ . ★

PROOF. Let  $A$  be a non-computable set. We want to build a set  $B$  satisfying the requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : \exists x \Phi_e^A(x) \uparrow \vee \exists x \Phi_e^A(x) \downarrow \neq B(x) \quad \mathcal{S}_e : \exists x \Phi_e^B(x) \uparrow \vee \exists x \Phi_e^B(x) \downarrow \neq A(x).$$

Let  $R_e \subseteq 2^{<\mathbb{N}}$  and  $S_e \subseteq 2^{<\mathbb{N}}$  be the set of strings forcing respectively  $\mathcal{R}_e$  and  $\mathcal{S}_e$ .

**Density of the set  $R_e$ .** Let  $\sigma \in 2^{<\mathbb{N}}$  and let  $x = |\sigma|$ . Two cases arise:

- Case 1:  $\Phi_e^A(x) \downarrow = i$  for an  $i \in \{0, 1\}$ . It is then sufficient to define  $\tau$  as the unique string of length  $|\sigma| + 1$  extending  $\sigma$  such that  $\tau(x) = 1 - i$ . For all  $X \in [\tau]$ ,  $X(x) = 1 - i \neq \Phi_e^A(x)$ , so  $\tau \in R_e$ .
- Case 2:  $\Phi_e^A(x) \uparrow$ . In this case,  $R_e = 2^{<\mathbb{N}}$  and  $\sigma \in R_e$ .

**Density of the set  $S_e$ .** Let  $\sigma \in 2^{<\mathbb{N}}$ . Three cases arise:

- Case 1: there exists an input  $x$  and a set  $X \succeq \sigma$  such that  $\Phi_e^X(x) \downarrow \neq A(x)$ . In this case, by the use property, there exists a finite string  $\tau \succeq \sigma$  such that  $\Phi_e^X(x) \downarrow \neq A(x)$  for all  $X \in [\tau]$ . The string  $\tau$  therefore forces the requirement  $\mathcal{S}_e$ , hence  $\tau \in S_e$ .
- Case 2: there exists an input  $x$  such that for all the sets  $X \succeq \sigma$ ,  $\Phi_e^X(x) \uparrow$ . In this case, the string  $\sigma$  already forces the requirement  $\mathcal{S}_e$  ensuring that  $\Phi_e^B(x) \uparrow$ . So  $\sigma \in S_e$ .
- Case 3: neither of the two previous cases appears. We then showed in the initial proof of Proposition 4-8.2 that this case could not happen, because the set  $A$  would be computable, contrary to our hypothesis.

Let  $B$  be a  $(R_e, S_e)_{e \in \mathbb{N}}$ -generic set. Such a set exists by Theorem 2.4. In particular,  $B$  satisfies all the requirements  $\mathcal{R}_e$  and  $\mathcal{S}_e$  simultaneously, so  $B \not\leq_T A$  and  $A \not\leq_T B$ . This concludes the proof of Proposition 4-8.2. ■

## 2.2. The topological approach

The genesis of the finite extension method, of genericity, and more generally of forcing, can be found in the work of René Baire, at the beginning of the 20th century. One of Baire's motivations at the time was to understand a little better certain “bizarre” functions, but which arise naturally in analysis. If we go back a bit, in the first half of the 19th century, the awareness emerges, notably through the work of Cauchy and Bolzano, that a sequence of continuous functions  $(f_n : \mathbb{R} \rightarrow \mathbb{R})_{n \in \mathbb{N}}$  which pointwise converges does not necessarily have a continuous function as its limit, and for good reason: once the notions of computability have been cor-

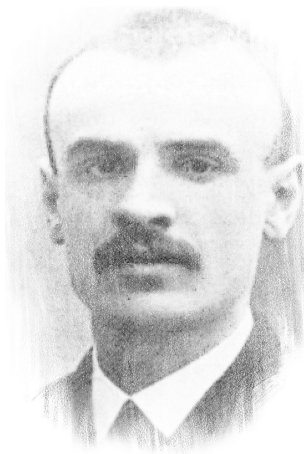
rectly extended to work in  $\mathbb{R}$ , the limits of *effectively continuous* functions — that is, computable — are exactly the  $\Delta_2^0$  functions, i.e., those for which  $f(r)$  is uniformly computable in  $r'$ , the jump of  $r$ . This is a simple application of Schoenfield's lemma.

A few decades later, Baire looks at this phenomenon, and tries to better understand these functions which are the limits of continuous functions, and whose irregularities make them difficult to manipulate and even to grasp with clarity. He will present his results in the Peccot course “Lessons on discontinuous functions” where he develops his famous mathematical tools of *Baire categories* to show in particular that a limit function of continuous functions, if it is no more necessarily continuous, will be continuous despite everything on a “large set of points”. According to modern terminology, the points of discontinuity of such a function will be a class *meager* or even of *category 1*. Theorem 3.20 to come can be seen as an actual version of this result.

We have defined in Section 8-2 the notion of open class of Cantor space, as being a class of the form  $\bigcup_{\sigma \in U} [\sigma]$  for any set  $U \subseteq 2^{<\mathbb{N}}$ . The density of a set of strings results in a notion of density of the corresponding open class in Cantor space.

**Definition 2.7.** A class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is said to be *dense* if it intersects any cylinder  $[\sigma]$ , ie  $[\sigma] \cap \mathcal{B} \neq \emptyset$  for all  $\sigma \in 2^{<\mathbb{N}}$ . ◇

The following exercise links the notion of density on the open classes of Cantor space, and on the partial order of binary strings.



René Baire, 1874–1932

**Exercise 2.8.** Given  $U \subseteq 2^{<\mathbb{N}}$  we denote by  $U^\prec$  the suffix closure of  $U$ , that is  $U^\prec = \{\tau \in 2^{<\mathbb{N}} : \exists \sigma \in U \ \tau \succeq \sigma\}$ .

Let  $U \subseteq 2^{<\mathbb{N}}$ . Show that  $U^\prec$  is dense in  $2^{<\mathbb{N}}$  iff the open class  $\bigcup_{\sigma \in U} [\sigma]$  is dense in Cantor space.  $\diamond$

Our goal is to define a notion of “negligible” or *meager* class, to give a topological definition of forcing. We start from the intuition that a cylinder  $[\sigma]$  is not thin. The notion of negligible class should be closed under subclass. Thus, if a class contains a non-empty open class of Cantor space, it is not negligible. To formalize this, we introduce the notion of interior.

**Definition 2.9.** We call *interior* of a class  $\mathcal{F} \subseteq 2^\mathbb{N}$  — which we denote by  $\text{int}(\mathcal{F})$  — the largest open class included in  $\mathcal{F}$ , that is, the union of all cylinders  $[\sigma]$  such that  $[\sigma] \subseteq \mathcal{F}$ .  $\diamond$

A negligible class must therefore in particular have an empty interior.

**Example 2.10.** The closed class  $\{X \in 2^\mathbb{N} : \forall n \ X(2n) = 0\}$  has an empty interior: it is indeed clear that it does not contain any cylinder  $[\sigma]$  because there are always  $X \in [\sigma]$  such that  $X(2n) = 1$  for  $n$  sufficiently large. Its complement  $\{X \in 2^\mathbb{N} : \exists n \ X(2n) \neq 0\}$  is therefore a dense open class, described by the union of the cylinders  $[\sigma 1]$  for any string  $\sigma$  of even size.

We now have the elements in hand to define the notion of meager class.

**Definition 2.11.** A class  $\mathcal{B} \subseteq 2^\mathbb{N}$  is said to be *meager* if  $\mathcal{B}$  is included in a countable union of closed classes with empty interior. The complement of a meager class is called *co-meager*.  $\diamond$

Note that by passing to the complement, a class is co-meager if it contains a countable intersection of dense open classes. We leave to the reader the care to show in the two following exercises that for a sequence  $\vec{W} = (W_n)_{n \in \mathbb{N}}$  of dense sets of strings, the class of  $\vec{W}$ -generic sets is a co-meager class.

**Exercise 2.12.** Let  $\vec{W} = (W_n)_{n \in \mathbb{N}}$  be a sequence of sets of strings such that each  $W_n^\prec$  is dense (using the notation of Exercise 2.8).

Show that the class  $\bigcap_n [W_n]$  is co-meager, where  $[W_n] = \bigcup_{\sigma \in W_n} [\sigma]$ .  $\diamond$

**Exercise 2.13.** Let  $\vec{W} = (W_n)_{n \in \mathbb{N}}$  be a sequence of dense sets of strings. Show that the class  $\bigcap_n [W_n]$  contains exactly the  $\vec{W}$ -generic sets.  $\diamond$

The two previous exercises therefore establish a link between the approach of forcing by the finite extension method and the topological approach:

if  $(\mathcal{R}_n)_{n \in \mathbb{N}}$  is a sequence of requirements such that the corresponding sets of strings  $(W_n)_{n \in \mathbb{N}}$  are dense, then the class of  $\vec{W}$ -generic — sets which satisfy all requirements simultaneously — is co-meager.

### Digression

Historically Baire calls *classes of category 1* the meager classes, and *classes of category 2* those that are not. This will give the name of “Baire category theory” to the study of these notions. Note that a class is not necessarily meager or co-meager. In particular, category 2 classes are not necessarily co-meager. As far as we are concerned, it is really the notions of meager and co-meager that interest us, and it is therefore this vocabulary that we will use.

### Fact

From Definition 2.11, it is clear that a countable union of meager classes is meager, and that a countable intersection of co-meager classes is co-meager.

One intuition that we will support in future developments is that meager classes are “small” and co-meager classes are “large”. The attribution of these adjectives should not be taken as absolute. There are other ways of judging class size, which do not coincide with the fact that the meager classes are small and the co-meager ones are large. One can for example find meager classes of measure 1 and co-meager classes of measure 0 (see Part II).

What should be understood rather, is that the co-meager classes are large enough to always be stable under countable intersection, and at the same time always dense, and even in a strong sense: if  $\mathcal{B}$  is a co-meager class then  $\mathcal{B} \cap [\sigma]$  is uncountable for any cylinder  $[\sigma]$ . This is in fact a reinforcement of Theorem 2.4. To see it, let us remember the following fact demonstrated with Proposition 8-2.3 and Proposition 8-3.2:

### Fact

The intersection of a finite number of open classes is open. Therefore we can always assume that a countable open intersection  $\bigcap_n \mathcal{U}_n$  is decreasing. By passing to the complement, one can always assume that a countable union of closed classes is increasing.

Recall that a class  $\mathcal{F} \subseteq 2^{\mathbb{N}}$  is *perfect* if it is the image of a continuous injection from  $2^{\mathbb{N}}$  to  $2^{\mathbb{N}}$  (see Section 8-2.4), that is,  $\mathcal{F}$  is of the form  $[T]$

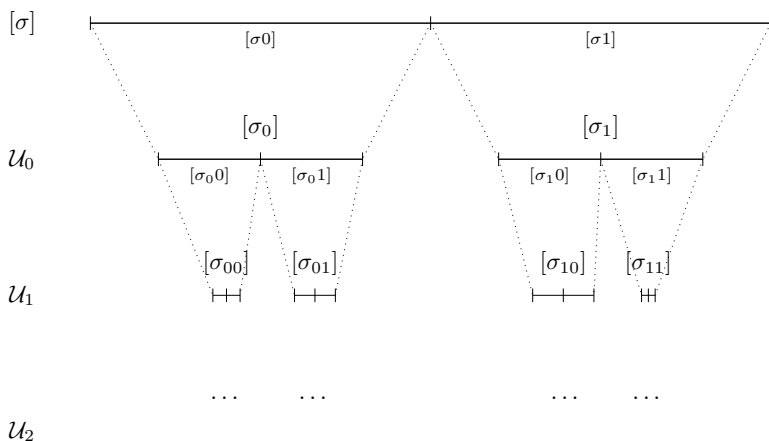


Figure 2.15: Illustration of the construction of a perfect sub-class of points in  $[\sigma] \cap \bigcap_n \mathcal{U}_n$ . As each open class  $\mathcal{U}_n$  is dense, one can find for every string  $\sigma_\tau i$  an extension  $\sigma_{\tau i} \succeq \sigma_\tau i$  such that  $[\sigma_{\tau i}] \subseteq \mathcal{U}_n$ .

for an f-tree  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  (see Section 7-5). The following lemma shows that a co-meager class has the power of the continuum.

**Lemma 2.14.** A co-meager class of Cantor space contains a perfect sub-class of points in each cylinder  $[\sigma]$ . ★

PROOF. Let  $\mathcal{B}$  be a co-meager class. Let  $\bigcap_n \mathcal{U}_n \subseteq \mathcal{B}$  be an intersection of dense open classes. We can assume without loss of generality  $\mathcal{U}_{n+1} \subseteq \mathcal{U}_n$ . The reader can use Figure 2.2 for a graphical representation of the following construction.

Let  $\sigma \in 2^{<\mathbb{N}}$ . Since  $\mathcal{U}_0$  is dense,  $\mathcal{U}_0 \cap [\sigma 0]$  is non-empty and so there is a string  $\sigma_0 \succ \sigma 0$  such that  $[\sigma_0] \subseteq \mathcal{U}_0$ . Likewise there is a string  $\sigma_1 \succ \sigma 1$  such that  $[\sigma_1] \subseteq \mathcal{U}_0$ . Suppose that for any string  $\tau$  of size  $n+1$  we have defined strings  $\sigma_\tau \succeq \sigma$  that are pairwise incomparable and such that  $[\sigma_\tau] \subseteq \mathcal{U}_n$ . For each of these strings  $\tau$  and for each  $i \in \{0, 1\}$  we define  $\sigma_{\tau i}$  as being a string which extends  $\sigma_\tau i$  and such that  $[\sigma_{\tau i}] \subseteq \mathcal{U}_{n+1}$ , which is possible because  $\mathcal{U}_{n+1}$  is dense.

It is clear that for any set  $X$  the class  $\bigcap_{\tau \prec X} [\sigma_\tau] \subseteq \bigcap_n \mathcal{U}_n$  contains a single element  $Y_X \in [\sigma] \cap \bigcap_n \mathcal{U}_n$ . We easily verify that the function  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  defined by  $T(\tau) = \sigma_\tau$  is an f-tree whose paths are the elements  $Y_X = T(\sigma_0) \prec T(\sigma_1) \prec \dots$  for  $X = \sigma_0 \prec \sigma_1 \prec \dots$ .

So  $\mathcal{B}$  contains a perfect subclass of points in the cylinder  $[\sigma]$ . ■

Note in particular as a consequence of the previous lemma that a countable union of closed classes with empty interior has empty interior: if such a union of closed classes contained a cylinder  $[\sigma]$ , its complement — a co-meager class — could not contain any point in  $[\sigma]$ , which would be a contradiction.

We now see the topological equivalent of Definition 2.1 of forcing for the  $\Sigma_1^0$  and  $\Pi_1^0$  requirements:

**Definition 2.16.** Let  $\mathcal{R}$  be a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement. Let  $\mathcal{B}_{\mathcal{R}}$  be the class of elements which satisfy  $\mathcal{R}$ . Then,  $\sigma$  forces  $\mathcal{R}$  iff  $[\sigma] \subseteq \mathcal{B}_{\mathcal{R}}$ . Note that if  $\mathcal{R}$  is  $\Sigma_1^0$  then  $\mathcal{B}_{\mathcal{R}}$  is an effectively open class and if  $\mathcal{R}$  is  $\Pi_1^0$  then  $\mathcal{B}_{\mathcal{R}}$  is an effectively closed class.  $\diamond$

Note for example that if the class of elements satisfying a  $\Pi_1^0$  requirement has empty interior, then no string forces this requirement: the open class of elements not satisfying the requirement is a dense, and will contain any sufficiently generic element. This brings us to the study of the sets which are in “sufficiently dense open classes”: the 1-generic and the weakly 1-generic, which we see now.

### 3. Effective genericity

Jockusch was undoubtedly one of the first to understand the usefulness of Cohen’s ideas in Computability Theory, and he initiated the study of an effective version of Cohen’s concepts, with the notions of 1-genericity and weak 1-genericity, resulting from the application of forcing to all  $\Sigma_1^0$  and  $\Pi_1^0$  requirements.

Genericity can be seen as both a notion of strength and weakness. A sufficiently generic set will, for example, always have hyperimmune degree. On the other hand, we will see that sufficiently generic sets cannot compute the halting problem, nor even a DNC function. Generally speaking, the finite extension method satisfies strength and weakness properties in the same way: by proving the density of some well-chosen sets.

#### 3.1. Weakly 1-generic sets

We begin by presenting the weakly 1-generic sets, introduced by Kurtz during his doctoral thesis, carried out under the supervision of Jockusch.

**Definition 3.1 (Kurtz [125]).** A set  $G \in 2^{\mathbb{N}}$  is *weakly 1-generic* if it is

generic for all dense c.e. sets. In other words,  $G$  is weakly 1-generic if  $G$  belongs to all the dense  $\Sigma_1^0$  classes of Cantor space.  $\diamond$

Note that there is only a countable quantity of dense  $\Sigma_1^0$  classes. The notion of weakly 1-generic set turns out not to be restrictive enough to hold properties normally inherent to generics, but in terms of Turing degree, the notion is of some interest, in particular via the following characterization.

**Theorem 3.2 (Kurtz [125])**

*Let  $G \subseteq \mathbb{N}$ . The following statements are equivalent:*

- (1)  $G$  is of hyperimmune degree.
- (2)  $G$  computes a function which is often infinitely equal to any computable function.
- (3)  $G$  computes a weakly 1-generic set.

PROOF. Let us first show (1)  $\rightarrow$  (3), the most difficult implication. Let  $f \leq_T G$  be a function which is not bounded by any computable function. We assume without loss of generality that  $f$  is increasing. Note that for any computable function  $g$ , there is an infinity of values  $n$  such that  $f(n) > g(n)$ . We compute from  $f$  a set  $G \in 2^{\mathbb{N}}$  which belongs to any dense  $\Sigma_1^0$  class. Let  $(W_e)_{e \in \mathbb{N}}$  be an enumeration of the  $\Sigma_1^0$  subsets of  $2^{<\mathbb{N}}$ . We construct  $G$  by successive approximation  $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ .

We first describe a recursive procedure to be performed each time we want to concatenate a string  $\tau$  to a string  $\sigma$  that we have so far computed. This procedure, which we will name  $R$ , takes a third parameter: an integer  $e$  which corresponds to the smallest integer such that  $\sigma\tau$  is enumerated in  $W_e$  at the computation step  $f(|\sigma|)$ . We will note  $R(\sigma, \tau, e)$  for the result of the call to this procedure. Finally, note that some integers are marked as “satisfied” when the procedure is called: these are the integers  $e$  such that  $\sigma$  extends a string of  $W_e$  at the current computation step.

The  $R(\sigma, \tau, e)$  procedure does the following: for each prefix  $\tau' \preceq \tau$  in order, it searches for the smallest integer  $e' < e$  that is not satisfied and such that a string of the form  $\sigma\tau'\rho$  is listed in  $W_{e'}[f(|\sigma\tau'|)]$ . If such an integer is found then the procedure returns the result of the recursive call to  $R(\sigma\tau', \rho, e')$ . Otherwise it returns  $\sigma\tau$ . Note that reducing the value of the last parameter in recursive calls causes the procedure to stop necessarily.

In step 0 we define  $\sigma_0 = \epsilon$ . Suppose  $\sigma_t$  defined in step  $t$ . In step  $t + 1$ , we look for the smallest unsatisfied integer  $e \leq t + 1$  such that a string of the form  $\sigma_t\tau$  is listed in  $W_e[f(|\sigma_t|)]$ . If we find such an integer  $e$  we define  $\sigma_{t+1}$  as being  $R(\sigma_t, \tau, e)$ . Otherwise  $\sigma_{t+1}$  to be  $\sigma_0$ . This concludes the construction.

Note that if  $W_e$  describes a dense open class, then the function  $f_e$  which to  $n$  associates the smallest computation time  $t$  such that all the strings of size  $n$  have an extension in  $W_e[t]$ , is a total computable function. We have in particular  $f_e(n) < f(n)$  for an infinity of values  $n$ . Suppose that  $W_e$  describes a dense open class, that  $e$  is not satisfied at time  $t$ , and that all  $e' < e$  which are satisfied at some point in the construction are satisfied at time  $t$ . Let  $n$  be the smallest integer greater than or equal to  $|\sigma_t|$  such that  $f(n) > f_e(n)$ . Let  $s \geq t$  be the smallest integer such that  $|\sigma_s| \leq n < |\sigma_{s+1}|$ . If  $|\sigma_s| = n$  then by minimality of  $e$ , the algorithm defines  $\sigma_{s+1} = \sigma_s \tau$  with  $\sigma_s \tau \in W_e[f(n)]$ . If not then by construction when defining  $\sigma_{s+1} = \sigma_s \tau$  for a certain string  $\tau$ , the algorithm checks for any prefix  $\tau' \preceq \tau$ , that we do not have an extension of  $\sigma_s \tau'$  listed in  $W_e[f(|\sigma_s \tau'|)]$ , and in particular for the prefix  $\tau'$  such that  $|\sigma_s \tau'| = n$ . If this is the case, the algorithm is restarted on this extension. As it is indeed the case by hypothesis, and by minimality of  $e$ , we will in fact have  $\sigma_s \tau \in W_e[f(n)]$  for  $\sigma_{s+1} = \sigma_s \tau$ . We conclude that the set  $G = \sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$  belongs to all the dense open classes.

Let us show (3)  $\rightarrow$  (2). Let  $G$  be a weakly 1-generic set. Note that if  $f$  is a total computable function then the set  $W_f = \{\sigma 0^{f(|\sigma|)} 1 : \sigma \in 2^{<\mathbb{N}}\}$  describes a dense  $\Sigma_1^0$  class. We compute from  $G$  the function  $g$  which to  $n$  associates the maximum number  $g(n)$  of 0 such that  $G \upharpoonright_n 0^{g(n)} \prec G$ . Since for any total computable function  $f$  the set  $G$  belongs to  $W_f$ , it is clear that  $g$  is equal at least once to any computable function. If  $g$  were equal only a finite number of times to a given computable function, one could modify a finite number of values of this function to have a computable function which is never equal to  $g$ . So  $g$  is infinitely often equal to any computable function.

Let us show (2)  $\rightarrow$  (1). Let  $f$  be a function infinitely equal to any computable function. Then,  $f + 1$  is infinitely often above any computable function. ■

We can relativize the notion of weak 1-genericity to any oracle:

**Definition 3.3 (Kurtz [125]).** Let  $A \in 2^{\mathbb{N}}$ . A set  $G$  is *weakly 1-generic relative to  $A$*  if  $G$  meets  $W$  for any  $\Sigma_1^0(A)$  dense set of strings  $W$ . ♦

The hierarchy of Turing jumps allows to define a hierarchy of genericity as follows:

**Definition 3.4 (Kurtz [125]).** A set  $G \in 2^{\mathbb{N}}$  is *weakly  $n$ -generic* if it is weakly 1-generic relative to  $\emptyset^{(n-1)}$ , that is to say if it meets all the  $\Sigma_1^0(\emptyset^{(n-1)})$  dense sets of strings, or equivalently all  $\Sigma_n^0$  dense sets of

strings. ◇

Some implications of Theorem 3.2 generalize to any oracle  $A$ :

**Exercise 3.5.** Show that any set  $G$  weakly 1-generic relative to  $A$  computes a function  $g : \mathbb{N} \rightarrow \mathbb{N}$  which is often infinitely equal to any  $A$ -computable function. ◇

**Exercise 3.6.** Let  $n \geq 1$  and  $G \in 2^{\mathbb{N}}$ . Show that if  $G$  computes a function which is infinitely often equal to all  $A$ -computable functions, then  $G$  computes an  $A$ -hyperimmune function. ◇

The direction (1)  $\rightarrow$  (3) of Theorem 3.2 does not hold in general. In the Turing jump hierarchy, this only works at the first level.

**Proposition 3.7 (Andrews, Gerdes and Miller [7]).** Any hyperimmune function relative to  $\emptyset'$  computes a weakly 2-generic set. ★

The idea to show the previous proposition is to use the fact that  $\emptyset'$ -computable objects are limit-computable. On the other hand, from  $n \geq 3$ , weakly  $n$ -generic sets can no longer be constructed simply using functions escaping collections of functions, in the following sense.

**Definition 3.8.** Let  $\mathcal{F}$  be a collection of functions. A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is  $\mathcal{F}$ -escaping if for all  $g \in \mathcal{F}$ , there exists  $n \in \mathbb{N}$  such that  $f(n) > g(n)$ . ◇

The following theorem expresses a deep structural difference between escaping functions and generic degrees, in the sense that whatever an oracle  $A$ , there exists an  $A$ -hyperimmune function which computes no weakly 3-generic set.

**Theorem 3.9 (Andrews, Gerdes and Miller [7])**

*For any countable collection of  $\mathcal{F}$  functions, there exists a  $\mathcal{F}$ -escaping function which is not of weakly 3-generic degree.*

**PROOF.** We will use a variant of the notion of f-tree defined in Section 7-5. We are going to define a total  $\Delta_3^0$  function  $T : \mathbb{N}^{<\mathbb{N}} \rightarrow \mathbb{N}^{<\mathbb{N}}$  such that:

- (1)  $\text{dom } T$  (the domain of  $T$ ) is a prefix-closed set;
- (2) for all  $\sigma, \tau \in \text{dom } T$ ,  $\sigma \preceq \tau$  if and only if  $T(\sigma) \preceq T(\tau)$ ;
- (3) for all  $\sigma \in \mathbb{N}^{<\mathbb{N}}$  and  $n \in \mathbb{N}$ , there exists an  $m \geq n$  such that  $T_s(\sigma n)$  extends  $T_s(\sigma) m$ .

The function  $T$  extends as a function from  $\mathbb{N}^{\mathbb{N}}$  to  $\mathbb{N}^{\mathbb{N}}$ , defining for all  $X \in \mathbb{N}^{\mathbb{N}}$ ,  $T(X)$  as the only element of the class  $\bigcap_{\sigma \prec X} [T(\sigma)]$ . A *path* of  $T$  is a

sequence  $P \in \mathbb{N}^{\mathbb{N}}$  of which an infinity of initial segments belong to  $\text{Im } T$ . In other words, a path is a sequence  $P \in 2^{\mathbb{N}}$  of the form  $P = T(X)$  for an  $X \in \mathbb{N}^{\mathbb{N}}$ . We denote by  $[T]$  the set of paths of  $T$ . By (3), for any countable collection of functions  $\mathcal{F}$ , there exists a path  $f \in [T]$  which is  $\mathcal{F}$ -escaping. We are going to construct  $T$  such that for any path  $f \in [T]$ ,  $f$  is not of weakly 3-generic degree.

The reader can use Figure 3.10 to understand the following construction. Let  $(\sigma_s)_{s \in \mathbb{N}}$  be a computable enumeration of  $\mathbb{N}^{<\mathbb{N}}$  which only shows each string after enumerating its prefixes. In particular,  $\sigma_0 = \epsilon$ . At the start of step  $s$ , we will have already defined  $T$  over  $\sigma_0, \dots, \sigma_s$ . We will also assume defined for each  $t \leq s$ , an infinite c.e. reservoir  $V_{t,s} \subseteq \mathbb{N}^{<\mathbb{N}}$  of strings extending  $T(\sigma_s)$ . We'll make sure that for all  $n \in \mathbb{N}$ ,  $T(\sigma_s n)$  extends a string of  $V_{t,s}$ . Simultaneously, we will create for each  $e$  a dense  $\Sigma_1^0(\emptyset'')$  set  $U_e \subseteq 2^{<\mathbb{N}}$  such that if  $\Phi_e^P$  is total for a path  $P \in [T]$ , then  $P$  does not meet  $U_e$ . Thus, whatever the path  $P \in [T]$ ,  $\Phi_e^P$  will not be a weakly 3-generic set.

Initially,  $f(\epsilon) = \epsilon$  and  $V_{0,0} = \mathbb{N}$ . At step  $s = \langle e, i \rangle$ , we will make sure that any string of length  $i$  has an extension in  $U_e$ . The set  $\{\sigma_t : t \leq s\}$  forms a finite tree, and for each leaf  $\tau$  of this tree,  $\Phi_e^{T(\tau)}$  can have different values. The set  $U_e$  must therefore add extensions to all strings of length  $i$ , while ensuring that it will avoid  $\Phi_e^{T(\sigma_t)}$  for all  $t \leq s$ . We therefore fix a sufficiently large length,  $k = i + s + 1$ , so that if we construct a set of “forbidden” binary strings  $\rho_0, \dots, \rho_s$  each of length  $k$ , any binary string of length  $i$  admits an extension of length  $k$  avoiding the forbidden strings.

For each  $t \leq s$ , we ask  $\emptyset''$  if there exists a string  $\rho_s$  of size  $k$  such that there is an infinity of binary strings  $\mu \in V_{t,s}$  having an extension  $\mu' \succeq \mu$  for which  $\Phi_e^{\mu'} \upharpoonright_k = \rho_s$ . If this is the case, we define  $V_{t,s+1}$  as the set of these  $\mu'$ . Note that  $V_{t,s+1}$  is then still computably enumerable. Otherwise for any string  $\rho$  of size  $k$ , only a finite number of strings of  $V_{t,s}$  have an extension which will be sent to an extension of  $\rho$  via  $\Phi_e$ . In particular, one can remove a finite number of strings from  $V_{t,s}$  so that no extension of the remaining strings will ever be sent to a string larger than  $k$ . We then take an arbitrary string  $\rho_s$ , and define  $V_{t,s+1}$  as the restriction of  $V_{t,s}$  to strings  $\mu$  which have no extension  $\mu'$  sent to a string larger in size than  $k$  via  $\Phi$ . Since we remove a finite number of strings,  $V_{s,t+1}$  is still c.e.

So we end up with a set of binary strings  $\rho_0, \dots, \rho_s$  each of length  $k$ , such that for any path  $P \in [T]$ , if  $\Phi_e^P$  is total, then  $\Phi_e^P \upharpoonright_k = \rho_t$  for one  $t \leq s$ . We list in  $U_e$  all the strings of length  $k$  other than  $\rho_0, \dots, \rho_s$ . Excluding these strings, we make sure that for any path  $P \in [T]$ , if  $\Phi_e^P$  is total, then it will not meet  $U_e$ . The length  $k$  being sufficiently large, any string of length  $i$  has an extension in  $U_e$ .

Below, the enumeration  $(\sigma_s)_{s \in \mathbb{N}}$  starts with  $\epsilon, 0, 1, 2, 00, 01, 10, 11, \dots$

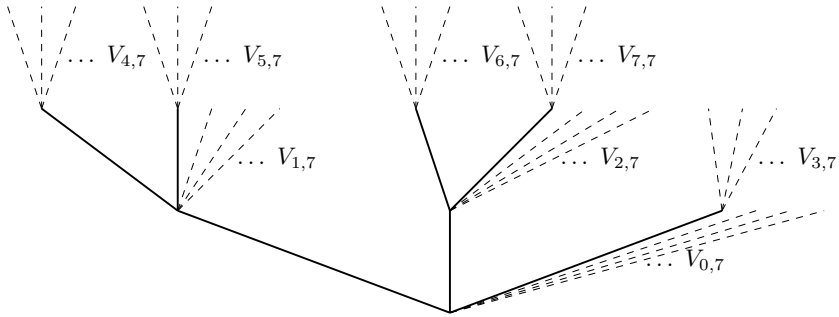


Figure 3.10: Illustration of the proof : at step 8 we will restrict each of the reservoirs from  $V_{0,7}$  to  $V_{7,7}$  by suppressing some of their branches, and by extending others, so that the current functional  $\Phi_e$  avoids, for each of the branches of the reservoirs  $V_{i,8}$ , a dense open class that we are currently enumerating thanks to  $\emptyset''$ . Once the reservoirs restricted, we complete our tree by taking an extension in each of them, which creates for each extension a new reservoir, and so on.

Finally, we define  $T(\sigma_{s+1})$ . Suppose that  $\sigma_{s+1} = \sigma_t n$  for a  $t \leq s$  and  $n \in \mathbb{N}$ . We choose  $\tau \in V_{t,s+1}$ , remove it from  $V_{t,s+1}$ , and we define  $T(\sigma_{s+1}) = \tau$ . Finally, we define  $V_{s+1,s+1} = \{\tau m : m \in \mathbb{N}\}$ . This completes the proof of Theorem 3.9. ■

**Exercize 3.11. (★)** Modify the proof of the previous theorem to show that for any countable collection of functions  $\mathcal{F}$ , there exists an  $\mathcal{F}$ -escaping function which does not compute any function which is often infinitely equal to any  $\emptyset''$ -computable function. ◇

### 3.2. 1-generic sets

We now see 1-genericity, a somewhat stronger and much richer notion than weak 1-genericity. The weakly 1-generic sets are those resulting from forcing for dense  $\Sigma_1^0$  requirements. But what about  $\Sigma_1^0$  requirements which are not dense?

The 1-genericity corresponds to the first level of forcing making it possible to tackle any  $\Sigma_1^0/\Pi_1^0$  requirements, via the following theorem, which will be demonstrated at the end of the subsection which follows (we refer the reader to Definition 2.1 for the notation  $\Vdash^*$ ):

**Theorem 3.12**

Let  $G$  be a 1-generic set. Let  $\mathcal{R}$  be a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement. Then,  $G$  satisfies  $\mathcal{R}$  iff there is a prefix  $\sigma \prec G$  such that  $\sigma \Vdash^* \mathcal{R}$ .

**3.2.1. Nature of 1-generic sets**

Recall the motivation of the definition of density: a dense set  $W$  is such that whatever the stage of the construction by the finite extension method, it is always possible to meet  $W$ . If  $W$  is not dense, then in the worst case, when we decide to meet it, our initial segment constructed so far may have no extension in  $W$ . This motivates the following definition.

**Definition 3.13.** A string  $\sigma \in 2^{<\mathbb{N}}$  *avoids* a set  $W \subseteq 2^{<\mathbb{N}}$  (denoted  $\sigma \perp W$ ) if not only  $\sigma \notin W$ , but also no extension of  $\sigma$  is in  $W$ .  $\diamond$

**Notation**

We note  $W^\perp = \{\tau \in 2^{<\mathbb{N}} : \tau \perp W\}$ .

**Lemma 3.14.** Let  $W \subseteq 2^{<\mathbb{N}}$  be an arbitrary set. Then the set  $W \cup W^\perp$  is dense.  $\star$

PROOF. Let  $\sigma \in 2^{<\mathbb{N}}$ . Either  $\sigma$  has an extension in  $W$ , and therefore in  $W \cup W^\perp$ , or  $\sigma$  avoids  $W$ , in which case  $\sigma \in W^\perp$ .  $\blacksquare$

Note that if  $W$  is a dense set, then  $W^\perp = \emptyset$ . Remember that the density of a set of strings  $W$  closed under suffix on  $2^{<\mathbb{N}}$  corresponds to the density on  $2^\mathbb{N}$  of its corresponding open class  $[W] = \bigcup_{\sigma \in W} [\sigma]$ . The set  $W^\perp$  also corresponds to an open one: the interior of the complement of the set  $[W]$ , that is to say the union of all the cylinders  $[\sigma]$  included in the complement of  $[W]$ .

If we reformulate it in Cantor space, Lemma 3.14 states that for any open class  $\mathcal{U} \subseteq 2^\mathbb{N}$ , the class  $\mathcal{U} \cup \text{int}(2^\mathbb{N} \setminus \mathcal{U})$  is a dense open class. If  $\mathcal{U}$  is basically dense then  $\text{int}(2^\mathbb{N} \setminus \mathcal{U}) = \emptyset$ , otherwise we “densify”  $\mathcal{U}$  by adding the interior of its complement. We are now ready to define the notion of 1-generic set.

**Definition 3.15 (Jockusch [99]).** A set  $G \in 2^\mathbb{N}$  is 1-generic if it is  $\{W_e \cup W_e^\perp : e \in \mathbb{N}\}$ -generic, where  $W_e$  is the computably enumerable set of strings of code  $e$ . Equivalently,  $G$  is 1-generic if  $G \in \mathcal{U} \cup \text{int}(2^\mathbb{N} \setminus \mathcal{U})$  for any  $\Sigma_1^0$  class  $\mathcal{U}$ .  $\diamond$

As mentioned above, if  $W \subseteq 2^{<\mathbb{N}}$  is a dense set, then  $W^\perp = \emptyset$ . It follows that every 1-generic set meets every dense  $\Sigma_1^0$  set, and is therefore weakly

1-generic. In particular, the degrees of weakly 1-generic sets coinciding with hyperimmune degrees, any 1-generic set is of hyperimmune degree, and therefore not computable.

If we look at the contraposition of the notion of 1-genericity, a set  $G$  is not 1-generic if there exists a c.e. set  $W \subseteq 2^{<\mathbb{N}}$  containing no prefix of  $G$  and such that  $W$  “is dense along  $G$ ”, that is, for all  $\sigma \prec G$  there exists  $\tau \succeq \sigma$  with  $\tau \in W$  and  $\tau \not\prec G$ . This idea is illustrated in Figure 3.16.

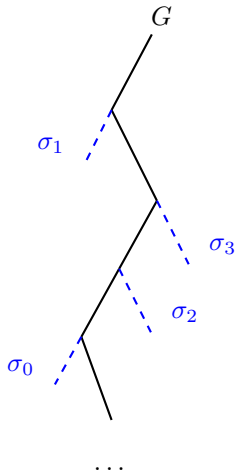


Figure 3.16: Illustration of a set  $G$  which is not 1-generic : one can enumerate strings  $\sigma_0, \sigma_1, \dots$  densely along  $G$ , without ever enumerating a prefix of  $G$ .

We now see formally why Theorem 3.12 claimed above is true.

**Proposition 3.17.** Let  $\mathcal{R}_e$  be a  $\Sigma_1^0$  requirement. Then the set  $W \subseteq 2^{<\mathbb{N}}$  of strings forcing  $\mathcal{R}_e$  is  $\Sigma_1^0$ . Moreover  $W^\perp$  is the set of strings forcing  $\neg\mathcal{R}_e$  and it is  $\Pi_1^0$ . ★

PROOF.  $\mathcal{R}_e$  being  $\Sigma_1^0$ , it can be written in the form  $\Phi(G, 0) \downarrow$  for a functional  $\Phi$ . A string  $\sigma$  forces  $\mathcal{R}_e$  if  $\Phi(X, 0) \downarrow$  for all  $X \in [\sigma]$ . By König’s lemma,  $\sigma$  forces  $\mathcal{R}_e$  if there exists a length  $n$  such that for all  $\tau \succeq \sigma$  of length  $n$ ,  $\Phi(\tau, 0) \downarrow$ . The set

$$W = \{\sigma \in 2^{<\mathbb{N}} : \exists n \forall \tau \in 2^n (\tau \succeq \sigma \rightarrow \Phi(\tau, 0) \downarrow)\}$$

is  $\Sigma_1^0$  (where  $2^n$  denotes the set of strings of size  $n$ ).

Let us show that  $W^\perp$  is the set of strings  $\sigma$  which force  $\neg\mathcal{R}_e$ . If  $\sigma$  does not force  $\neg\mathcal{R}_e$ , then there exists an  $X \in [\sigma]$  such that  $\Phi(X, 0) \downarrow$ . By the use property, for  $n$  sufficiently large and for all  $Y \in [X \upharpoonright_n]$ ,  $\Phi(Y, 0) \downarrow$ . We can assume  $n \geq |\sigma|$ . Let  $\tau = X \upharpoonright_n$ . Then,  $\tau \in W$ , so  $\sigma \notin W^\perp$ . By contraposition, if  $\sigma \in W^\perp$ , then  $\sigma$  forces  $\neg\mathcal{R}_e$ . Now suppose that  $\sigma \notin W^\perp$ . Then, there exists  $\tau \succeq \sigma$  such that  $\tau \in W$ . In particular, there is an  $X \in [\tau] \subseteq [\sigma]$  such that  $\Phi(X, 0) \downarrow$ . It follows that  $\sigma$  does not force  $\neg\mathcal{R}_e$ .

As the set  $W$  is  $\Sigma_1^0$ , it is clear that the set  $W^\perp$  is  $\Pi_1^0$ . ■

**Corollary 3.18**

*Let  $\mathcal{R}_e$  be a  $\Sigma_1^0$  requirement. Then, the set  $D$  of strings which force  $\mathcal{R}_e$  or which force  $\neg\mathcal{R}_e$  is dense.*

PROOF. Immediate by Lemma 3.14 and Proposition 3.17. ■

Note that any set satisfies the formula “ $\mathcal{R}_e \vee \neg\mathcal{R}_e$ ”, and therefore that any string forces “ $\mathcal{R}_e \vee \neg\mathcal{R}_e$ ”. On the other hand, it is much stronger for a string  $\sigma$  to force  $\mathcal{R}_e$  or to force  $\neg\mathcal{R}_e$ , because any set  $A \in [\sigma]$  must have the same behavior towards  $\mathcal{R}_e$ .

We can now show Theorem 3.12 announced: if  $G$  is a 1-generic set and  $\mathcal{R}$  a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement, then  $G$  satisfies  $\mathcal{R}$  iff there is a  $\sigma \prec G$  prefix such that  $\sigma \Vdash^* \mathcal{R}$ .

PROOF OF THEOREM 3.12. Let  $G$  be a 1-generic set. Let  $\mathcal{R}$  be a  $\Sigma_1^0$  requirement. If a prefix  $\sigma \prec G$  forces  $\mathcal{R}$  or  $\neg\mathcal{R}$  then by definition  $G$  satisfies respectively  $\mathcal{R}$  or  $\neg\mathcal{R}$ .

Conversely, suppose that  $G$  satisfies  $\mathcal{R}$ . Let  $W$  be the c.e. set of strings that force  $\mathcal{R}$ . Since  $G$  is 1-generic it meets  $W \cup W^\perp$ . If  $G$  meets  $W^\perp$  then  $\sigma$  forces  $\neg\mathcal{R}$  for a prefix  $\sigma \prec G$  and therefore  $G$  satisfies  $\neg\mathcal{R}$  which is a contradiction. So  $G$  meets  $W$  and a prefix  $\sigma \prec G$  forces  $\mathcal{R}$ .

Symmetrically if  $G$  satisfies  $\neg\mathcal{R}$  then a prefix  $\sigma \prec G$  forces  $\neg\mathcal{R}$ . ■

### 3.2.2. Properties of 1-generic sets

In general, the function which to  $X$  associates  $X'$  is not continuous. Indeed, it is sometimes necessary to know an infinity of bits of  $X$  to know if  $n \in X'$ . René Baire showed — in another form of course — that this function was, on the other hand, continuous over a co-meager class of points. The 1-genericity is the level of genericity required to account for this theorem.

**Definition 3.19.** A set  $G \in 2^\mathbb{N}$  is *generalized low* if

$$G' \leq_T G \oplus \emptyset'$$

◇

If a set  $G$  is generalized low, then not only the function which to  $X$  associates  $X'$  is continuous in  $G$ , but even more it is computable in  $G \oplus \emptyset'$ . Any set has a priori no reason to be generalized low. For example  $\emptyset'$  is not, but it is the case for 1-generic sets.

**Theorem 3.20**

*The 1-generic sets are generalized low.*

PROOF. For any  $e$ , define the class  $\mathcal{U}_e = \{X : \Phi_e(X, e) \downarrow\}$ . Let  $W_e \subseteq 2^{<\mathbb{N}}$  be a  $\Sigma_1^0$  set which represents  $\mathcal{U}_e$ , i.e., such that  $[W_e] = \mathcal{U}_e$  (where  $[W_e] = \bigcup_{\sigma \in W_e} [\sigma]$ ). Note that  $e \in G'$  iff  $G \in \mathcal{U}_e$ . We also have  $G \in \mathcal{U}_e$  iff there exists  $\tau \preceq G$  such that  $\tau \in W_e$ , and by definition of the 1-genericity of  $G$ , we have  $G \notin \mathcal{U}_e$  iff there is  $\tau \preceq G$  such that  $\sigma \notin W_e$  for any  $\sigma$  compatible with  $\tau$ . The question of whether  $\sigma \notin W_e$  for any  $\sigma$  compatible with  $\tau$  is  $\Pi_1^0$  uniformly in  $\tau$  and therefore can be asked to  $\emptyset'$ . To know if  $e \in G'$ , it suffices therefore to look for a prefix  $\tau$  of  $G$  such that we are in one case or the other, which will necessarily happen. ■

We have seen that any 1-generic set has a hyperimmune degree, and therefore cannot be computed. Therefore, 1-genericity is not a weakness property. We will now see that it is not a strength property either, in the sense that certain computational powers can never be reached by the 1-generic degrees. In particular, no 1-generic degree computes  $\emptyset'$ .

**Theorem 3.21 (Demuth and Kučera [44])**

*No 1-generic set is of DNC degree.*

PROOF. Suppose  $n \mapsto \Phi(G, n)$  is a DNC function. We consider the class  $\mathcal{U} = \{X : \exists n \Phi(X, n) \downarrow = \Phi_n(n) \downarrow\}$ . By hypothesis  $G \notin \mathcal{U}$ . Consider a string  $\sigma$ . We define by Kleene's fixed point theorem the code  $e_\sigma$  of the function which, on the input  $e_\sigma$ , searches for an extension  $\tau_\sigma \succeq \sigma$  such that  $\Phi(\tau_\sigma, e_\sigma) \downarrow$  and assigns  $\Phi(\tau_\sigma, e_\sigma)$  to  $\Phi_{e_\sigma}(e_\sigma)$ . The process being uniform, we enumerate all the strings of the form  $\tau_\sigma$  in a c.e. set  $W$ .

Note that for all  $\sigma \prec G$  the function code  $e_\sigma$  halts on the input  $e_\sigma$  because there exists at least one extension of  $\sigma$  —namely  $G$  itself— for which  $\Phi$  halts on the input  $e_\sigma$ . Since  $n \mapsto \Phi(G, n)$  is DNC, we have  $\Phi(G, e_\sigma) \neq \Phi_{e_\sigma}(e_\sigma)$  and therefore  $\sigma \prec \tau_\sigma \not\prec G$ .

We therefore have a c.e. set  $W$  of which no element is a prefix of  $G$ , but which contains an extension of each prefix of  $G$ . It follows that  $G$  is not 1-generic. ■

The restriction of genericity makes it possible to construct sets benefiting from certain advantages of genericity while not being too complex from a computational point of view.

**Exercise 3.22. (★)** Show by a direct argument that no 1-generic set is computable. ◇

**Exercise 3.23. (★)** Show that there exists a 1-generic  $\Delta_2^0$  set. Deduce that there exists a 1-generic set of low degree.  $\diamond$

We will see in the next section with Corollary 3.34 that there are also 1-generic sets of high degree. Let us end with an interesting exercise, which requires elaborating on the techniques of Theorem 3.28, and which uses the notion of effective join of Definition 4-5.6, but extended to a countable sequence of sets:

**Definition 3.24.** The *effective join*  $\bigoplus_{n \in \mathbb{N}} A_n$  of a sequence of sets  $(A_n)_{n \in \mathbb{N}}$  is the set  $Y$  such that  $\langle n, m \rangle \in Y$  iff  $m \in A_n$ .  $\diamond$

In the following exercises, the notation  $\bigoplus_{j \neq i} G_j$  therefore corresponds to the effective join of the sequence  $(G_n)_{n \in \mathbb{N}}$  from which we remove the set  $G_i$ .

**Exercise 3.25. (★★)** Let  $G = \bigoplus_{n \in \mathbb{N}} G_n$  be a 1-generic set. Show that  $G_i$  is hyperimmune relative to  $\bigoplus_{j \neq i} G_j$  for all  $i \in \mathbb{N}$ .  $\diamond$

**Exercise 3.26. (★★) (Miller).** Let  $X = \bigoplus_{n \in \mathbb{N}} X_n$  be a set such that  $X_i$  is hyperimmune relative to  $\bigoplus_{j \neq i} X_j$  for all  $i \in \mathbb{N}$ . Show that  $X$  computes a 1-generic set.  $\diamond$

### 3.2.3. Relativization of 1-generic sets

Just as we relativized weak 1-genericity, we now relativize 1-genericity.

**Definition 3.27 (Jockusch [99]).** Let  $A \in 2^{\mathbb{N}}$ . A set  $G \in 2^{\mathbb{N}}$  is 1-*generic* relative to  $A$  if  $G \in \mathcal{U} \cup \text{int}(2^{\mathbb{N}} \setminus \mathcal{U})$  for any class  $\mathcal{U}$  which is  $\Sigma_1^0(A)$ . We will say that  $G$  is  $n$ -generic if it is 1-generic with respect to  $\emptyset^{(n-1)}$ , or in an equivalent way if it meets  $W \cup W^\perp$  for any  $\Sigma_n^0$  set of strings  $W$ .  $\diamond$

We will study this relativization in more detail in Section 5. Let us see for the moment a first key theorem.

#### Theorem 3.28

Let  $X$  be non-computable and  $G$  be a 1-generic set relative to  $X$ . Then, we have  $G \not\leq_T X$ .

**PROOF.** We are going to build a  $\Sigma_1^0(X)$  class  $\mathcal{U}$  such that  $Y \in \mathcal{U} \cup \text{int}(2^{\mathbb{N}} \setminus \mathcal{U})$  implies  $\Phi(Y) \neq X$ .

We simply enumerate in the set  $X$ -c.e. which describes  $\mathcal{U}$ , all strings  $\sigma$  such that  $\exists n \Phi(\sigma, n) \downarrow \neq X(n)$ . Suppose now that  $\tau$  is a string for which  $[\tau] \subseteq 2^{\mathbb{N}} \setminus \mathcal{U}$ , that is, no extension  $\sigma \succeq \tau$  is such that  $\exists n \Phi(\sigma, n) \downarrow \neq X(n)$ . Let us show that for all  $Y \succeq \tau$  we have  $\exists n \Phi(Y, n) \uparrow$ . Suppose this is not the case.

Then, we can compute  $X$  as follows: to know  $X(n)$ , it suffices to look for an extension  $\sigma \succeq \tau$  such that  $\Phi(\sigma, n) \downarrow$ . By hypothesis, such an extension exists, and still by hypothesis, it is such that  $\Phi(\sigma, n) \downarrow = X(n)$ . As this is true for every  $n$ , this contradicts the fact that  $X$  is non-computable. So if  $[\tau] \subseteq 2^{\mathbb{N}} \setminus \mathcal{U}$  then for all  $Y \succeq \tau$  we have  $\exists n \Phi(Y, n) \uparrow$ . We deduce that for all  $Y \in \mathcal{U} \cup \text{int}(2^{\mathbb{N}} \setminus \mathcal{U})$  we have  $\Phi(Y) \neq X$ . As  $G$  is 1-generic relative to  $X$  then  $G \in \mathcal{U} \cup \text{int}(2^{\mathbb{N}} \setminus \mathcal{U})$  and the same is true for any functional  $\Phi$ . So  $G \not\leq_T X$ . ■

Let us now see the various implications between the notions of  $n$ -genericity and  $n$ -weak genericity.

**Proposition 3.29.** Let  $G$  be a set. Then, for all  $n > 0$ ,  $G$  weakly  $(n+1)$ -generic implies  $G$   $n$ -generic implies  $G$  weakly  $n$ -generic. The implications are strict. ★

PROOF. If  $G$  is weakly  $(n+1)$ -generic, then it meets any  $\Sigma_1^0(\emptyset^{(n)})$  dense set. For any  $\Sigma_1^0(\emptyset^{(n-1)})$  set  $W$ ,  $W \cup W^\perp$  is dense and  $\Sigma_1^0(\emptyset^{(n)})$ , so  $G$  meets  $W \cup W^\perp$ . Thus  $G$  is  $n$ -generic. If  $G$  is  $n$ -generic, then for any  $\Sigma_1^0(\emptyset^{(n-1)})$  set  $W$ ,  $G$  meets  $W \cup W^\perp$ . If  $W$  is dense, then  $W^\perp = \emptyset$ , so  $G$  meets  $W$ . Thus  $G$  is weakly  $n$ -generic.

To see that the first implication is strict, it suffices to construct an  $n$ -generic  $\emptyset^{(n)}$ -computable set and to see that no weakly  $(n+1)$ -generic set is  $\emptyset^{(n)}$ -computable because  $\{\sigma : \sigma \not\leq X\}$  is a  $\Sigma_{n+1}^0$  dense set of strings for any set  $X$  which is  $\Delta_{n+1}^0$ . We can consult Exercize 3.30 for an example of an  $n$ -generic set which is not weakly  $n$ -generic. ■

**Exercize 3.30. (★★)** A set  $X$  is *left-c.e.* relative to  $A$  if there exists an  $A$ -computable sequence of sets  $(X_s)_{s \in \mathbb{N}}$  such that  $X_s$  is lexicographically smaller than  $X_{s+1}$  for all  $s$ , and such that  $\lim_s X_s = X$ .

Show that for all  $A$  there exists a weakly 1-generic set relative to  $A$  and left-c.e. relative to  $A$ . Show that no 1-generic set relative to  $A$  is left-c.e. relative to  $A$ . ◇

### 3.3. Posner/Robinson theorem

In a 1981 article, Posner and Robinson study degrees strictly under the halting set, but the join of which makes it possible to compute the halting set. A generalized and modern version of their main theorem is the following: for any set non-computable  $A$  — and in particular as “weak” as computably possible — there exists a set  $G$  such that  $A \oplus G \geq_T G'$ . Informally, there always exists a set  $G$  whose computational distance between itself and its jump, is “reduced” to  $A$ , and this for any  $A$ .

The modern presentation of the theorem is that of Jockusch and Shore, who show something more general:

**Theorem 3.31 (Jockusch and Shore [101])**

*Let  $A, Z$  be non-computable sets. There exists a 1-generic set  $G$  such that  $A \oplus G \geq_T Z$ . Moreover, we can obtain  $G$  in a computable way from  $A \oplus Z \oplus \emptyset'$ .*

PROOF. The idea is to build a 1-generic set  $G$ , which will encode  $Z$ , so that  $G$  and  $A$  allow to find the construction sequence. The construction itself will be computable in  $A \oplus Z \oplus \emptyset'$ . We can assume without loss of generality that  $A$  is not a c.e. set (otherwise, one replaces  $A$  by its complement). Let  $(W_e)_{e \in \mathbb{N}}$  be an enumeration of the  $\Sigma_1^0$  subsets of  $2^{<\mathbb{N}}$ .

We define  $\sigma_0 = \epsilon$ , the empty word. Suppose  $\sigma_e$  defined. We consider the set

$$D_e = \{m : \exists \tau \text{ such that } \sigma_e Z(e) 0^m 1 \tau \in W_e\}.$$

Note that  $D_e$  is a c.e. set. In particular as  $A$  is not c.e. there is some  $m \in D_e$  with  $m \notin A$  or some  $m \notin D_e$  with  $m \in A$ . We consider the smallest  $m$  such that we are in one case or the other. Note that  $\emptyset' \oplus A$  allows to find uniformly this integer  $m$ .

In the first case, we define  $\sigma_{e+1}$  as being  $\sigma_e Z(e) 0^m 1 \tau$  for the first string  $\tau$  such that  $\sigma_e Z(e) 0^m 1 \tau$  is listed in  $W_e$ . In the second case, we define  $\sigma_{e+1}$  as being  $\sigma_e Z(e) 0^m 1$ . Note that in this case no string of  $W_e$  can extend  $\sigma_{e+1}$ . We define  $G$  as being  $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ . This completes the construction.

It is clear that  $G$  is 1-generic and computable in  $A \oplus Z \oplus \emptyset'$ . How do you now compute  $Z$  from  $G \oplus A$ ? Suppose we know the string  $\sigma_e$ . We then necessarily know the  $e$ -th bit of  $Z$ : it is the bit  $i$  such that  $\sigma_e i \prec G$ . We can then find  $\sigma_{e+1}$  as follows: we look at the number  $m$  of 0 which follows  $\sigma_e i$  in  $G$ . If  $m \in A$ , this means that  $\sigma_{e+1} = \sigma_e i 0^m 1$ . If  $m \notin A$ , this means that  $\sigma_{e+1} = \sigma_e i 0^m 1 \tau$  for the first string  $\tau$  found in  $W_e$ . Finding this string  $\tau$  is then a computable process. We can therefore in all cases find  $\sigma_{e+1}$ , and by repeating the process, compute  $Z$  from  $A \oplus G$ . ■

**Corollary 3.32 (Posner and Robinson [179])**

*Let  $A$  be a non-computable set. There exists a set  $G$  such that  $A \oplus G \geq_T G'$ . If  $A$  is  $\Delta_2^0$  then we have  $A \oplus G \equiv_T G'$ .*

PROOF. We apply the previous theorem with  $Z = \emptyset'$ . We therefore have a 1-generic set  $G$  such that  $G \leq_T \emptyset' \oplus A$  and such that  $G \oplus A \geq_T \emptyset'$ . By using the fact that the 1-generics are generalized low, we thus have  $G \oplus A \geq_T G \oplus \emptyset' \equiv_T G'$ . It is clear that if  $A$  is  $\Delta_2^0$  then  $A \oplus G \equiv_T G'$ . ■

Another interesting corollary is the jump inversion theorem: any set that computes the halting problem can be seen as the Turing degree of the jump of a set.

**Corollary 3.33 (Jump inversion theorem, Friedberg [61])**

*Let  $Z \geq_T \emptyset'$ . There exists a set  $G$  such that  $Z \equiv_T G'$ .*

PROOF. We apply the previous theorem with  $Z$  and  $A = \emptyset'$ . We therefore have a 1-generic set  $G$  such that  $G \oplus \emptyset' \leq_T Z \oplus \emptyset' \equiv_T Z$  and such that  $G \oplus \emptyset' \geq_T Z$ . By using the fact that the 1-generics are generalized low, we thus have  $G' \equiv_T Z$ . ■

Theorem 3.31 also allows us to deduce the existence of high degrees which are Turing incomplete, and even of non-DNC degree.

**Corollary 3.34**

*There is a high set of non-DNC degree, and in particular Turing incomplete.*

PROOF. It suffices to apply Theorem 3.31 to find a 1-generic set  $G$  such that  $\emptyset' \oplus G \geq_T \emptyset''$ , which implies  $G' \geq_T \emptyset''$ . Note that as  $G$  is 1-generic, it is not of DNC degree and in particular does not compute  $\emptyset'$ . ■

### 3.4. Meager/co-meager nature of computational properties

Let us go back for a moment to the topological notions of meager and co-meager classes introduced by Baire. We have already mentioned that not every class is necessarily meager or co-meager. On the other hand, it is the case for the classes having good closure properties, and in particular for all those known as Borel (we will precisely define this term in Chapter 17) and closed under Turing equivalence. This will be the case in particular for all the notions of computability that we will see, and we will ask ourselves the question of whether these latter are meager or co-meager. Also a class is without loss of generality co-meager if it contains any *sufficiently generic* element. In practice, 1-genericity is sufficient for the various computational properties seen so far.

We have the high degrees — the low degrees being a countable class, they are not of interest — the computably dominated vs hyperimmune degrees, and finally the DNC and PA degrees. Is the class of sets of each of these degrees meager or co-meager? We already have the answer when it comes to computably dominated and hyperimmune degrees:

**Proposition 3.35.** The class of computably dominated sets is meager. That of hyperimmune sets is co-meager. ★

PROOF. According to Theorem 3.2 if  $G$  is weakly 1-generic then it is not of computably dominated degree. ■

**Proposition 3.36.** The class of DNC sets is meager, as is of course the class of PA sets. ★

PROOF. According to Theorem 3.21 if  $X$  is 1-generic then it is not of DNC degree, and the class of 1-generic sets is co-meager. ■

We are now tackling the high degrees. For this, we use our cone avoidance theorem 3.28.

**Proposition 3.37.** If  $X$  is non-computable, then the class of sets which compute  $X$  is meager. ★

PROOF. The class of 1-generic sets relative to  $X$  is an intersection of dense open classes, and is therefore co-meager. By Theorem 3.28, none of them computes  $X$  and therefore the class of sets computing  $X$  is in its complement, a meager class. ■

We can finally show by combining what we have seen that the class of high sets is also meager:

**Proposition 3.38.** The class of high sets is meager. ★

PROOF. For this, we use a relativized version of Theorem 3.28: if  $X$  is not  $\emptyset'$ -computable then for any set  $G$  which is 1-generic relative to  $X \oplus \emptyset'$  we have  $G \oplus \emptyset' \not\geq_T X$ . This relativized version is shown in the same way and does not present any particular difficulty.

We now use Theorem 3.20: if  $G$  is 1-generic then  $G' \leq_T G \oplus \emptyset'$ . We deduce that if  $G$  is 1-generic then  $G$  is high iff  $G \oplus \emptyset' \geq_T \emptyset''$ . Moreover, according to the relativized version of Theorem 3.28, no 1-generic relative to  $\emptyset''$  is such that  $G \oplus \emptyset' \geq_T \emptyset''$ . As any 1-generic set relative to  $\emptyset''$  is also 1-generic, one therefore has that no 1-generic relative to  $\emptyset''$  is high. So the class of high sets is meager. ■

## 4. $\Sigma_n^0/\Pi_n^0$ forcing

The concept of 1-genericity can be seen as an effective formal framework around the finite extension method, which enables to control the halting

or not of functionals, that is to say to control the truth value of  $\Sigma_1^0$  or  $\Pi_1^0$  predicates. This is a first level of forcing: given a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement  $\mathcal{R}_e$ , according to Corollary 3.18, any string  $\sigma$  admits an extension  $\tau \succeq \sigma$  such that any set  $X \in [\tau]$  satisfies  $\mathcal{R}_e$  or such that any set  $X \in [\tau]$  satisfies  $\neg\mathcal{R}_e$ .

From level  $\Sigma_2^0/\Pi_2^0$ , things get more complex and the finite extension method can no longer work in the same way, as the following example shows.

**Example 4.1.** Consider the requirement  $\mathcal{R}$ : “ $\exists x \forall y \geq x \ G(y) = 0$ ” which expresses the finiteness of the set  $G$ . For all  $\sigma \in 2^{<\mathbb{N}}$ ,  $[\sigma]$  contains both a finite set and an infinite set. There is therefore no cylinder of which all the elements satisfy  $\mathcal{R}$  or of which all the elements satisfy  $\neg\mathcal{R}$ .

### Check-in on requirements

We now tackle the  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) requirements, that is to say the requirements relating to a set  $G$  and which are expressed by a  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formula of second-order arithmetic, with  $G$  as a free set variable. Equivalently, a requirement is  $\Sigma_n^0$  if there exists a functional  $\Phi_e(G, x_1, \dots, x_{n-1})$  such that the sets  $G$  satisfying the requirement are those such that:

$$\begin{aligned} \exists x_1 \forall x_2 \dots \forall x_{n-1} \Phi_e(G, x_1, x_2, \dots, x_{n-1}) \downarrow & \text{ for } n \text{ odd} \\ \exists x_1 \forall x_2 \dots \exists x_{n-1} \Phi_e(G, x_1, x_2, \dots, x_{n-1}) \uparrow & \text{ for } n \text{ even.} \end{aligned}$$

The  $\Pi_n^0$  requirements have the analogous equivalence starting with a universal quantification. By convention the negation  $\neg\mathcal{R}$  in front of a  $\Sigma_n^0$  formula (resp.  $\Pi_n^0$ ) will not be considered as a symbol of the language, but as a transformation operation over  $\mathcal{R}$ , which reverses the quantifiers, and replaces the final  $\Delta_0^0$  predicate by its negation, to make  $\neg\mathcal{R}$  a  $\Pi_n^0$  formula (resp.  $\Sigma_n^0$ ).

We must therefore abstract things a little, in order to give a more general definition of forcing allowing to control the truth value of predicates of arbitrary complexity. Before we begin, let's introduce a definition that will be used in future proofs.

**Definition 4.2.** A set  $D \subseteq 2^{<\mathbb{N}}$  is *dense below*  $\sigma$  if for all  $\tau \succeq \sigma$ , there exists a  $\rho \succeq \tau$  such that  $\rho \in D$ .  $\diamond$

### 4.1. The semantic approach

The forcing relation will be defined by induction on  $n$ , between finite strings and  $\Sigma_n^0$  predicates. One of the objectives will then be to keep the following property.

(D): The set of strings forcing  $\mathcal{R}$  or forcing  $\neg\mathcal{R}$  is dense.

In order to examine what we need, let's take the previous example, namely an arbitrary  $\Sigma_2^0$  requirement  $\mathcal{R}$  “ $\exists x \Phi(G, x) \uparrow$ ”, and see what does not work when we try to prove the density of the set  $Q \subseteq 2^{<\mathbb{N}}$  of strings all of whose elements satisfy  $\mathcal{R}$  or all of whose elements satisfy  $\neg\mathcal{R}$ .

Let  $\sigma \in 2^{<\mathbb{N}}$  be a string. Let's do a case analysis. Case 1: there is an extension  $\tau \succeq \sigma$  and an  $x \in \mathbb{N}$  such that for all  $\rho \succeq \tau$ ,  $\Phi(\rho, x) \uparrow$ . In this case, by the use property, for all  $A \in [\tau]$ ,  $\Phi(A, x) \uparrow$ . It follows that  $\mathcal{R}$  is satisfied for all  $A \in [\tau]$ , so that  $\tau \in Q$ . Case 2: for any extension  $\tau \succeq \sigma$  and any  $x \in \mathbb{N}$ , there exists a string  $\rho \succeq \tau$  such that  $\Phi(\rho, x) \downarrow$ . This is where our attempt fails. Yet, intuitively, in this case, we should be able to continue building a sequence while ensuring that the resulting set satisfies  $\neg\mathcal{R}$ . Indeed, whatever the state of progress of the construction, we end up with a string  $\tau \succeq \sigma$ , so by hypothesis, for any  $x \in \mathbb{N}$ , it is always possible to find an extension  $\rho \succeq \tau$  such that  $\Phi(G, x) \downarrow$  for all  $G \in [\rho]$ : for all  $x$ , the set  $Q_x$  of strings  $\rho$  such that  $\Phi(\rho, x) \downarrow$  is dense below  $\sigma$ , that is, for all  $\tau \succeq \sigma$  there exists  $\rho \succeq \tau$  such that  $\rho \in Q_x$ . So if a sufficiently generic set  $G$  extends  $\sigma$ , then it will meet each of  $Q_x$ . We will therefore have  $\forall x \Phi(G, x) \downarrow$ , in other words  $G$  will satisfy  $\neg\mathcal{R}$ .

This analysis therefore motivates the following definition for Cohen forcing:

**Definition 4.3.** A string  $\sigma$  *semantically forces* a requirement  $\mathcal{R}$ , in which case we will write  $\sigma \Vdash \mathcal{R}$ , if any “sufficiently generic” set which extends  $\sigma$  satisfies  $\mathcal{R}$ : there exists a countable sequence of dense sets of strings  $(W_n)_{n \in \mathbb{N}}$  such that if  $G \in [\sigma]$  meets every  $W_n$ , then the requirement will be satisfied for  $G$ .  $\diamond$

Note that the relation  $\Vdash$  is more general than the relation  $\Vdash^*$  for the  $\Sigma_1^0$  and  $\Pi_1^0$  requirements. For example, the empty string  $\epsilon$  semantically forces the requirement  $\mathcal{R}$ : “ $\exists x G(x) = 1$ ”, because the set of strings containing a 1 is dense, and therefore any sufficiently generic set will satisfy  $\mathcal{R}$ . On the other hand, the infinite sequence of zeros  $0^\infty$  belongs to  $[\epsilon]$  and does not satisfy the requirement  $\mathcal{R}$ .

## 4.2. The syntactic approach

Definition 4.3 is simple to express in natural language, but escapes the arithmetic hierarchy: a direct translation requires existentially quantifying on all the countable sequences of dense sets of strings, then universally quantifying on the generic sets for those sets of strings. We are going to define a relation much simpler syntactically speaking, which will make it possible to reason more easily and in particular to prove the essential

properties which one expects from the forcing relation, namely that it is closed under extension and that the set of strings forcing a requirement or its negation is dense.

Let us now define a syntactic forcing relation for any arithmetic requirement, by induction on its quantifications. The first level will be the one we are already used to: let  $\mathcal{R}$  a  $\Sigma_1^0$  requirement — and therefore of the form  $\Phi_e(G) \downarrow$  — then a string  $\sigma$  forces  $\mathcal{R}$  if  $\Phi_e(\sigma) \downarrow$  and therefore if all  $X \in [\sigma]$  satisfies the requirement. The same goes  $\Pi_1^0$  for requirements.

**Definition 4.4.**

- (1)  $\sigma \Vdash^* \mathcal{R}$  for a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement  $\mathcal{R}$  iff all  $X \in [\sigma]$  satisfy  $\mathcal{R}$ .
- (2)  $\sigma \Vdash^* \exists x \mathcal{R}(x)$  for a  $\Pi_k^0$  requirement  $\mathcal{R}(x)$  for  $k > 0$  with free variable  $x$  iff there is an  $n \in \mathbb{N}$  such that  $\sigma \Vdash^* \mathcal{R}(n)$ .
- (3)  $\sigma \Vdash^* \forall x \mathcal{R}(x)$  for a  $\Sigma_k^0$  requirement  $\mathcal{R}(x)$  for  $k > 0$  with free variable  $x$  iff for all  $\tau \succeq \sigma$  and all  $n \in \mathbb{N}$ ,  $\tau \nVdash^* \neg \mathcal{R}(n)$ . ◇

First of all, note that (3) of the previous definition admits an equivalent formulation, sometimes more suited to what we want to demonstrate:

**Lemma 4.5.** Let  $\mathcal{R}$  be a  $\Pi_n^0$  requirement, we have  $\sigma \Vdash^* \mathcal{R}$  iff

$$\forall \tau \succeq \sigma \quad \tau \nVdash^* \neg \mathcal{R}$$

PROOF. This is a simple reformulation of the definition.

Case 1:  $\mathcal{R}$  is a  $\Pi_1^0$  requirement of the form  $\Phi(G) \uparrow$ . Then,  $\sigma \Vdash^* \mathcal{R}$  iff  $\Phi(X) \uparrow$  for all  $X \in [\sigma]$  iff  $\Phi(X) \uparrow$  for all  $\tau \succeq \sigma$  and all  $X \in [\tau]$  iff for all  $\tau \succeq \sigma$ , there exists  $X \in [\tau]$  such that  $\Phi(X) \uparrow$  (by the use property) iff for all  $\tau \succeq \sigma$ ,  $\tau \nVdash^* \Phi(G) \downarrow$ .

Case 2:  $\mathcal{R}$  is a  $\Pi_{k+1}^0$  requirement of the form  $\forall x \mathcal{Q}(x)$  for a  $\Sigma_k^0$  requirement  $\mathcal{Q}(x)$  for  $k \geq 1$ . Then,  $\sigma \Vdash^* \mathcal{R}$  iff for all  $\tau \succeq \sigma$  and all  $n \in \mathbb{N}$ ,  $\tau \nVdash^* \neg \mathcal{Q}(n)$  iff  $\forall \tau \succeq \sigma \quad \tau \nVdash^* \exists x \neg \mathcal{Q}(x)$  iff  $\forall \tau \succeq \sigma \quad \tau \nVdash^* \neg \mathcal{R}$ . ■

The first property that we expect from a forcing relation is its closure under extension. Indeed, “ $\sigma$  forces  $\mathcal{R}$ ” means that the property  $\mathcal{R}$  is already decided on the final constructed object, which should not change during the following stages of the construction.

**Proposition 4.6.** Let  $\sigma, \tau \in 2^{<\mathbb{N}}$  and  $\mathcal{R}$  be an arithmetic requirement. If  $\sigma \Vdash^* \mathcal{R}$  and  $\sigma \preceq \tau$ , then  $\tau \Vdash^* \mathcal{R}$ . ★

PROOF. By induction on the arithmetic complexity of the requirement.

Case 1:  $\mathcal{R}$  is  $\Sigma_1^0$  or  $\Pi_1^0$ . By definition, for all  $X \in [\sigma]$ ,  $\mathcal{R}(X)$  is true. As  $\tau \succeq \sigma$ , then  $[\tau] \subseteq [\sigma]$ , so for all  $X \in [\tau]$ ,  $\mathcal{R}(X)$  is true. So  $\tau \Vdash^* \mathcal{R}$ .

Case 2:  $\mathcal{R}$  is of the form  $\exists x \mathcal{S}(x)$  for a  $\Pi_k^0$  requirement  $\mathcal{S}(x)$  for  $k > 0$ . By definition, there exists an  $n \in \mathbb{N}$  such that  $\sigma \Vdash^* \mathcal{S}(n)$ . By induction hypothesis,  $\tau \Vdash^* \mathcal{S}(n)$ , therefore  $\tau \Vdash^* \exists x \mathcal{S}(x)$ .

Case 3:  $\mathcal{R}$  is of the form  $\forall x \mathcal{S}(x)$  for a  $\Sigma_k^0$  requirement  $\mathcal{S}(x)$  for  $k > 0$ . According to Lemma 4.5,  $\sigma \Vdash^* \mathcal{R}$  implies  $\forall \rho \succeq \sigma \rho \Vdash^* \neg \mathcal{R}$ . If  $\tau \succeq \sigma$  then we also have  $\forall \rho \succeq \tau \rho \Vdash^* \neg \mathcal{R}$ , which again according to Lemma 4.5 implies  $\tau \Vdash^* \mathcal{R}$ . ■

The second property, and perhaps the most important, is the density of the set of strings forcing a requirement or its negation. Proposition 4.7 means in particular that the truth value of any arithmetic requirement will be decided after a finite moment of the construction. This is what gives all the power of the forcing relationship.

**Proposition 4.7.** Let  $\mathcal{R}$  be an arithmetic requirement. The set

$$\{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash^* \mathcal{R} \text{ or } \sigma \Vdash^* \neg \mathcal{R}\}$$

is dense. ★

PROOF. We can assume without loss of generality that  $\mathcal{R}$  is  $\Sigma_n^0$  (in the opposite case we repeat the argument with  $\neg \mathcal{R}$ ). Let  $\sigma$  be a string. According to Lemma 4.5, either  $\tau \Vdash^* \mathcal{R}$  for an extension  $\tau \succeq \sigma$ , or  $\sigma \Vdash^* \neg \mathcal{R}$ . ■

A last property that we also expect is of course the validity of the definition of the forcing relation, that is to say that if a string  $\sigma$  forces a requirement, then this requirement will be effectively satisfied for any sufficiently generic set that extends  $\sigma$ . Let us insist again on what it means to be “sufficiently generic” in this context: there exists a countable sequence of dense sets of strings  $(W_n)_{n \in \mathbb{N}}$  such that if  $G \in [\sigma]$  meets every  $W_n$ , then the requirement will be satisfied for  $G$ .

**Proposition 4.8.** Let  $\mathcal{R}$  be an arithmetic requirement and  $\sigma \in 2^{<\mathbb{N}}$ . If  $\sigma \Vdash^* \mathcal{R}$ , then  $\sigma \Vdash \mathcal{R}$ : if  $G \in [\sigma]$  is sufficiently generic then  $\mathcal{R}$  is satisfied for  $G$ . ★

PROOF. By induction on the arithmetic complexity of the requirement.

Case 1:  $\mathcal{R}$  is  $\Sigma_1^0$  or  $\Pi_1^0$ . Suppose that  $\sigma \Vdash^* \mathcal{R}$ . By definition, any set  $G \in [\sigma]$  satisfies  $\mathcal{R}$ , so *a fortiori* any sufficiently generic set  $G \in [\sigma]$ . Thus,  $\sigma \Vdash \mathcal{R}$ .

Case 2:  $\mathcal{R}$  is of the form  $\exists x \mathcal{S}(x)$  for a  $\Pi_k^0$  requirement  $\mathcal{S}(x)$  for  $k > 0$ . Suppose  $\sigma \Vdash^* \exists x \mathcal{S}(x)$ . By definition, there is an  $n \in \mathbb{N}$  such that  $\sigma \Vdash^* \mathcal{S}(n)$ . By induction hypothesis,  $\sigma$  forces  $\mathcal{S}(n)$ , therefore  $\sigma \Vdash \exists x \mathcal{S}(x)$ .

Case 3:  $\mathcal{R}$  is of the form  $\forall x\mathcal{S}(x)$  for a  $\Sigma_k^0$  requirement  $\mathcal{S}(x)$  for  $k > 0$ . Suppose that  $\sigma \Vdash^* \forall x\mathcal{S}(x)$ . By definition, for all  $\tau \succeq \sigma$  and  $n \in \mathbb{N}$ ,  $\tau \nVdash^* \neg\mathcal{S}(x)$ . By Proposition 4.7, for all  $n$ , the set  $D_n = \{\tau \in 2^{<\mathbb{N}} : \tau \Vdash^* \mathcal{S}(n)\}$  is dense below  $\sigma$ : for all  $\tau \succeq \sigma$  there exists  $\rho \succeq \tau$  such that  $\rho \Vdash^* \mathcal{S}(n)$ . Let  $G$  be a sufficiently generic set extending  $\sigma$ . We suppose in particular that the level of genericity of  $G$  guarantees that it meets every set  $D_n$ . So for all  $n$ , there exists a prefix  $\tau_n \prec G$  such that  $\tau_n \Vdash^* \mathcal{S}(n)$ . For such a string  $\tau_n$ , by induction hypothesis, there exists a countable sequence of dense sets of strings  $(D_{n,m})_{m \in \mathbb{N}}$  such that if  $G \in [\tau_n]$  meets each  $D_{n,m}$  then it satisfies  $\mathcal{S}(n)$ . We have  $G \in [\tau_n]$  and since  $G$  is sufficiently generic it meets every  $D_{n,m}$  and it therefore satisfies  $\mathcal{S}(n)$ . As is the case for all  $n$ , then  $G$  satisfies  $\forall x\mathcal{S}(x)$ , so  $\sigma \Vdash \forall x\mathcal{S}(x)$ . ■

The heart of forcing is undoubtedly in the previous proposition, and in particular in case 3 of its proof: it is there that the mechanism of the relation  $\sigma \Vdash^* \mathcal{R}$  is exerted, which guarantees that if  $G$  belongs to  $[\sigma]$  and if  $G$  is sufficiently generic, then the requirement  $\mathcal{R}$  will be satisfied for  $G$ . This is actually a sophisticated modification of the finite extension method, the key point being the following: regardless of the prefix  $\sigma$  of  $G$  that we have built so far, we can expand  $\sigma$  to meet any dense set of strings fixed in advance.

We now see in the following section that this idea is not new: the mathematician René Baire had already formalized all of these mechanisms at the beginning of the 20th century, in particular via the following theorem: “every Borel class has the Baire property”.

### 4.3. The topological approach: the Baire property

An important technique for the study of complex objects in mathematics consists in reducing oneself to simpler objects while controlling the approximation margin of error. In particular, in the study of sets, whether from the point of view of Measure Theory or Category Theory, there are a number of theorems of the form “any complex set is equivalent to a simple set modulo a negligible quantity of elements”. In Measure Theory for example, we have the three Littlewood principles [144], which state that any measurable set is “almost” a finite union of intervals, any function is “almost” continuous, and any convergent sequence is “almost” uniformly convergent. In this context “almost” means: “except on a set of measure less than  $\varepsilon$  for  $\varepsilon$  as small as we want”.

The Baire property expresses the fact that a class  $S$  is “almost” open, where almost means in this context: “except on a meager class”.

**Definition 4.9.** A class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  has the *Baire property* if there is an open class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$  such that  $\mathcal{B} \Delta \mathcal{U}$  is a meager class. Here  $\mathcal{B} \Delta \mathcal{U}$  is the class of elements on which  $\mathcal{B}$  and  $\mathcal{U}$  do not match, ie  $(\mathcal{B} \setminus \mathcal{U}) \cup (\mathcal{U} \setminus \mathcal{B})$ .  $\diamond$

Consider a requirement  $\mathcal{R}$  and let  $\mathcal{B}_{\mathcal{R}}$  and  $\mathcal{B}_{\neg\mathcal{R}}$  be the classes of the elements which respectively satisfy and do not satisfy this requirement (in particular  $\mathcal{B}_{\mathcal{R}} \cap \mathcal{B}_{\neg\mathcal{R}} = \emptyset$  and  $\mathcal{B}_{\mathcal{R}} \cup \mathcal{B}_{\neg\mathcal{R}} = 2^{\mathbb{N}}$ ). Suppose that  $\mathcal{B}_{\mathcal{R}}$  and  $\mathcal{B}_{\neg\mathcal{R}}$  both have the Baire property and fix two open classes  $\mathcal{U}_{\mathcal{R}}$  and  $\mathcal{U}_{\neg\mathcal{R}}$  such that  $\mathcal{B}_{\mathcal{R}} \Delta \mathcal{U}_{\mathcal{R}}$  and  $\mathcal{B}_{\neg\mathcal{R}} \Delta \mathcal{U}_{\neg\mathcal{R}}$  are both meager. This means that there is a countable union of closed classes of empty interior containing  $\mathcal{B}_{\mathcal{R}} \Delta \mathcal{U}_{\mathcal{R}}$  and  $\mathcal{B}_{\neg\mathcal{R}} \Delta \mathcal{U}_{\neg\mathcal{R}}$ . By passing to the complement, there exists a countable intersection of dense open classes  $\bigcap_n \mathcal{U}_n$  such that for all  $X \in \bigcap_n \mathcal{U}_n$ , we have  $X \in \mathcal{B}_{\mathcal{R}}$  iff  $X \in \mathcal{U}_{\mathcal{R}}$  and  $X \in \mathcal{B}_{\neg\mathcal{R}}$  iff  $X \in \mathcal{U}_{\neg\mathcal{R}}$ .

In particular  $X \in \mathcal{B}_{\mathcal{R}}$  iff there is a prefix  $\sigma \prec X$  such that  $[\sigma] \subseteq \mathcal{U}_{\mathcal{R}}$ . We will then say that  $\sigma$  forces the requirement  $\mathcal{R}$ . Note that if  $X \notin \mathcal{B}_{\mathcal{R}}$  then  $X \in \mathcal{B}_{\neg\mathcal{R}}$  and there then a prefix  $\sigma \prec X$  such that  $[\sigma] \subseteq \mathcal{U}_{\neg\mathcal{R}}$ . At this point  $\sigma$  forces the requirement  $\neg\mathcal{R}$ . The following definition is simply a reformulation of Definition 4.3 which defines semantic forcing.

**Definition 4.10.** Let  $\mathcal{R}$  be a requirement. We say that  $\sigma$  *semantically forces*  $\mathcal{R}$  and we write  $\sigma \Vdash \mathcal{R}$  if  $[\sigma] \setminus \mathcal{B}_{\mathcal{R}}$  is a meager class, where  $\mathcal{B}_{\mathcal{R}}$  is the class of the elements which satisfy  $\mathcal{R}$ .  $\diamond$

As mentioned earlier, the fundamental property expected of the forcing relation for a requirement is as follows.

(D): The set of strings forcing  $\mathcal{R}$  or forcing  $\neg\mathcal{R}$  is dense.

We therefore start with a characterization of the requirements for which this property is true, using the Baire property.

**Proposition 4.11.** Let  $\mathcal{R}$  be a requirement and  $\mathcal{B}_{\mathcal{R}}$  the class of elements which satisfy  $\mathcal{R}$ . The following statements are equivalent:

- (1)  $\mathcal{B}_{\mathcal{R}}$  has the Baire property
- (2)  $\{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \mathcal{R} \text{ or } \sigma \Vdash \neg\mathcal{R}\}$  is dense ★

PROOF. Let  $\mathcal{U}_{\mathcal{R}} = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \mathcal{R}\}$  and  $\mathcal{U}_{\neg\mathcal{R}} = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \neg\mathcal{R}\}$ .

(1)  $\Rightarrow$  (2). Suppose that  $\mathcal{B}_{\mathcal{R}}$  has the Baire property. Let  $\mathcal{U}$  be an open class such that  $\mathcal{B}_{\mathcal{R}} \Delta \mathcal{U}$  is meager. In particular,  $\mathcal{U} \setminus \mathcal{B}_{\mathcal{R}}$  is meager, so  $\mathcal{U} \subseteq [\mathcal{U}_{\mathcal{R}}]$ . Also,  $\mathcal{B}_{\mathcal{R}} \setminus \mathcal{U} = (2^{\mathbb{N}} \setminus \mathcal{U}) \setminus \mathcal{B}_{\neg\mathcal{R}}$  is meager, so  $\text{int}(2^{\mathbb{N}} \setminus \mathcal{U}) \setminus \mathcal{B}_{\neg\mathcal{R}}$  is meager. It follows that  $\text{int}(2^{\mathbb{N}} \setminus \mathcal{U}) \subseteq [\mathcal{U}_{\neg\mathcal{R}}]$ . Since  $\mathcal{U} \cup \text{int}(2^{\mathbb{N}} \setminus \mathcal{U})$  is dense in  $2^{\mathbb{N}}$ ,  $[\mathcal{U}_{\mathcal{R}}] \cup [\mathcal{U}_{\neg\mathcal{R}}]$  is also dense, so by Exercize 2.8,  $\mathcal{U}_{\mathcal{R}} \cup \mathcal{U}_{\neg\mathcal{R}}$  is dense in  $2^{<\mathbb{N}}$ .

(2)  $\Rightarrow$  (1). Suppose that  $U_{\mathcal{R}} \cup U_{\neg\mathcal{R}}$  is dense in  $2^{<\mathbb{N}}$ . By density, the complement of  $[U_{\mathcal{R}}] \cup [U_{\neg\mathcal{R}}]$  is a closed class of empty interior, therefore meager. Let us show that  $\mathcal{B}_R \Delta [U_{\mathcal{R}}]$  is meager. By definition, for all  $\sigma \in U_{\mathcal{R}}$ ,  $[\sigma] \setminus \mathcal{B}_R$  is meager, so  $[U_{\mathcal{R}}] \setminus \mathcal{B}_R$  is a countable union of meager classes, so is meager. By the same reasoning, if  $\mathcal{B}_{\neg\mathcal{R}}$  is the class of elements which satisfy  $\neg\mathcal{R}$ ,  $[U_{\neg\mathcal{R}}] \setminus \mathcal{B}_{\neg\mathcal{R}}$  is meager. As  $[U_{\neg\mathcal{R}}] \setminus \mathcal{B}_{\neg\mathcal{R}} = \mathcal{B}_R \cap [U_{\neg\mathcal{R}}]$ , and since the complement of  $[U_{\mathcal{R}}] \cup [U_{\neg\mathcal{R}}]$  is meager, then  $\mathcal{B}_R \setminus [U_{\mathcal{R}}]$  is meager. So  $\mathcal{B}_R \Delta [U_{\mathcal{R}}]$  is meager. ■

Since the syntactic forcing relation implies the semantic relation, we can deduce the density of the semantic forcing relation for arithmetic requirements:

**Lemma 4.12.** Let  $\mathcal{R}$  be an arithmetic requirement. The set

$$\{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash \mathcal{R} \vee \sigma \Vdash \neg\mathcal{R}\}$$

is dense. ★

PROOF. For all  $\sigma$  there exists an extension  $\tau \succeq \sigma$  such that  $\tau \Vdash^* \mathcal{R}$  or such that  $\tau \Vdash^* \neg\mathcal{R}$ . If  $\tau \Vdash^* \mathcal{R}$  then  $\tau \Vdash \mathcal{R}$  and if  $\tau \Vdash^* \neg\mathcal{R}$  then  $\tau \Vdash \neg\mathcal{R}$ . ■

The following corollary follows directly from the characterization of the classes having the Baire property.

**Corollary 4.13**

*Let  $\mathcal{R}$  is an arithmetic requirement. The class  $\mathcal{B}_{\mathcal{R}}$  of elements which satisfy  $\mathcal{R}$ , has the Baire property.*

PROOF. Immediate by Lemma 4.12 and Proposition 4.11. ■

We will develop in Chapter 17 and Section 27-3.1 the theory of Borel classes, which formalizes and generalizes the notion of arithmetic requirements, and which all have the Baire property.

Let us end this section by clarifying the links between the semantic forcing relation and the syntactic relation  $\Vdash^*$  that we defined in the previous section:

**Theorem 4.14**

*Let  $\mathcal{R}$  be an arithmetic requirement. Then  $\sigma \Vdash \mathcal{R}$  iff  $\{\tau \in 2^{<\mathbb{N}} : \tau \Vdash^* \mathcal{R}\}$  is dense below  $\sigma$ .*

To show the theorem we will use two lemmas. The first corresponds to Proposition 4.6 for the relation  $\Vdash^*$ .

**Lemma 4.15.** Suppose  $\sigma \Vdash \mathcal{R}$  for a string  $\sigma$  and a requirement  $\mathcal{R}$ . If  $\tau \succeq \sigma$  then  $\tau \Vdash \mathcal{R}$ . ★

PROOF. Let  $\mathcal{B}_{\mathcal{R}}$  be the class of elements which satisfy  $\mathcal{R}$ . We have  $[\tau] \setminus \mathcal{B}_{\mathcal{R}} \subseteq [\sigma] \setminus \mathcal{B}_{\mathcal{R}}$ . So if  $[\sigma] \setminus \mathcal{B}_{\mathcal{R}}$  is meager then  $[\tau] \setminus \mathcal{B}_{\mathcal{R}}$  is meager. ■

**Lemma 4.16.** Suppose  $\sigma \Vdash \mathcal{R}$  for a string  $\sigma$  and a requirement  $\mathcal{R}$ . Then,  $\sigma \nVdash \neg \mathcal{R}$  ★

PROOF. Let  $\mathcal{B}_{\mathcal{R}}$  be the class of elements which satisfy  $\mathcal{R}$  and Let  $\mathcal{B}_{\neg \mathcal{R}}$  be the class of elements which satisfy  $\neg \mathcal{R}$ .

Let us assume by the absurdity  $\sigma \Vdash \mathcal{R}$  and  $\sigma \Vdash \neg \mathcal{R}$ . Then,  $[\sigma] \setminus (\mathcal{B}_{\mathcal{R}} \cap [\sigma])$  and  $[\sigma] \setminus (\mathcal{B}_{\neg \mathcal{R}} \cap [\sigma])$  are both meager. Moreover  $([\sigma] \setminus (\mathcal{B}_{\mathcal{R}} \cap [\sigma])) \cup ([\sigma] \setminus (\mathcal{B}_{\neg \mathcal{R}} \cap [\sigma])) = [\sigma]$ , which contradicts the fact that the union of two meager classes is meager and therefore of empty interior. ■

PROOF OF THEOREM 4.14. Recall Proposition 4.8: for any arithmetic requirement  $\mathcal{R}$  and for all  $\sigma$ , if  $\sigma \Vdash^* \mathcal{R}$  then  $\sigma \Vdash \mathcal{R}$ .

Let us show that if  $\{\tau \in 2^{<\mathbb{N}} : \tau \Vdash^* \mathcal{R}\}$  is dense below  $\sigma$ , then  $\sigma \Vdash \mathcal{R}$ . According to Proposition 4.8,  $A = \{\tau \in 2^{<\mathbb{N}} : \tau \Vdash \mathcal{R}\}$  is dense below  $\sigma$ . So the class  $\{X \in [\sigma] : \forall n \ X \restriction_n \nVdash \mathcal{R}\}$  is therefore a closed class of empty interior, which we denote by  $\mathcal{F}$ . Let  $\mathcal{B}_{\mathcal{R}}$  be the class of elements which satisfy  $\mathcal{R}$ . By definition of  $A$ , for all  $\tau \in A$ , the class  $\mathcal{M}_{\tau} = [\tau] \setminus (\mathcal{B}_{\mathcal{R}} \cap [\tau])$  is meager. It follows that the class  $[\sigma] \setminus (\mathcal{B}_{\mathcal{R}} \cap [\sigma])$  is included in the union of  $\mathcal{F}$  and all the classes  $\mathcal{M}_{\tau}$ . It is therefore meager.

Finally, let us show that if  $\sigma \Vdash \mathcal{R}$  then  $\{\tau \in 2^{<\mathbb{N}} : \tau \Vdash^* \mathcal{R}\}$  is dense below  $\sigma$ . By contraposition, let us suppose that there exists  $\tau \succeq \sigma$  such that for all  $\rho \succeq \tau$  we have  $\rho \nVdash^* \mathcal{R}$ . By Proposition 4.7, there must then exist  $\rho \succeq \tau$  such that  $\rho \Vdash^* \neg \mathcal{R}$ . By Proposition 4.8, we then have  $\rho \Vdash \neg \mathcal{R}$ . According to Lemma 4.16,  $\rho \nVdash \mathcal{R}$  and therefore according to Lemma 4.15,  $\sigma \nVdash \mathcal{R}$ . ■

## 5. Arbitrarily generic sets

Genericity can be seen as a notion of “typicality”, in the sense that anything that can happen infinitely often will end up happening. In the case of Cohen forcing, a dense set has infinitely often the possibility of being met, and if a set is typical this will happen, which corresponds to the notion of generic set.

We have seen relativizations of weakly 1-generic and 1-generic sets, and in particular the concepts of  $n$ -generic and weakly  $n$ -generic set. In this

section, we study the properties of typical sets, that is, the properties that sufficiently generic sets will have.

Note that if a property is verified by any sufficiently generic set, it is without loss of generality verified for any weakly 1-generic set relative to  $A$  for a certain oracle  $A$  sufficiently powerful: it suffices that  $A$  encodes the intersection of dense open classes corresponding to the required level of genericity. We will therefore endeavor to determine “the right level” of genericity necessary to satisfy such and such a property. In practice, this will often correspond to being  $n$ -generic for some  $n$ .

### 5.1. Properties of sufficiently generic sets

We have seen that 1-genericity was the level of genericity corresponding to the forcing of  $\Sigma_1^0/\Pi_1^0$  requirements. Unsurprisingly, we now see that the  $n$ -genericity is the level of genericity corresponding to the forcing of  $\Sigma_n^0/\Pi_n^0$  requirements, and we will show the following theorem.

#### Theorem 5.1

*Let  $G$  be an  $n$ -generic set. Let  $\mathcal{R}$  be a  $\Sigma_n^0$  or  $\Pi_n^0$  requirement. Then,  $G$  satisfies  $\mathcal{R}$  iff there is a prefix  $\sigma \prec G$  such that  $\sigma \Vdash^* \mathcal{R}$ .*

Let us note that this iteration is not trivial: it is necessary to appeal for that to the developments of the forcing relation of the preceding sections.

We will now proceed to the proof of Theorem 5.1 by exploiting the definitional simplicity of the syntactic forcing relation. Let's start with the following proposition.

**Proposition 5.2.** Let  $\mathcal{R}$  be an arithmetic requirement.

- (1) If  $\mathcal{R}$  is  $\Sigma_n^0$ , then the predicate  $\sigma \Vdash^* \mathcal{R}$  is  $\Sigma_n^0$
- (2) If  $\mathcal{R}$  is  $\Pi_n^0$ , then the predicate  $\sigma \Vdash^* \mathcal{R}$  is  $\Pi_n^0$  ★

PROOF. We will prove (1) and (2) simultaneously by induction on the arithmetic complexity of the requirement. The  $\Sigma_1^0$  and  $\Pi_1^0$  case has already been treated with Proposition 3.17.

If  $\mathcal{R}$  is  $\Sigma_{m+1}^0$  with  $m > 0$ , then it can be expressed in the form  $\exists x \mathcal{S}(x)$  where  $\mathcal{S}$  is a  $\Pi_m^0$  requirement. We have  $\sigma \Vdash^* \mathcal{R}$  iff there exists an  $n \in \mathbb{N}$  such  $\sigma \Vdash^* \mathcal{S}(n)$ . By induction hypothesis, the predicate  $\sigma \Vdash^* \mathcal{S}(n)$  is  $\Pi_m^0$ , so the predicate  $\sigma \Vdash^* \mathcal{R}$  is  $\Sigma_{m+1}^0$ .

If  $\mathcal{R}$  is  $\Pi_{m+1}^0$  with  $m > 0$ , then it can be expressed in the form  $\forall x \mathcal{S}(x)$  where  $\mathcal{S}$  is a  $\Sigma_m^0$  requirement. We have  $\sigma \Vdash^* \mathcal{R}$  iff for all  $\tau \succeq \sigma$  and all  $n \in \mathbb{N}$ ,  $\tau \nVdash^* \neg \mathcal{S}(n)$ . In particular,  $\neg \mathcal{S}(n)$  is  $\Pi_m^0$ , so by induction

hypothesis, the predicate  $\tau \Vdash^* \neg \mathcal{S}(n)$  is  $\Pi_m^0$ , therefore  $\tau \nVdash^* \neg \mathcal{S}(n)$  is  $\Sigma_m^0$ , and the predicate  $\sigma \Vdash^* \mathcal{R}$  is  $\Pi_{m+1}^0$ . ■

We now see the level of genericity required to make Proposition 4.8 true.

**Proposition 5.3.** Let  $\mathcal{R}$  be an arithmetic requirement such that  $\sigma \Vdash^* \mathcal{R}$  for  $\sigma \in 2^{<\mathbb{N}}$ .

- (1) If  $\mathcal{R}$  is  $\Sigma_n^0$ , then  $\mathcal{R}$  is satisfied for all weakly  $(n-2)$ -generic sets which extend  $\sigma$ .
- (2) If  $\mathcal{R}$  is  $\Pi_n^0$ , then  $\mathcal{R}$  is satisfied for all weakly  $(n-1)$ -generic sets which extend  $\sigma$ . ★

PROOF. By definition of the forcing relation, if  $\mathcal{R}$  is  $\Sigma_1^0$  or  $\Pi_1^0$  then it is satisfied for any set which extends  $\sigma$ . Suppose the proposition is true for  $n$ . Let  $\mathcal{R} = \exists x \mathcal{Q}(x)$  be a  $\Sigma_{n+1}^0$  requirement with  $\mathcal{Q}(x)$  a  $\Pi_n^0$  requirement. We have  $\sigma \Vdash^* \mathcal{R}$  iff there exists  $m \in \mathbb{N}$  such that  $\sigma \Vdash^* \mathcal{Q}(m)$ . By induction hypothesis  $\mathcal{Q}(m)$  is satisfied for any weakly  $(n-1)$ -generic set which extends  $\sigma$  and therefore it is the same for  $\mathcal{R}$ .

Suppose now that  $\mathcal{R} = \forall x \mathcal{Q}(x)$  is a  $\Pi_{n+1}^0$  requirement with  $\mathcal{Q}(x)$  a  $\Sigma_n^0$  requirement. We have  $\sigma \Vdash^* \mathcal{R}$  iff for all  $m \in \mathbb{N}$  and for all  $\tau \succeq \sigma$   $\tau \nVdash^* \neg \mathcal{Q}(m)$ . According to Proposition 4.7 this means that for  $m$  fixed, the set  $A_m = \{\tau : \tau \Vdash^* \mathcal{Q}(m)\}$  is dense below  $\sigma$ . According to Proposition 5.2 each set  $A_m$  is  $\Sigma_n^0$ . In particular if  $G$  is weakly  $n$ -generic and extends  $\sigma$  then  $G$  meets  $A_m$ . By induction hypothesis if  $G$  meets  $A_m$  and is weakly  $n$ -generic then  $\mathcal{Q}(m)$  is true for  $G$ . As is the case for all  $m$  then  $\mathcal{R}$  is true for weakly  $n$ -generic set  $G$  which extends  $\sigma$ . ■

We can finally show Theorem 5.1.

PROOF OF THEOREM 5.1. The  $\Sigma_1^0/\Pi_1^0$  case has already been treated with Theorem 3.12. Let  $n > 1$ . We can assume without loss of generality that  $\mathcal{R}$  is  $\Sigma_n^0$ . In the opposite case, we reproduce the following argument with  $\neg \mathcal{R}$  instead of  $\mathcal{R}$ .

Let  $U = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash^* \mathcal{R}\}$ . According to Lemma 4.5,  $U^\perp = \{\sigma \in 2^{<\mathbb{N}} : \sigma \Vdash^* \neg \mathcal{R}\}$ . According to Proposition 5.2, the set  $U$  is  $\Sigma_n^0$  and the set  $U^\perp$  is  $\Pi_n^0$ . Note that as  $G$  is  $n$ -generic, it meets  $U \cup U^\perp$ .

Suppose that  $\mathcal{R}$  is satisfied for  $G$ . Then,  $G$  cannot meet  $U^\perp$  because in this case we would have a prefix  $\sigma \prec G$  such that  $\sigma \Vdash^* \neg \mathcal{R}$  and by Proposition 5.3,  $\neg \mathcal{R}$  would therefore be satisfied on  $G$ , which contradicts the fact that  $\mathcal{R}$  is satisfied on  $G$ . So  $G$  meets  $U$  and we have a prefix  $\sigma \prec G$  such that  $\sigma \Vdash^* \mathcal{R}$ .

Suppose that  $\neg\mathcal{R}$  is satisfied for  $G$ . Symmetrically,  $G$  meets necessarily  $U^\perp$  and we have a prefix  $\sigma \prec G$  such that  $\sigma \Vdash^* \mathcal{R}$ .

Conversely if a prefix  $\sigma \prec G$  is such that  $\sigma \Vdash^* \mathcal{R}$  or  $\sigma \Vdash^* \neg\mathcal{R}$  then respectively  $\mathcal{R}$  or  $\neg\mathcal{R}$  is satisfied on  $G$  according to Proposition 5.3. ■

## 5.2. Turing degree of sufficiently generic sets

Let us first recap a result that we have already established: given a set  $A$ , if  $G$  is sufficiently generic then  $A$  does not compute  $G$  and  $G$  does not compute  $A$ . **TODO**

### Theorem 5.4

*Let  $A \in 2^\mathbb{N}$  be a non-computable set. If  $G$  is 1-generic relative to  $A$ , then  $A$  and  $G$  are of incomparable Turing degrees.*

PROOF. According to Exercize 3.5 and Exercize 3.6, if  $G$  is weakly 1-generic relative to  $A$ , it is of degree  $A$ -hyperimmune and therefore cannot be computed by  $A$ . According to Theorem 3.28, if  $G$  is 1-generic relative to  $A$ , it does not compute  $A$ . ■

Note that Theorem 5.4 can be generalized to show that for any countable sequence of sets fixed in advance  $A_0, A_1, \dots$ , any set  $G$  sufficiently generic for Cohen forcing is incomparable with the items from this list. Indeed if  $G$  is sufficiently generic, it will be 1-generic relative to  $A_i$  for all  $i$ . This result, despite its simplicity, admits of several interesting consequences:

### Corollary 5.5

*Let  $G$  be a sufficiently generic set for Cohen forcing. Then,  $G$  is not arithmetic and does not compute any non-computable arithmetic set.*

Let's now see how to reinforce and iterate Theorem 5.4: for  $A$  not  $\emptyset^{(n)}$ -computable, if a set  $G$  is sufficiently generic, not only will it not compute  $A$ , but also the  $n$ -th Turing jump will not compute  $A$ . We will actually see something a little more precise: if  $A$  is not  $\Sigma_n^0$  and if  $G$  is sufficiently generic, then  $A$  will not be  $\Sigma_n^0(G)$ . Let us first make sure of the complexity of the requirement  $a \in G^{(n)}$  via the following lemma.

**Lemma 5.6.** For all  $a \in \mathbb{N}$ , the requirement  $a \in X^{(n)}$  is  $\Sigma_n^0$  uniformly in  $a$ . ★

PROOF. For the case  $n = 1$  we have  $a \in X'$  iff  $\Phi_a(X, a) \downarrow$ , which is indeed a  $\Sigma_1^0$  requirement. In the case  $n$ , suppose that  $F_n(G, x)$  is a  $\Sigma_n^0$  formula of

second-order arithmetic such that for all  $X \in 2^{\mathbb{N}}$  and for all  $a \in \mathbb{N}$ ,  $F(X, a)$  is true iff  $a \in X^{(n)}$ . We then have:

$$\begin{aligned} a \in X^{(n+1)} &\leftrightarrow \exists \sigma \forall i < |\sigma| \left( \begin{array}{l} (\sigma(i) = 0 \wedge i \notin X^{(n)}) \\ \vee (\sigma(i) = 1 \wedge i \in X^{(n)}) \end{array} \right) \wedge \Phi_a(\sigma, a) \downarrow \\ &\leftrightarrow \exists \sigma \forall i < |\sigma| \left( \begin{array}{l} (\sigma(i) = 0 \wedge \neg F_n(X, i)) \\ \vee (\sigma(i) = 1 \wedge F_n(X, i)) \end{array} \right) \wedge \Phi_a(\sigma, a) \downarrow. \end{aligned}$$

By using the fact that  $\neg F_n(G, x)$  and  $F_n(G, x)$  are both  $\Sigma_{n+1}^0$  formulas, and using the closure of the  $\Sigma_{n+1}^0$  formulas under bounded quantification, finite union and finite intersection, we can define a  $\Sigma_{n+1}^0$  formula  $H(G, \tau)$  such that

$$H(X, \sigma) \leftrightarrow \forall i < |\sigma| ((\sigma(i) = 0 \wedge \neg F_n(X, i)) \vee (\sigma(i) = 1 \wedge F_n(X, i))).$$

Then,

$$a \in X^{(n+1)} \leftrightarrow \exists \sigma H(X, \sigma) \wedge \Phi_a(\sigma, a) \downarrow,$$

which is indeed a  $\Sigma_{n+1}^0$  formula. ■

### Theorem 5.7

Let  $A$  be a non  $\Sigma_{n+1}^0$  set for  $n \geq 0$ . Then, for any 1-generic set  $G$  relative to  $A \oplus \emptyset^{(n)}$ ,  $A$  is not  $\Sigma_{n+1}^0(G)$ .

PROOF. Let  $G$  be a 1-generic set relative to  $A \oplus \emptyset^{(n+1)}$ . We must show that for any  $e$ , the set  $A$  is different from  $W_e^{G^{(n)}}$ , the set  $G^{(n)}$ -c.e. code  $e$ . Let

$$U = \{\sigma : \exists m \notin A \ \sigma \Vdash^* m \in W_e^{G^{(n)}}\}.$$

According to Lemma 5.6, the requirement  $m \in W_e^{G^{(n)}}$  is  $\Sigma_{n+1}^0$ . So according to Proposition 5.2, the set  $U$  is  $\Sigma_1^0(A \oplus \emptyset^{(n)})$ . If  $G$  meets  $U$  then according to Proposition 5.3, we will have  $m \in W_e^{G^{(n)}}$  for  $m \notin A$  and therefore  $A \neq W_e^{G^{(n)}}$ .

If  $G$  does not meet  $U$ , since  $G$  is 1-generic relative to  $A \oplus \emptyset^{(n+1)}$ ,  $G$  meets  $U^\perp = \{\sigma : \forall m \notin A \ \forall \tau \succeq \sigma \ \tau \nVdash^* m \in W_e^{G^{(n)}}\}$ . According to Lemma 4.5, we have  $U^\perp = \{\sigma : \forall m \notin A \ \sigma \Vdash^* m \notin W_e^{G^{(n)}}\}$ . Let  $\sigma \in U^\perp$  be a fixed string. Then, the set  $D_\sigma = \{m : \sigma \Vdash^* m \notin W_e^{G^{(n)}}\}$  is a  $\Pi_{n+1}^0$  set which by hypothesis contains  $\mathbb{N} \setminus A$ . As  $\mathbb{N} \setminus A$  is not  $\Pi_{n+1}^0$ , there is necessarily an element  $m \in A$  such that  $m \in D_\sigma$ . So if  $G$  meets  $U^\perp$  it also meets the set  $\{\sigma : \exists m \in A \ \sigma \Vdash^* m \notin W_e^{G^{(n)}}\}$ . Then, according to Proposition 5.3, we will have  $m \notin W_e^{G^{(n)}}$  for  $m \in A$  and therefore  $A \neq W_e^{G^{(n)}}$ . ■

**Corollary 5.8**

Let  $A$  be a non- $\emptyset^{(n)}$ -computable set for  $n \geq 0$ . Then, if  $G$  is 1-generic relative to  $A \oplus \emptyset^{(n)}$ , the set  $A$  is not  $G^{(n)}$ -computable.

PROOF. As  $A$  is not  $\emptyset^{(n)}$ -computable it is not  $\Delta_{n+1}^0$ . Since  $A$  is not  $\Delta_{n+1}^0$ , either  $A$  is not  $\Sigma_{n+1}^0$ , or  $\bar{A}$  is not  $\Sigma_{n+1}^0$ . By Theorem 5.7, for any 1-generic set  $G$  relative to  $A \oplus \emptyset^{(n)}$ ,  $A$  is not  $\Sigma_{n+1}^0(G)$  in the first case, and  $\bar{A}$  is not  $\Sigma_{n+1}^0(G)$  in the second case. In any case,  $A$  is not  $\Delta_{n+1}^0(G)$  and therefore it is not  $G^{(n)}$ -computable. ■

We saw with Theorem 3.20 that the Turing jump of 1-generic sets was a continuous function on the class of 1-generic sets: the 1-generics are all generalized low. This result can be put into perspective: the  $n$ -th Turing jump is a continuous function on the class of  $n$ -generic sets:

**Theorem 5.9**

Let  $G$  be an  $n$ -generic set. Then,  $G^{(n)} \leq_T G \oplus \emptyset^{(n)}$ .

PROOF. Let  $n > 0$  and  $G$  be an  $n$ -generic set. Let  $U_e = \{\sigma : \sigma \Vdash^* e \in G^{(n)}\}$ . According to Lemma 5.6, the requirement  $e \in G^{(n)}$  is  $\Sigma_n^0$  and therefore  $U_e$  is a  $\Sigma_n^0$  set. Consider  $U_e^\perp = \{\sigma : \forall \tau \succeq \sigma \ \tau \nVdash^* e \in G^{(n)}\}$ . In particular,  $U_e^\perp$  is a  $\Pi_n^0$  set. According to Lemma 4.5, we have  $U_e^\perp = \{\sigma : \sigma \Vdash^* e \notin G^{(n)}\}$ . As  $G$  is  $n$ -generic, it meets  $U_e \cup U_e^\perp$ . It suffices using  $\emptyset^{(n)}$  to search for a prefix of  $G$  in  $U_e$  or  $U_e^\perp$ . According to Proposition 5.3, in the first case we have  $e \in G^{(n)}$  and in the second  $e \notin G^{(n)}$ . ■

Note that Corollary 5.8 is also deduced from the previous theorem and from a relativized version of Theorem 3.28. Beware, in the literature, the notion of generalized low <sub>$n$</sub>  does not correspond to the property of theorem 5.9.

**Definition 5.10.** A set  $G \in 2^{\mathbb{N}}$  is *generalized low <sub>$n$</sub>*  if  $G^{(n)} \leq_T (G \oplus \emptyset')^{(n-1)}$ . ◆

Note that if  $X$  is generalized low <sub>$n$</sub>  it is also generalized low <sub>$n+1$</sub> . Each  $n$ -generic set is indeed generalized low <sub>$n$</sub> , but Theorem 5.9 proves something stronger.



# Chapter 11

## Effective forcing

On the strength of the intuitions created with the study of Cohen forcing, we can then introduce the abstract forcing notions on an arbitrary partial order, and develop all the associated machinery.

### 1. Fundamentals of forcing

Now that we have familiarized ourselves with the notions of density and genericity on the partial order of binary strings, equipped with the extension relation, we are ready to approach the concepts of forcing in all their generality, while keeping the intuitions of the finite extension method. The benefits of this abstraction will only appear from the introduction of the forcing relation, which gives all its power to the formalism. So far, we have seen the notion of genericity in the partial order of binary strings as a systematization of constructions with the finite extension method.

**Partial order.** In all its generality, a forcing notion is quite simply a partial order  $(\mathbb{P}, \leq)$ , whose elements are called *conditions*. A condition intuitively represents an approximation of the object that we are constructing. A *extension* of  $c \in \mathbb{P}$  is a condition  $d \leq c$ .

#### Remark

Beware, for historical reasons, the order relation of the forcing is reversed. An extension of a condition  $c$  is therefore smaller condition  $d$ . The underlying idea comes from the fact that  $d$  is a more precise approximation than  $c$ , and that therefore the set of “candidate” objects

that we are building is a subset of the candidates of  $c$ , because the more constraints we add, the more candidates we exclude.

**Filter.** In the case of the finite extension method, the constructed object is an element of  $2^{\mathbb{N}}$ , using a strictly increasing infinite sequence of strings. In the language of an arbitrary partial order, we are going to construct an infinite decreasing sequence of conditions. The produced object is a maximal filter.

**Definition 1.1.** Two conditions  $c_0, c_1$  are *compatible* if there is a condition  $d$  which extends both  $c_0$  and  $c_1$ . Otherwise,  $c_0$  and  $c_1$  are *incompatible*. A *filter* is an upward-closed set  $F \subseteq \mathbb{P}$ , such that for any  $c_0, c_1 \in F$ , there exists a condition  $d \in F$  such that  $d \leq c_0, c_1$ . A filter is *maximal* if it is not included in a strictly larger filter. ◇

If we consider the partial order on the strings, equipped with the suffix relation, the maximal filters are in one-to-one correspondence with Cantor space. Indeed, for any  $X \in 2^{\mathbb{N}}$ , the set  $\{X \upharpoonright_n : n \in \mathbb{N}\}$  is a maximal filter, and conversely, any maximal filter is of this form.

In the context of a countable partial order, it may be more intuitive to think of a filter as the upward closure of an infinite decreasing sequence of conditions. In particular, for any decreasing infinite sequence of conditions  $c_0 \geq c_1 \geq c_2 \geq \dots$ , the set

$$F = \{d \in \mathbb{P} : \exists n \ c_n \leq d\}$$

is a filter. This intuition corresponds more to the construction of the finite extension method.

### — Notation —

As explained, a condition  $c \in \mathbb{P}$  can be seen as an approximation of the object being constructed, namely a maximal filter. We can therefore associate with each condition the set  $[c]_{\in}$  of maximum filters containing  $c$ , representing the candidate objects. In particular, according to intuition, if  $d \leq c$ , then the approximation  $d$  is more precise than the approximation  $c$ , thus reducing the number of candidates. We therefore have  $[d]_{\in} \subseteq [c]_{\in}$ .

Note that for any maximal filter  $F$ ,  $\bigcap_{c \in F} [c]_{\in} = \{F\}$ . In other words,  $F$  is the unique candidate of all the conditions of the filter simultaneously. We have already encountered several forcing notions in the previous chapters. Here are a few. We will see that for each of these forcing notions, the maximum filters can be interpreted as sets of integers.

**Example 1.2.**

1. *Cohen forcing* is the partial order of strings equipped with its suffix relation  $(2^{<\mathbb{N}}, \succeq)$ . For any  $\sigma \in 2^{<\mathbb{N}}$ ,  $[\sigma]_\infty$  is in bijection with the set  $[\sigma] = \{X \in 2^\mathbb{N} : \sigma \prec X\}$ , by the function which to  $F \in [\sigma]_\infty$  associates the unique element of  $\bigcap_{\sigma \in F} [\sigma]$ .
2. *Jockusch-Soare forcing* is the partial order of non-empty  $\Pi_1^0$  classes, ordered by the inclusion relation. For any  $\Pi_1^0$  class  $\mathcal{P}$ , the set  $[\mathcal{P}]_\infty$  is in bijection with  $\mathcal{P}$ . A maximal filter  $F$  containing  $\mathcal{P}$  can therefore be seen as the only element of  $\bigcap_{Q \in F} Q$ , which is a member of  $\mathcal{P}$ .
3. *Sacks forcing* is the partial order of computable  $f$ -trees, ordered by the sub- $f$ -tree relation. For any  $f$ -computable tree  $T$ , the set  $[T]_\infty$  is in bijection with the set

$$[T] = \left\{ \bigcup_n T(X \upharpoonright_n) : X \in 2^\mathbb{N} \right\} = \{Y \in 2^\mathbb{N} : \exists^\infty n \ Y \upharpoonright_n \in \text{Im } T\}$$

In each of the preceding examples, for any maximal filter  $F$ , we will denote by  $\dot{F}$  the corresponding element of  $2^\mathbb{N}$ . We therefore have for each of these forcing notions the equality  $[c] = \{\dot{F} : F \in [c]_\infty\}$ .

**Density, genericity.** Recall that in the finite extension method, requirements can be represented as the set of strings that force them. This representation has the advantage of being abstracted from the notion of requirement and is generalized to any partial order.

**Definition 1.3.** Let  $(\mathbb{P}, \leq)$  be a partial order. A set  $D \subseteq \mathbb{P}$  is *dense* in  $(\mathbb{P}, \leq)$  if for all  $c \in \mathbb{P}$ , there exists a  $d \leq c$  such that  $d \in D$ .  $\diamond$

The intuition that it is useful to keep with the notion of density is that if a set  $D$  is dense, then whatever the finite piece of the decreasing sequence of conditions that we have already constructed, it will never be too late to integrate an element of  $D$  in the sequence.

**Definition 1.4.** Let  $\vec{D} = (D_n)_{n \in \mathbb{N}}$  be a collection of sets of conditions. A filter  $F \subseteq \mathbb{P}$  is  $\vec{D}$ -*generic* if it intersects  $D_n$  for all  $n \in \mathbb{N}$ .  $\diamond$

The following proposition shows that if the sets of conditions are dense, there is a generic filter for these sets. The proof of this proposition corresponds to the construction, by the finite extension method, of an increasing infinite sequences of strings satisfying each requirement.

**Proposition 1.5.** Let  $\vec{D} = (D_n)_{n \in \mathbb{N}}$  be a countable collection of dense sets, and  $c \in \mathbb{P}$  a condition. There is a  $\vec{D}$ -generic filter containing  $c$ . ★

PROOF. Let us define an infinite decreasing sequence of conditions  $c_0 \geq c_1 \geq c_2 \geq \dots$  inductively as follows:  $c_0 = c$ . If  $c_n$  is defined,  $c_{n+1} \leq c_n$  is a condition belonging to  $D_n$ . Such an extension of  $c_n$  exists by density of the set  $D_n$ . Let  $F = \{d \in \mathbb{P} : \exists n \, d \geq c_n\}$ . The set  $F$  is a filter containing  $c$  and meeting  $D_n$  for all  $n$ . ■

As explained in Section 10-2, there is in general an uncountable quantity of dense sets, and a filter  $F$  cannot be generic for all these sets simultaneously. The notion of genericity is therefore dependent on a countable collection  $\vec{D}$  of dense sets. We will say that any *sufficiently generic* filter for a forcing notion satisfies such property if there exists a countable collection of dense sets  $\vec{D}$  such that any filter  $\vec{D}$ -generic satisfies this property.

**Definition 1.6.** Let  $(\mathbb{P}, \leq)$  be a partial order,  $c \in \mathbb{P}$  and let  $D \subseteq \mathbb{P}$  be a set. We say that  $D$  is *dense below*  $c$  if for all  $d \leq c$ , there exists an  $e \leq d$  such that  $e \in D$ . ◇

The following exercise will be useful in the rest of this development.

**Exercise 1.7.** Suppose that a set  $D \subseteq \mathbb{P}$  is dense below  $c \in \mathbb{P}$ . Show that any sufficiently generic filter  $F$  containing  $c$  intersects  $D$ . ◇

## 2. Forcing relation

So far, we have developed notions expressed purely in terms of partial order, namely, the notions of density, filter and genericity. We will now define a generalization of the forcing relation introduced in Section 10-4. For that, we will restrict ourselves to forcing notions producing sets of integers.

**Definition 2.1.** A *Cantor forcing* is a partial order  $(\mathbb{P}, \leq)$  equipped with a function  $F \mapsto \dot{F}$  from maximal filters towards Cantor space  $2^{\mathbb{N}}$ , such for all  $c \in \mathbb{P}$ , and all  $\sigma \in 2^{<\mathbb{N}}$  for which  $[c] \cap [\sigma] \neq \emptyset$ , there exists a condition  $d \leq c$  such that  $[d] \subseteq [\sigma]$ . Here,  $[c]$  denotes the set  $\{\dot{F} : F \in [c]_{\in}\}$ . ◇

### 2.1. Semantic forcing relation

By abuse of language, we will say that a set  $G \in 2^{\mathbb{N}}$  is *sufficiently generic* for a Cantor forcing if it is of the form  $\dot{F}$  for a sufficiently generic filter  $F$ .

**Definition 2.2.** A condition  $c$  *semantically forces* a requirement  $\mathcal{R}$ , in which case we write  $c \Vdash \mathcal{R}$  if  $\dot{F}$  satisfies  $\mathcal{R}$  for any maximal filter  $F$  sufficiently generic and containing  $c$ .  $\diamond$

This semantic definition gives us “for free” some properties of the relation:

**Proposition 2.3.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing,  $c, d \in \mathbb{P}$  and  $\mathcal{R}$  a requirement.

(1) If  $c \Vdash \mathcal{R}$  and  $d \leq c$ , then  $d \Vdash \mathcal{R}$ .

(2) If  $c \Vdash \mathcal{R}$ , then  $c \nVdash \neg \mathcal{R}$ .  $\star$

PROOF. (1) Let  $F$  be a sufficiently generic filter containing  $d$ . By upward-closure of the filters,  $c \in F$ , so as  $c$  forces  $\mathcal{R}$ ,  $\dot{F}$  satisfies  $\mathcal{R}$ . It follows that  $d$  forces  $\mathcal{R}$ .

(2) If  $c$  forces  $\mathcal{R}$  and  $\neg \mathcal{R}$ , then for any sufficiently generic filter  $F$  containing  $c$ ,  $\dot{F}$  satisfies  $\mathcal{R}$  and  $\neg \mathcal{R}$ , contradiction.  $\blacksquare$

**Exercise 2.4.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing,  $c \in \mathbb{P}$  and  $\mathcal{R}$  be a requirement. Show that if the set  $\{d \in \mathbb{P} : d \Vdash \mathcal{R}\}$  is dense below  $c$ , then  $c \Vdash \mathcal{R}$ .  $\diamond$

On the other hand, some properties are much less obvious to prove. We will see for example that for any arithmetic requirement  $\mathcal{R}$ , the set of conditions which force  $\mathcal{R}$  or which force  $\neg \mathcal{R}$  is dense. As in the case of Cohen forcing with the partial order of the strings  $(2^{<\mathbb{N}}, \supseteq)$ , we will define a syntactic forcing relation which will allow us to account for this phenomenon.

We first insist on the three fundamental properties that a syntactic forcing relation must have.

**Definition 2.5 (Forcing relation).** Given a Cantor forcing  $(\mathbb{P}, \leq)$ , we call *forcing relation* a relation  $\Vdash^\circ$  satisfying the following properties for all  $c, d \in \mathbb{P}$  and any arithmetic requirement  $\mathcal{R}$ .

(1) If  $c \Vdash^\circ \mathcal{R}$  then  $c \Vdash \mathcal{R}$ .

(2) If  $c \Vdash^\circ \mathcal{R}$  and  $d \leq c$ , then  $d \Vdash^\circ \mathcal{R}$ .

(3) The set  $\{c \in \mathbb{P} : c \Vdash^\circ \mathcal{R} \text{ or } c \Vdash^\circ \neg \mathcal{R}\}$  is dense.  $\diamond$

Properties (1-3) correspond to propositions 10-4.8, 10-4.6 and 10-4.7 for Cohen forcing. It follows immediately from (1) that if  $c \Vdash^\circ \mathcal{R}$ , then  $c \nVdash^\circ \neg \mathcal{R}$ . Recall that a set  $S \subseteq \mathbb{P}$  is dense below  $c \in \mathbb{P}$  if for all  $d \leq c$ , there exists an  $e \leq d$  such that  $e \in S$ .

**Proposition 2.6.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing,  $c \in \mathbb{P}$  and  $\mathcal{R}$  an arithmetic requirement. Let  $\Vdash^\circ$  be a forcing relation. Then  $c \Vdash \mathcal{R}$  iff  $\{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$  is dense below  $c$ . ★

PROOF. Suppose that  $c \Vdash^\circ \mathcal{R}$ . Let  $d \leq c$ . By Definition 2.5 (3), there exists an  $e \leq c$  such that  $e \Vdash^\circ \mathcal{R}$  or  $e \Vdash^\circ \neg \mathcal{R}$ . If the second case arises, then by Definition 2.5 (1),  $e \Vdash \neg \mathcal{R}$ . So we have  $e \Vdash \mathcal{R}$  and  $e \Vdash \neg \mathcal{R}$ , which contradicts Theorem 2.3 (2). The first case therefore arises. We have shown the density of the set  $\{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$  under  $c$ .

Conversely, suppose that the set  $\{d \in \mathbb{P} : d \Vdash^\circ \mathcal{R}\}$  is dense below  $c$ . Let  $F$  be a sufficiently generic filter containing  $c$ . By genericity, there exists  $d \in F$  such that  $d \Vdash^\circ \mathcal{R}$ , and by Definition 2.5 (1),  $d \Vdash \mathcal{R}$ , so  $\dot{F}$  satisfies  $\mathcal{R}$ . It follows that  $c \Vdash \mathcal{R}$ . ■

## 2.2. Syntactic forcing relation

We now define our syntactic forcing relation  $\Vdash^*$ , which directly generalizes the relation  $\Vdash^*$  defined in Section 10-4 for Cohen forcing. As for Cohen forcing, the interest of the relation  $c \Vdash^* \mathcal{R}$  is that it is simple to define: relative to  $\mathbb{P}$ , it has the same arithmetic complexity as that of the requirement  $\mathcal{R}$ . We will see in the following sections that  $\mathbb{P}$  as a partial order is unfortunately rarely computable, which leads to additional computational complexities which must be dealt with on a case-by-case basis.

**Definition 2.7.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing. We define the relation  $\Vdash^*$  for all  $c \in \mathbb{P}$  and any arithmetic requirement  $\mathcal{R}$ :

- (1)  $c \Vdash^* \mathcal{R}$  for a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement  $\mathcal{R}$  iff all  $X \in [c]$  satisfy  $\mathcal{R}$ .
- (2)  $c \Vdash^* \exists x \mathcal{R}(x)$  for a  $\Pi_k^0$  requirement  $\mathcal{R}(x)$  with  $k \geq 1$  iff there is an  $n \in \mathbb{N}$  such that  $c \Vdash^* \mathcal{R}(n)$ .
- (3)  $c \Vdash^* \forall x \mathcal{R}(x)$  for a  $\Sigma_k^0$  requirement  $\mathcal{R}(x)$  with  $k \geq 1$  iff for all  $d \leq c$  and all  $n \in \mathbb{N}$ ,  $d \not\Vdash^* \neg \mathcal{R}(n)$ . ◆

Note that just as in the case of Cohen forcing, we have  $c \Vdash^* \forall x \mathcal{R}(x)$  iff  $\forall d \leq c \ d \not\Vdash^* \exists x \neg \mathcal{R}(x)$ . We leave as an exercise the proof that the relation  $\Vdash^*$  respects the items (1-3) of Definition 2.5. Each time, it is a simple proof by induction on the complexity of the requirements.

**Exercize 2.8. (★)** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing,  $c, d \in \mathbb{P}$  and  $\mathcal{R}$  be an arithmetic requirement. Show that if  $c \Vdash^* \mathcal{R}$  and  $d \leq c$ , then  $d \Vdash^* \mathcal{R}$ . ◇

**Exercise 2.9. (★)** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing and let  $\mathcal{R}$  be an arithmetical requirement. Show that  $\{c \in \mathbb{P} : c \Vdash^* \mathcal{R} \text{ or } c \Vdash^* \neg \mathcal{R}\}$  is dense.  $\diamond$

**Exercise 2.10. (★)** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing,  $c \in \mathbb{P}$  and  $\mathcal{R}$  an arithmetic requirement. Show that if  $c \Vdash^* \mathcal{R}$ , then  $c \Vdash \mathcal{R}$ .  $\diamond$

### Complexity of $\Vdash$

The complexity of  $\Vdash^*$  has consequences on the complexity of the semantic forcing relation  $\Vdash$ . According to the previous exercises,  $\Vdash^*$  respects the items (1-3) of Definition 2.5. So according to Proposition 2.6, the relation  $c \Vdash \mathcal{R}$  is equivalent to  $\forall d \leq c \exists e \leq d e \Vdash^* \mathcal{R}$ , which for example for a  $\Sigma_n^0$  requirement will be a  $\Pi_{n+1}^0(\mathbb{P})$  predicate. This is a considerable simplification of the semantic relation, but which remains —relative to  $\mathbb{P}$ — of arithmetical complexity greater than that of the requirements which it forces, which will make us prefer the relation  $\Vdash^*$ .

## 3. Forcing with trees

Tree-based forcing is one of the major families of forcing. We detail here two examples, already encountered in the previous chapters: Jockusch-Soare forcing and the computable Sacks forcing. These notions were originally created to control the single jump, via the forcing of  $\Sigma_1^0/\Pi_1^0$  requirements or the double jump, via the forcing of  $\Sigma_2^0/\Pi_2^0$  requirements. We will discuss this again in Section 4, and we will content ourselves for the moment with seeing how these forcings have been used implicitly on several occasions in this book.

### 3.1. Jockusch-Soare forcing

The Jockusch-Soare forcing corresponds to the partial order of non-empty  $\Pi_1^0$  classes, partially ordered by the inclusion relation. We can associate with any maximal filter  $F$  on this order a set of integers  $\dot{F} \in 2^{\mathbb{N}}$  which is the element of the singleton  $\bigcap_{\mathcal{P} \in F} \mathcal{P}$ . Equipped with this interpretation of the filters, Jockusch-Soare forcing is a Cantor forcing (see Definition 2.1).

We have already encountered several uses of Jockusch-Soare forcing in Chapter 8 on  $\Pi_1^0$  classes and PA degrees. Here is a reformulation of the computably dominated basis theorems and the cone avoidance basis theorems, via the forcing vocabulary:

**Theorem (Reformulation of Theorem 8-4.5 and Theorem 8-4.7)**

*Let  $A$  be a non-computable set. Let  $G$  be a sufficiently generic set for Jockusch-Soare forcing. Then,  $G$  is computably dominated and does not compute  $A$ .*

PROOF. Let  $(\mathbb{P}, \leq)$  be Jockusch-Soare forcing, i.e., the set of non-empty  $\Pi_1^0$  classes ordered by inclusion. The proof of Theorem 8-4.5 shows the following: for any functional  $\Phi_e$  and all  $c \in \mathbb{P}$ , there exists  $d \leq c$  such that  $d \Vdash^* \exists n \Phi_e(G, n) \uparrow$  or there exists  $d \leq c$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $d \Vdash^* \Phi_e(G, n) \downarrow < f(n)$  for all  $n$ . So the set  $C_e$  of the conditions which force  $\Phi_e$  to be partial or else bounded by a computable function is dense. Any sufficiently generic filter contains a condition in each of  $C_e$ . So if  $G \in 2^{\mathbb{N}}$  is sufficiently generic it is computably dominated.

The proof of Theorem 8-4.7 shows the following: for any functional  $\Phi_e$ , the set  $D_e = \{c \in \mathbb{P} : c \Vdash^* \exists n \Phi_e(G, n) \uparrow \text{ or } \exists n c \Vdash^* \Phi_e(G, n) \downarrow \neq A(n)\}$  is dense. So if  $G$  is sufficiently generic, it does not compute  $A$ . ■

Regarding the low basis theorem, things are different: it is indeed Jockusch-Soare forcing that we use to build a low set, but it is an effective use of this forcing, where the passage from a step  $n$  to a step  $n + 1$  is controlled using  $\emptyset'$ . It is therefore not strictly speaking a forcing result, in the sense that it is not a property satisfied by any sufficiently generic set, but on the contrary by a small countable class of sets which are not very generic.

We have seen with Cohen forcing that any sufficiently generic set differs from a countable quantity of sets fixed in advance (see Theorem 10-5.4). In particular, no sufficiently generic set for Cohen forcing is arithmetic. This is not the case with Jockusch-Soare forcing. Indeed, for any computable set  $X$ , the singleton  $\{X\}$  is a  $\Pi_1^0$  class and the unique filter containing  $\{X\}$  —namely the filter of all  $\Pi_1^0$  classes having  $X$  as an infinite path— is maximal. The set  $X$  is therefore as generic as we want under the condition  $\{X\}$ .

However, it is possible to slightly modify Jockusch-Soare forcing in order to avoid computable elements.

**Proposition 3.2.** Let  $(\mathbb{P}, \leq)$  be the partial order of non-empty  $\Pi_1^0$  classes without computable element, ordered by inclusion. Let  $(A_n)_{n \in \mathbb{N}}$  be any sequence of sets. If  $G \in 2^{\mathbb{N}}$  is generic enough for  $\mathbb{P}$  it is different from each  $A_n$ . ★

PROOF. Let us fix any set  $A$  and show that if  $G$  is sufficiently generic, it is different from  $A$ . The result will follow automatically for the continuation  $(A_n)_{n \in \mathbb{N}}$ .

Let  $D \subseteq \mathbb{P}$  be the set of  $\Pi_1^0$  classes of  $\mathbb{P}$  not containing  $A$ . Let us show that  $D$  is dense in  $\mathbb{P}$ . Let  $\mathcal{P} \in \mathbb{P}$ . By Proposition 8-3.6, the class  $\mathcal{P}$  contains

at least two elements. Let  $B \in \mathcal{P}$  be such that  $B \neq A$ , and let  $n \in \mathbb{N}$  be such that  $A(n) \neq B(n)$ . Then the class  $\mathcal{Q} = \{X \in \mathcal{P} : X(n) = B(n)\}$  is a non-empty  $\Pi_1^0$  class included in  $\mathcal{P}$  and such that  $\mathcal{Q} \in D$ . So  $D$  is dense. ■

The modification of Jockusch-Soare forcing of the preceding proposition can be declined in multiple ways to obtain different results: one can consider the partial order of the non-empty  $\Pi_1^0$  classes containing only PA degrees, or even those of the non-empty  $\Pi_1^0$  classes containing only random sets in the sense of Martin-Löf (see Chapter 18).

### 3.2. Computable Sacks forcing

Sacks forcing generally designates the partial order of perfect trees of  $2^{<\mathbb{N}}$ , without any particular effectiveness restriction. We restrict ourselves in Computability Theory to computable perfect trees, which have already been approached via the notion of *f-tree*.

Recall that an *f-tree* is a total function  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that for all  $\sigma, \tau \in \text{dom } T$ ,  $\sigma \preceq \tau$  if and only if  $T(\sigma) \preceq T(\tau)$ . A *sub-f-tree* of an f-tree  $T$  is an f-tree  $S$  such that  $\text{Im } S \subseteq \text{Im } T$ . An f-tree  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  extends into a function  $\hat{T} : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  defined by  $\{\hat{T}(X)\} = \bigcap_n [T(X \upharpoonright_n)]$ . A *path* of  $T$  is an element of  $\text{Im } \hat{T}$ . We denote by  $[T]$  the set of paths of  $T$ .

We call *computable Sacks forcing* the set of computable f-trees partially ordered by the sub-f-tree relation. As there will be no possible ambiguity in this book, we will sometimes simply say *Sacks forcing*. We can associate with any maximal filter  $F$  on this order a set of integers  $\dot{F} \in 2^{\mathbb{N}}$  which is the element of the singleton  $\bigcap_{T \in F} [T]$ . Equipped with this interpretation of the filters, Sacks forcing is a Cantor forcing (see Definition 2.1).

#### Remark

Recall that a class  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  is perfect if  $\mathcal{P} = [T]$  for an f-tree  $T$ . Sacks forcing is not, however, the restriction of Jockusch-Soare forcing to perfect  $\Pi_1^0$  classes: some perfect  $\Pi_1^0$  classes cannot necessarily be represented by a computable f-tree. Indeed, for any computable f-tree  $T$ ,  $[T]$  contains an infinity of computable elements, namely  $T(X)$  for any computable set  $X$ , which is for example not the case of the  $\Pi_1^0$  class of  $\text{DNC}_2$  functions, which is nevertheless a perfect class.

As explained in the previous remark, any computable f-tree possesses an infinity of computable paths. On the other hand, any sufficiently generic set for Sacks forcing will be non-computable:

**Exercise 3.3. (★)** Let  $(A_n)_{n \in \mathbb{N}}$  be any sequence of sets. Show that any sufficiently generic set  $G$  for Sacks forcing is different from each  $A_n$ . ◇

A careful examination of the first proof that we have given of the existence of a computably dominated degree different from  $\mathbf{0}$ , using f-trees, in fact shows that we have the following result.

**Theorem (reformulation du théorème 7-5.6)**

*Let  $G$  be a sufficiently generic set for Sacks forcing. Then,  $G$  is non-computable and computably dominated.*

PROOF. Let  $(\mathbb{P}, \leq)$  be Sacks forcing. The proof of Theorem 7-5.6 shows the following: for any functional  $\Phi_e$  and any  $c \in \mathbb{P}$ , there exists  $d \leq c$  such that  $d \Vdash^* \exists n \Phi_e(G, n) \uparrow$  or there exists  $d \leq c$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $d \Vdash^* \Phi_e(G, n) \downarrow < f(n)$  for all  $n$ . So the set  $C_e$  of the conditions which force  $\Phi_e$  to be partial or else bounded by a computable function is dense. Any sufficiently generic filter contains a condition of each of  $C_e$ . So if  $G \in 2^{\mathbb{N}}$  is sufficiently generic, it is computably dominated.

Moreover, Exercize 3.3 shows that if  $G$  is sufficiently generic, it is not computable. ■

**Exercize 3.5. (★)** Let  $A$  be a non-computable set. Show that if  $G$  is sufficiently generic for Sacks forcing, it does not compute  $A$ . ◇

**Exercize 3.6. (★★)** A set  $X$  is *computably traceable* (Terwijn and Zambella [221]) if there exists a computable bound  $h : \mathbb{N} \rightarrow \mathbb{N}$  such that for any  $X$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , there exists a computable sequence  $(T_n)_{n \in \mathbb{N}}$  of finite sets such that  $|T_n| \leq h(n)$  and such that  $f(n) \in T_n$  for all  $n$ .

1. Show that if  $X$  is sufficiently generic for Sacks forcing, it is computably traceable.
2. Show that if  $X$  is computably traceable via a computable bound  $h$ , then it is computably traceable for any computable bound  $h'$  such that  $h'(n) \leq h'(n+1)$  and  $\lim_n h(n) = +\infty$  (Terwijn and Zambella [221]).
3. Let  $(2^n)^{\mathbb{N}}$  be the set of functions  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(n) < 2^n$ . Let  $(2^n)^{<\mathbb{N}}$  be the set of function prefixes of  $(2^n)^{\mathbb{N}}$ . Let  $(\mathbb{P}, \leq)$  be the forcing conditions given by  $T \in \mathbb{P}$  if  $T \subseteq (2^n)^{<\mathbb{N}}$  is a computable tree which satisfies the following property: for all  $\sigma \in T$  there exists  $\tau \in T$  with  $\tau \succeq \sigma$  such that for all  $i < 2^{|\tau|}$  the string  $\tau i$  is in  $T$ . In other words, each node has a maximally branching extension. Show that any set sufficiently generic for this forcing is computably dominated and not computably traceable.

◇

## 4. Computational complexity and forcing question

Cohen forcing presents a particularity that distinguishes it from other computability-theoretic forcings: the partial order  $(2^{<\mathbb{N}}, \succeq)$  is computable: the set  $2^{<\mathbb{N}}$  is computable and given  $\sigma \in 2^{<\mathbb{N}}$ , we can compute the set of strings  $\tau \succeq \sigma$ . Another of its particularities is that the forcing relation of  $\Sigma_1^0$  and  $\Pi_1^0$  requirements is respectively  $\Sigma_1^0$  and  $\Pi_1^0$ . These two properties make it possible to obtain Proposition 10-5.2: if  $\mathcal{R}$  is a  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) requirement, the predicate  $\sigma \Vdash^* \mathcal{R}$  is  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ). This very fine control of the complexity of the forcing relation then allows a fine control of the iterated jumps of the generic sets, in order to obtain for example the following results:

1. Corollary 10-5.8: if  $X$  is not  $\emptyset^{(n)}$ -computable and if  $G$  is sufficiently generic then  $G^{(n)}$  does not compute  $X$ .
2. Theorem 10-5.9: if  $G$  is sufficiently generic then  $G \oplus \emptyset^{(n)} \geq_T G^{(n)}$ .

It is in general the kind of property that one seeks to obtain with any forcing notion: the control of the truth value of arithmetic requirements. The finer this control is, the better the theorems we get. Unfortunately, things will rarely be as simple as with Cohen forcing. Let us examine the computational complexity of Jockusch-Soare forcing and that of Sacks forcing.

### 4.1. Complexity of partial orders

To talk about the computability of partial orders of abstract objects, it is necessary to first agree on their representation by sets of integers. For Cohen forcing, the partial order of the strings admits a natural bijective coding, while the notions of Jockusch-Soare and Sacks forcing involve more complex objects.

**Jockusch-Soare forcing.** A natural idea is to identify  $\mathbb{P}$  with the set of codes of non-empty  $\Pi_1^0$  classes. Note that we then have repetitions because several integers code for the same class, which in practice is not a problem.

**Proposition 4.1.** The partial order of Jockusch-Soare forcing is  $\Pi_2^0$ . ★

PROOF. Note that the set of codes of non-empty  $\Pi_1^0$  classes is  $\Pi_1^0$ . Indeed if a  $\Pi_1^0$  class is empty then the computable tree  $T \subseteq 2^{<\mathbb{N}}$  which represents it has no infinite path, and according to König's lemma there exists an integer  $n$  such that no length string  $n$  belongs to  $T$ , which is a  $\Sigma_1^0$  event.

Now, given two non-empty  $\Pi_1^0$  classes  $\mathcal{P}, \mathcal{Q}$  we have  $\mathcal{P} \subseteq \mathcal{Q}$  iff  $\forall t \exists s \mathcal{P}[s] \subseteq \mathcal{Q}[t]$ , which is a  $\Pi_2^0$  predicate. ■

Note that it is possible to improve the complexity of the partial order of Jockusch-Soare forcing, by working only on a well-chosen part of the codes of non-empty  $\Pi_1^0$  classes. There is indeed a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(e)$  always codes for a non-empty  $\Pi_1^0$  class and such that if  $e$  codes for a non-empty  $\Pi_1^0$  class then  $e$  and  $f(e)$  code for the same class: given  $e$ , it suffices to stop the co-enumeration of the corresponding  $\Pi_1^0$  class if this one is about to make the class empty.

We can also in practice improve the complexity of the partial order, by restricting there too the set of codes on which we work: given the code  $e$  of a non-empty  $\Pi_1^0$  class  $\mathcal{P}$ , we can consider the set  $A$  of the codes of all the  $\Pi_1^0$  classes  $\mathcal{Q}$  such that  $\mathcal{P} \cap \mathcal{Q}$  is not empty. The set  $A$  is  $\Pi_1^0$  and contains at least one code corresponding to each non-empty  $\Pi_1^0$  class included in  $\mathcal{P}$ . This does not of course make it possible to decide whether two codes represent comparable forcing conditions, but it simplifies the computational complexity of finding the list of all the conditions of  $\mathbb{P}$  found under a given condition.

**Computable Sacks Forcing.** Here, the natural idea is to identify the conditions of the computable Sacks forcing with codes of functions  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  corresponding to f-trees. Againn the coding is not injective, but in practice this is not a problem.

**Proposition 4.2.** The partial order of computable Sacks forcing is  $\Pi_2^0$ . ★

PROOF. The set of conditions is the set of codes  $e$  of functions  $T_e : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that  $\forall \sigma \in 2^{<\mathbb{N}} \exists t \in \mathbb{N} T_e(\sigma)[t] \downarrow$  and such that  $\forall \sigma_0, \sigma_1 \in 2^{<\mathbb{N}} \sigma_0 \prec \sigma_1 \leftrightarrow T_e(\sigma_0) \prec T_e(\sigma_1)$ . These are  $\Pi_2^0$  conditions to check.

Given a computable f-tree  $T$ , the set of codes  $e$  of computable f-trees  $S_e$  such that  $[S_e] \subseteq [T]$  is the set of codes  $e$  of f-tree (which is a  $\Pi_2^0$  condition), which checks, for any string  $\sigma$  and for any integer  $n$  sufficiently large such that  $|T(\tau)| \geq |S_e(\sigma)|$  for any string  $\tau$  of size  $n$ , the existence of a string  $\tau$  such that  $|\tau| \leq n$  for which  $S_e(\sigma) = T(\tau)$ . This is a  $\Pi_1^0$  condition. ■

#### 4.2. Forcing $\Sigma_1^0/\Pi_1^0$ requirements

The complexity of the forcing relation for the  $\Sigma_1^0$  and  $\Pi_1^0$  requirements is not directly linked to the complexity of the partial order. We will see in particular that the syntactic forcing relation of Jockusch-Soare forcing is more complex than that of Sacks forcing. **TODO, from here**

**Jockusch-Soare forcing.** The complexity of the forcing relation does not follow that of the complexity of the requirements to be forced, including already for the  $\Sigma_1^0/\Pi_1^0$  requirements.

**Proposition 4.3.** Let  $(\mathbb{P}, \leq)$  be Jockusch-Soare forcing and  $\mathcal{R}$  a requirement. Let  $\mathcal{P}_e$  be the non-empty  $\Pi_1^0$  class of code  $e$ .

- (1) If  $\mathcal{R}$  is  $\Sigma_1^0$ , then the predicate  $\mathcal{P}_e \Vdash^* \mathcal{R}$  is  $\Sigma_1^0$
- (2) If  $\mathcal{R}$  is  $\Pi_1^0$ , then the predicate  $\mathcal{P}_e \Vdash^* \mathcal{R}$  is  $\Pi_2^0$  ★

PROOF. Let  $T_e \subseteq 2^{<\mathbb{N}}$  be a computable tree such that  $[T_e] = \mathcal{P}_e$ .

(1) Let  $\mathcal{R}$  be of the form  $\Phi(G, 0) \downarrow$  for a functional  $\Phi$ . Then,  $\mathcal{P}_e \Vdash^* \mathcal{R}$  iff  $\Phi(X, 0) \downarrow$  for all  $X \in \mathcal{P}_e$  iff (by the use property and König's lemma)  $\exists n \forall \sigma \in T_e \cap 2^n, \Phi(\sigma, 0) \downarrow$ .

(2) Let  $\mathcal{R}$  be of the form  $\Phi(G, 0) \uparrow$  for a functional  $\Phi$ . Then, we can obtain the code  $a \in \mathbb{N}$  of the  $\Pi_1^0$  class such that  $\mathcal{P}_a = \{X : \Phi(G, 0) \uparrow\}$ . We then have  $\mathcal{P}_e \Vdash^* \mathcal{R}$  iff  $\mathcal{P}_e \subseteq \mathcal{P}_a$ . As seen in the proof of Proposition 4.2, the inclusion relation on the codes of  $\Pi_1^0$  classes is  $\Pi_2^0$ . ■

We will see in the next two sections how to get around the problem of complexity raised by the previous proposition.

**Exercise 4.4. (★)** Let  $(\mathbb{P}, \leq)$  be the partial order of infinite computable trees  $T \subseteq 2^{<\mathbb{N}}$ . Let  $\Vdash^\circ$  be the relation defined by

- (1)  $T \Vdash^\circ \exists n \Phi(G, n) \downarrow$  if there exists  $n, t \in \mathbb{N}$  such that for any  $\sigma \in T$  of length  $t$ ,  $\Phi(\sigma, n) \downarrow$
- (2)  $T \Vdash^\circ \forall n \Phi(G, n) \uparrow$  if for all  $\sigma \in T$  and all  $n < |\sigma|$ ,  $\Phi(\sigma, n) \uparrow$

To show that

- (a)  $(\mathbb{P}, \leq)$  is a Cantor forcing, where  $[T]$  is the class of the paths of  $T$ .
- (b) The sufficiently generic sets for Jockusch-Soare forcing and for  $(\mathbb{P}, \leq)$  coincide.
- (c) The set  $\{T \in \mathbb{P} : T \Vdash^\circ \exists n \Phi(G, n) \downarrow \text{ or } T \Vdash^\circ \forall n \Phi(G, n) \uparrow\}$  is dense in  $(\mathbb{P}, \leq)$
- (d) The relations  $T \Vdash^\circ \exists n \Phi(G, n) \downarrow$  and  $T \Vdash^\circ \forall n \Phi(G, n) \uparrow$  are respectively  $\Sigma_1^0$  and  $\Pi_1^0$ .

◇

**Computable Sacks Forcing.** The forcing of  $\Sigma_1^0$  and  $\Pi_1^0$  requirements is in this case of minimal complexity.

**Proposition 4.5.** Let  $(\mathbb{P}, \leq)$  be Sacks forcing and let  $\mathcal{R}$  be a requirement. Let  $T_e$  be the computable f-tree of code  $e$ .

- (1) If  $\mathcal{R}$  is  $\Sigma_1^0$ , then the predicate  $T_e \Vdash^* \mathcal{R}$  is  $\Sigma_1^0$
- (2) If  $\mathcal{R}$  is  $\Pi_1^0$ , then the predicate  $T_e \Vdash^* \mathcal{R}$  is  $\Pi_1^0$  ★

PROOF. (1) Let  $\mathcal{R}$  be of the form  $\Phi(G, 0) \downarrow$  for a functional  $\Phi$ . We have  $T_e \Vdash^* \mathcal{R}$  iff there exists  $n, t$  such that  $\Phi(\sigma, 0)[t] \downarrow$  for any string  $\sigma \in \text{Im } T_e$  of size  $n$ .

- (2) Let  $\mathcal{R}$  be of the form  $\Phi(G, 0) \uparrow$  for a functional  $\Phi$ . We have  $T_e \Vdash^* \mathcal{R}$  iff for all  $\sigma \in \text{Im } T_e$  and for all  $t$ , we have  $\Phi(\sigma, 0)[t] \uparrow$ . ■

**Continuity of the Turing jump.** We now see a theorem that contrasts with the fact that every sufficiently generic set for Cohen forcing is generalized low. This is generally not the case for tree-based forcings, and it is in particular not the case for computable Sack forcing.

**Theorem 4.6**

*Let  $X$  be any set. Let  $G$  be a sufficiently generic set for computable Sacks forcing. Then,  $X \oplus G \not\leq_T G'$ .*

PROOF. Let  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  be a computable f-tree. Let  $\Phi_e$  be a Turing functional. We will build a subf-tree  $S$  of  $T$  such that for each of the paths  $G$  of  $S$  we have  $\Phi_e(X \oplus G, n) \neq G'(n)$  for a certain  $n$ . First consider a computable sub-tree  $S$  of  $T$  such that for all  $\sigma \in \text{Im } S$  there exists  $\tau \succeq \sigma$  with  $\tau \in \text{Im } T$  and  $\tau \notin \text{Im } S$ .

Now consider the code  $a$  of the partial computable functional  $\Phi_a$  such that  $\Phi_a(Y, a) \uparrow$  for all  $Y \in [S]$  and such that  $\Phi_a(Y, a) \downarrow$  for all  $Y \notin [S]$ . Note in particular that  $\Phi_a(Y, a) \downarrow$  for all  $Y \in [T] \setminus [S]$ . Suppose first that  $\Phi_e(X \oplus \sigma, a) \uparrow \notin \{0, 1\}$  for all  $\sigma \in \text{Im } S$ . Then, we can take  $S$  as a forcing extension of  $T$  in order to force the partiality of  $\Phi_e$  on the input  $a$ . Otherwise let  $\sigma \in \text{Im } S$  be such that  $\Phi_e(X \oplus \sigma, a) \downarrow = i$  for  $i \in \{0, 1\}$ . If  $i = 0$ , then we choose a string  $\tau \succeq \sigma$  such that  $\tau \in \text{Im } T$  and  $\tau \notin \text{Im } S$ , and we take as a forcing extension a subf-tree of  $T$  whose image only contains extensions of  $\tau$ . Note that we then have  $\Phi_a(Y, a) \downarrow$  for any path  $Y$  of our forcing extension. If  $i = 1$ , then we take as a forcing extension the subf-tree of  $S$  whose image only contains extensions of  $\sigma$ . Note that we then have  $\Phi_a(Y, a) \uparrow$  for any path  $Y$  of our forcing extension.

In both cases we force  $\Phi_e(X \oplus G, a)$  to be different from  $G'(a)$  for any set  $G$  of our forcing condition. ■

An analogous theorem for Jockusch-Soare forcing will not necessarily be true, for example because of the existence of  $\Pi_1^0$  classes containing a single computable element. Even if we restrict ourselves to  $\Pi_1^0$  classes which do not contain computable points, things are not so simple and will depend on the  $\Pi_1^0$  class with which we start the forcing.

**Exercise 4.7. (★★)** Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class. Then,  $\mathcal{P}$  is *thin* (definition due to Downey [45]) if for any non-empty  $\Pi_1^0$  subclass  $\mathcal{Q} \subseteq \mathcal{P}$ , there exists a finite sequence of cylinders  $[\sigma_0], \dots, [\sigma_n]$  such that  $\mathcal{Q} = \mathcal{P} \cap ([\sigma_0] \cup \dots \cup [\sigma_n])$ .

1. Show the existence of a perfect thin  $\Pi_1^0$  class (this is an algorithm of the priority method type as explained in Chapter 13).
2. Show that if  $\mathcal{P}$  is thin and if  $X \in \mathcal{P}$  then  $X \oplus \emptyset'' \not\geq_T X'$ .

◇

**Exercise 4.8. (★)** This exercise anticipates on Part II within which Lemma 18-3.3 should be useful. We consider a variant of Jockusch-Soare forcing with  $\Pi_1^0$  classes containing only random ones, in the sense of Martin-Löf. Show that for any  $X$  and any set  $Z$  sufficiently generic for this forcing we have  $Z \oplus X \not\geq_T Z'$ .

◇

#### 4.2.1. Forcing $\Sigma_2^0/\Pi_2^0$ requirements

Tree-based forcing are generally suitable, not to force  $\Sigma_1^0$  or  $\Pi_1^0$  requirements (as shown by Theorem 4.6 they do not allow to obtain generalized low sets), but to force  $\Sigma_2^0$  or  $\Pi_2^0$  requirements, on which they work particularly well. We saw in Section 10-4 that in the case of Cohen forcing, if we define the forcing relation by “any maximal filter satisfies the requirement”, then the set of conditions forcing a  $\Pi_2^0$  requirement or its negation, is not dense in general. This led us to define the forcing relation for any *sufficiently generic* maximal filter.

Unlike Cohen forcing, tree-based forcings, like Jockusch-Soare forcing or Sacks forcing, generally the existence of forcing conditions whose members all satisfy  $\Sigma_2^0$  or  $\Pi_2^0$  formulas. The reader will be able to note that it is indeed this mechanism which is at work to force a generic set to be computably dominated. To see this, we introduce the notion of *forcing question*, denoted  $? \vdash$ , which will be developed and studied in the following sections for arbitrary arithmetic requirements.

**Definition 4.9.** Let  $\mathcal{R}$  be a  $\Sigma_2^0$  requirement corresponding to  $\exists n \Phi(G, n) \uparrow$  for a functional  $\Phi$ .

(1) Let  $\mathcal{P}$  be a condition of Jockusch-Soare forcing. We define

$$\mathcal{P} ? \vdash \exists n \Phi(G, n) \uparrow$$

if there is  $n$  such that  $\mathcal{P} \Vdash^* \Phi(G, n) \downarrow$ .

(2) Let  $T$  be a computable Sacks forcing condition. We define

$$T ? \vdash \exists n \Phi(G, n) \uparrow$$

if there exists  $n$  and  $\sigma$  such that  $T \restriction_\sigma \Vdash^* \Phi(G, n) \uparrow$ , where  $T \restriction_\sigma$  is the  $f$ -tree  $S$  defined by  $S(\tau) = T(\sigma\tau)$ .  $\diamond$

The first interest of the forcing question relation that we have defined is its complexity, which is the same as that of the requirement concerned.

**Proposition 4.10.** Let  $c$  be a condition of Jockusch-Soare forcing or Sacks forcing. Let  $\mathcal{R}$  be a  $\Sigma_2^0$  requirement corresponding to  $\exists n \Phi(G, n) \uparrow$  for a functional  $\Phi$ . The predicate  $c ? \vdash \exists n \Phi(G, n) \uparrow$  is  $\Sigma_2^0$ .  $\star$

PROOF. Let's start with Jockusch-Soare forcing. According to Proposition 4.3, given  $n$ , the predicate  $\mathcal{P} \Vdash^* \Phi(G, n) \downarrow$  is  $\Sigma_1^0$  and therefore the predicate  $\mathcal{P} \Vdash^* \Phi(G, n) \downarrow$  is  $\Pi_1^0$ . So the predicate  $\exists n \mathcal{P} \Vdash^* \Phi(G, n) \downarrow$  is  $\Sigma_2^0$  and the predicate  $\mathcal{P} ? \vdash \Phi(G, n) \uparrow$  is therefore  $\Sigma_2^0$ .

Let's move on to the computable Sacks forcing. According to Proposition 4.5, given  $n$ , and an  $f$ -tree  $S$ , the predicate  $S \Vdash^* \Phi(G, n) \uparrow$  is  $\Pi_1^0$ . So the predicate  $\exists n \exists \sigma T \restriction_\sigma \Vdash^* \Phi(G, n) \uparrow$  is  $\Sigma_2^0$  and the predicate  $T ? \vdash \exists n \Phi(G, n) \uparrow$  is therefore  $\Sigma_2^0$ .  $\blacksquare$

The second interest of the forcing question relation, is that it allows to *decide* if a condition  $c$  can be extended into a condition  $d$  to force a  $\Sigma_2^0$  requirement, or the negation of this requirement. Moreover, in the case of  $\Sigma_2^0/\Pi_2^0$  formulas, we can find an extension  $d \leq c$  such that the requirement will be satisfied *for all the elements* of  $[d]$  (As for the  $\Sigma_1^0/\Pi_1^0$  requirements).

**Proposition 4.11.** Let  $c$  be a condition of Jockusch-Soare forcing or Sacks forcing. Let  $\mathcal{R}$  be a  $\Sigma_2^0$  requirement corresponding to  $\exists n \Phi(G, n) \uparrow$  for a functional  $\Phi$ .

1. If  $c ? \vdash \exists n \Phi(G, n) \uparrow$  then there exists  $d \leq c$  such that  $\exists n \Phi(X, n) \uparrow$  is true for all  $X \in [d]$ .
2. If  $c \not? \vdash \exists n \Phi(G, n) \uparrow$  then there exists  $d \leq c$  such that  $\forall n \Phi(X, n) \downarrow$  is

true for all  $X \in [d]$ .

In both cases, we can find  $d$  uniformly in  $c$  and  $\Phi$  using  $\emptyset''$ . ★

PROOF. Let's start with Jockusch-Soare forcing. Let  $\mathcal{P}$  be a non-empty  $\Pi_1^0$  class.

1. If  $\mathcal{P} \Vdash \exists n \Phi(G, n) \uparrow$  then there exists  $n$  such that  $\mathcal{P} \Vdash^* \Phi(G, n) \downarrow$ . This means that the class  $\mathcal{Q} = \{X \in \mathcal{P} : \Phi(X, n) \uparrow\}$  is a non-empty  $\Pi_1^0$  class. This is then our forcing extension.
2. If  $\mathcal{P} \nVdash \exists n \Phi(G, n) \uparrow$  then  $\mathcal{P} \Vdash^* \Phi(G, n) \downarrow$  for all  $n$ . In this case for all  $X \in \mathcal{P}$  we already have  $\forall n \Phi(X, n) \downarrow$ .

Note that  $\emptyset''$  can decide between the two cases and therefore find the appropriate forcing extension. Let's move on to the computable Sacks forcing. Let  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  be a computable f-tree.

1. If  $T \Vdash \exists n \Phi(G, n) \uparrow$  then there exists  $\sigma \in 2^{<\mathbb{N}}$  and  $n \in \mathbb{N}$  such that  $T \restriction_\sigma \Vdash^* \Phi(G, n) \uparrow$ . In this case the f-tree  $T \restriction_\sigma$  is our forcing extension.
2. If  $T \nVdash \exists n \Phi(G, n) \uparrow$  then for all  $\sigma \in 2^{<\mathbb{N}}$  and for all  $n \in \mathbb{N}$  we have  $T \restriction_\sigma \nVdash^* \Phi(G, n) \uparrow$ . This implies that for all  $\sigma \in 2^{<\mathbb{N}}$  and for all  $n \in \mathbb{N}$ , there exists  $\tau \succeq \sigma$  such that  $\Phi(T(\sigma\tau), n) \downarrow$ . Then, we proceed as in the proof of Theorem 7-5.6 to compute a subf-tree  $S$  of  $T$  such that  $\forall n \Phi(X, n) \downarrow$  for all  $X \in [S]$ .

Note that there again  $\emptyset''$  can decide between the two cases and therefore find the appropriate forcing extension. ■

Before seeing how to extend the forcing question relation to requirements of arbitrary complexity, let us see how to use the developments obtained so far to show that any set sufficiently generic for Sacks forcing or Jockusch-Soare forcing is generalized  $\text{low}_2$ .

**Theorem 4.12**

*If  $G$  is sufficiently generic for Sacks forcing or Jockusch-Soare forcing, it is generalized  $\text{low}_2$  in a strong sense:  $\emptyset'' \oplus G' \geq_T G''$ .*

PROOF. For all  $n$ , according to Lemma 10-5.6 the requirement  $n \in G''$  is  $\Sigma_2^0$ . According to Proposition 4.10 and Proposition 4.11 we can enumerate with the help of  $\emptyset''$  a set  $D_1$  of conditions  $c$  such that  $n \in X''$  for all  $X \in [c]$ , and a set  $D_2$  of conditions  $c$  such that  $n \notin X''$  for all  $X \in [c]$ , the whole such that  $D_1 \cup D_2$  is a dense set of conditions. If  $G$  is sufficiently generic, there exists a condition  $c \in D_1 \cup D_2$  such that  $G \in [c]$ . The condition  $c$  being a tree, we need  $G'$  to know if  $G \in [c]$ . Once the

condition  $c$  has been found such that  $G \in [c]$ , if  $c \in D_1$  then  $n \in G''$  and if  $c \in D_2$  then  $n \notin G''$ . ■

#### 4.2.2. Forcing $\Sigma_n^0/\Pi_n^0$ requirements

For more complex requirements, it is not possible to find conditions in which all the elements satisfy the requirement, and it is necessary to return to the inductive definition of forcing. Our objective is now to show an analogue of the theorem 10-5.7 of preservation of the arithmetic hierarchy. As the relation  $\Vdash^*$  for the Sacks and Jockusch-Soare forcings is too complex, we will extend and instead use the forcing question relation defined previously for the  $\Sigma_2^0$  requirements.

**Definition 4.13.** Let  $c$  be a condition of the Jockusch-Soare or Sacks forcing. Let  $\mathcal{R}$  be an arithmetic requirement. We define the relation  $c ?\vdash$  between  $c$  and  $\mathcal{R}$  as follows:

- (1)  $c ?\vdash \mathcal{R}$  for a  $\Sigma_1^0$  requirement  $\mathcal{R}$  iff  $c \Vdash^* \mathcal{R}$ .
- (2)  $c ?\vdash \exists x \mathcal{R}(x)$  where  $\mathcal{R}(x)$  is a  $\Pi_1^0$  requirement iff  $c ?\vdash \exists x \mathcal{R}(x)$  within the meaning of Definition 4.9.
- (3)  $c ?\vdash \exists x \mathcal{R}(x)$  where  $\mathcal{R}(x)$  is a  $\Pi_k^0$  requirement for  $k \geq 2$  iff there is a  $d \leq c$  and an  $n \in \mathbb{N}$  such that  $d ?\vdash \mathcal{R}(n)$ .
- (4)  $c ?\vdash \mathcal{R}$  where  $\mathcal{R}(x)$  is a  $\Pi_k^0$  requirement iff  $c \not\vdash \neg \mathcal{R}$ . ◇

We now extend Proposition 4.10 and Proposition 4.11, first by showing that the forcing question relation is  $\Sigma_n^0$  for  $\Sigma_n^0$  requirements:

**Proposition 4.14.** Let  $c$  be a condition of Jockusch-Soare forcing or Sacks forcing. Let  $\mathcal{R}$  be a  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) requirement. The relation  $c ?\vdash \mathcal{R}$  is  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ). ★

PROOF. The case where  $\mathcal{R}$  is a  $\Sigma_1^0$  requirement has been dealt with in Proposition 4.3 and Proposition 4.5. The case where  $\mathcal{R}$  is a  $\Sigma_2^0$  requirement was dealt with in Proposition 4.10.

Suppose that  $\mathcal{R}$  is a  $\Sigma_{n+1}^0$  requirement equal to  $\exists x \mathcal{Q}(x)$  where  $\mathcal{Q}(x)$  is a  $\Pi_k^0$  requirement for  $k \geq 2$ . Then,  $c ?\vdash \exists x \mathcal{Q}(x)$  iff there exists a forcing condition  $d$  and an integer  $n$  such that  $d \leq c$  and such that  $d ?\vdash \mathcal{Q}(n)$ . The predicate  $d \leq c$  is  $\Pi_2^0$  (in the case of Jockusch-Soare forcing and Sacks forcing) and the predicate  $d ?\vdash \mathcal{Q}(n)$  is by  $\Pi_k^0$  induction for  $k \geq 2$ . It follows that the predicate  $c ?\vdash \exists x \mathcal{Q}(x)$  is  $\Sigma_{k+1}^0$ .

Finally if  $\mathcal{R}$  is a  $\Pi_n^0$  requirement. Then,  $\mathcal{P} ?\vdash \mathcal{R}$  iff  $\mathcal{P} \not\vdash \neg \mathcal{R}$  which is  $\Pi_n^0$  by induction hypothesis. ■

We now show the extension of Proposition 4.11: if  $c ?\vdash \mathcal{R}$  then we will be able to find an extension  $d \leq c$  which will force  $\mathcal{R}$ , but be careful, it is here semantic forcing and not syntactic forcing.

**Proposition 4.15.** Let  $c$  be a condition of the Jockusch-Soare or Sacks forcing. Let  $\mathcal{R}$  be an arithmetic requirement. If  $c ?\vdash \mathcal{R}$  then there exists  $d \leq c$  such that  $d \Vdash \mathcal{R}$ . ★

PROOF. If  $\mathcal{R}$  is a  $\Sigma_1^0$  or  $\Pi_1^0$  requirement then the result is trivial by definition. If  $\mathcal{R}$  is a  $\Sigma_2^0$  or  $\Pi_2^0$  requirement then it is Proposition 4.11.

Suppose  $\mathcal{R} = \exists x \mathcal{S}(x)$  where  $\mathcal{S}(x)$  is a  $\Pi_k^0$  requirement for  $k \geq 2$ . Suppose that  $c ?\vdash \exists x \mathcal{S}(x)$ . By definition, there is an extension  $d \leq c$  and an integer  $n \in \mathbb{N}$  such that  $d ?\vdash \mathcal{S}(n)$ . By induction hypothesis, there exists an  $e \leq d$  such that  $e \Vdash \mathcal{S}(n)$ . In particular  $e \Vdash \exists n \mathcal{S}(n)$ .

Suppose  $\mathcal{R} = \forall x \mathcal{S}(x)$  where  $\mathcal{S}(x)$  is a  $\Sigma_k^0$  requirement for  $k \geq 2$ . Suppose that  $c ?\vdash \forall x \mathcal{S}(x)$ . Then,  $c ?\not\vdash \exists x \neg \mathcal{S}(x)$ . By definition, for any extension  $d \leq c$  and any  $n \in \mathbb{N}$ ,  $d ?\not\vdash \neg \mathcal{S}(n)$ . Let us show that for all  $n$ , the set  $D_n = \{d : d \Vdash \mathcal{S}(n)\}$  is dense below  $c$ . Let  $n \in \mathbb{N}$  and  $d \leq c$ . By definition as  $d ?\not\vdash \neg \mathcal{S}(n)$  then  $d ?\vdash \mathcal{S}(n)$ . By induction hypothesis, as  $d ?\vdash \mathcal{S}(n)$ , then there exists an extension  $e \leq d$  such that  $e \Vdash \mathcal{S}(n)$ . The set  $D_n$  is therefore dense below  $c$ . It follows from Exercize 2.4 that  $c \Vdash \mathcal{S}(n)$  for all  $n$ , therefore  $c \Vdash \forall x \mathcal{S}(x)$ , in other words  $c \Vdash \mathcal{R}$ . ■

This forcing question will allow us a fine control of what computes arbitrary iterations of the Turing jump. This is the subject of the next section.

### 4.3. Forcing question

Let us try to abstract a little from what has been done so far, in order to study the notion of forcing question, independently of the forcing we are working with.

**Definition 4.16.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing. A *forcing question* is a relation  $?\vdash$  between conditions and requirements, such that for any condition  $c \in \mathbb{P}$  and any arithmetic requirement  $\mathcal{R}$

- (1) If  $c ?\vdash \mathcal{R}$ , then there exists an extension  $d \leq c$  such that  $d$  forces  $\mathcal{R}$
- (2)  $c ?\vdash \mathcal{R}$  or  $c ?\vdash \neg \mathcal{R}$ . ◇

It follows from (1) and (2) that if  $c ?\not\vdash \mathcal{R}$ , then there exists an extension  $d \leq c$  such that  $d$  forces  $\neg \mathcal{R}$ . Any forcing relation  $\Vdash$  induces a forcing question  $?\vdash$  by defining  $c ?\vdash \mathcal{R}$  for a  $\Sigma_n^0$  requirement  $\mathcal{R}$  if there exists an extension  $d \leq c$  such that  $d \Vdash \mathcal{R}$  and  $c ?\vdash \mathcal{R}$  for a  $\Pi_n^0$  requirement if  $c ?\not\vdash \neg \mathcal{R}$ .

If the forcing notion is computable, as is the case for Cohen forcing, the complexity of the forcing question for  $\Sigma_n^0$  requirements inherits from the complexity of the forcing relation for the same requirements.

**Exercise 4.17.** Let  $(\mathbb{P}, \leq)$  be a Cantor forcing, and  $\Vdash$  a forcing relation. Show that the relation defined by  $c ?\vdash \mathcal{R}$  if there exists an extension  $d \leq c$  such that  $d \Vdash \mathcal{R}$  is a forcing question.  $\diamond$

### 4.3.1. Preservation of the arithmetic hierarchy

A certain number of weakness properties of sufficiently generic sets for Cantor forcings depend on the existence of a forcing question with good definitional properties. In what follows, we fix a Cantor forcing  $(\mathbb{P}, \leq)$  as well as a forcing question  $?\vdash$ . The following Proposition 4.19 is the first example, and generalizes Theorem 10-5.7.

**Definition 4.18.** A forcing question  $?\vdash$  *preserves the arithmetical hierarchy* if for any condition  $c$  and any  $\Sigma_n^0$  requirement  $\mathcal{R}$ , the relation  $c ?\vdash \mathcal{R}$  is  $\Sigma_n^0$  uniformly in  $\mathcal{R}$ .  $\diamond$

**Proposition 4.19.** Let  $?\vdash$  be a forcing question which preserves the arithmetic hierarchy. Then, for any  $n \geq 1$ , for any set  $A$  which is not  $\Sigma_n^0$ , and for any sufficiently generic set  $G$ ,  $A$  is not  $\Sigma_n^0(G)$ .  $\star$

PROOF. For all  $e \in \mathbb{N}$ , let

$$D_e = \{c \in \mathbb{P} : (\exists m \notin A \ c \Vdash m \in W_e^{G^{(n-1)}}) \text{ or } (\exists m \in A \ c \Vdash m \notin W_e^{G^{(n-1)}})\}$$

Let us show that  $D_e$  is a dense set in  $(\mathbb{P}, \leq)$ . Let  $c \in \mathbb{P}$  and let

$$U = \{m \in \mathbb{N} : c ?\vdash m \in W_e^{G^{(n-1)}}\}$$

According to Lemma 10-5.6, the requirement  $m \in W_e^{G^{(n-1)}}$  is  $\Sigma_n^0$  uniformly in  $m$ , so since the forcing question preserves the arithmetic hierarchy, the set  $U$  is  $\Sigma_n^0$ . The set  $A$  not being  $\Sigma_n^0$ , then the symmetric difference  $U \Delta A = (U \setminus A) \cup (A \setminus U)$  is not empty. Let  $m \in U \Delta A$ .

Case 1:  $m \in U \setminus A$ . Then, by definition of the forcing question, there exists an extension  $d \leq c$  such that  $d \Vdash m \in W_e^{G^{(n-1)}}$ . In particular,  $d \in D_e$ .

Case 2:  $m \in A \setminus U$ . Then, still by the definition of the forcing question, there exists an extension  $d \leq c$  such that  $d \Vdash m \notin W_e^{G^{(n-1)}}$ . Again,  $d \in D_e$ .

In all cases, there is an extension of  $d$  in  $D_e$ , so the set  $D_e$  is dense. Let  $F$  be a sufficiently generic filter for  $(\mathbb{P}, \leq)$ . By density of  $D_e$ , we can assume that  $F$  intersects  $D_e$  for all  $e \in \mathbb{N}$ . Let  $G = \dot{F}$ . By definition of the forcing relation, for all  $e \in \mathbb{N}$ , either  $m \in W_e^{G^{(n-1)}}$  for an  $m \notin A$ , or  $m \notin W_e^{G^{(n-1)}}$  for an  $m \in A$ , so  $A$  is not  $\Sigma_n^0(G)$ .  $\blacksquare$

**Corollary 4.20**

Let  $? \vdash$  be a forcing question which preserves the arithmetic hierarchy. Then, for any  $n \geq 0$  and any set  $A$  not  $\emptyset^{(n)}$ -computable, for any sufficiently generic set  $G$ ,  $A$  is not  $G^{(n)}$ -computable.

PROOF. As  $A$  is not  $\emptyset^{(n)}$ -computable, it is not  $\Delta_{n+1}^0$ . Either  $A$  is not  $\Sigma_{n+1}^0$ , or  $\bar{A}$  is not  $\Sigma_{n+1}^0$ . By Proposition 4.19, for any set  $G$  sufficiently generic for  $(\mathbb{P}, \leq)$ ,  $A$  is not  $\Sigma_{n+1}^0(G)$  in the first case, and  $\bar{A}$  is not  $\Sigma_{n+1}^0(G)$  in the second case. In any case,  $A$  is not  $\Delta_{n+1}^0(G)$  and therefore not  $G^{(n)}$ -computable. ■

**Corollary 4.21**

Let  $n \geq 0$  and  $A \in 2^{\mathbb{N}}$  be a non  $\emptyset^{(n)}$ -computable set. If  $G$  is sufficiently generic for Jockusch-Soare forcing or for Sacks forcing,  $A$  is not  $G^{(n)}$ -computable.

PROOF. According to Proposition 4.14, these two forcing notions preserve the arithmetic hierarchy. ■

**4.3.2. Preservation of hyperimmunity**

Recall that a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is hyperimmune relative to  $X$  if it is not dominated by any  $X$ -computable function (see Section 7-4). No function is hyperimmune relative to all Turing degrees, starting with the Turing degree of the function itself. However, if a function is hyperimmune, it is also hyperimmune relative to any computably dominated degree. We can therefore consider that the computably dominated degrees “preserve” the hyperimmunities of all functions simultaneously. We will now study to what extent sufficiently generic sets for Cantor forcings preserve hyperimmunities.

We saw in Section 10-3.1 that any weakly 1-generic set  $X$  was of hyperimmune degree. More precisely, the principal function  $p_X$  of  $X$  which to  $n$  associates the  $n$ -th element of  $X$  is hyperimmune. The sets which are sufficiently generic for Cohen forcing therefore do not preserve all the hyperimmunities simultaneously. However, it is possible to ensure that they preserve hyperimmunity from any fixed hyperimmune function.

**Definition 4.22.** A forcing question  $? \vdash$  is *compact* if for any  $c \in \mathbb{P}$ , any arithmetic requirement  $\mathcal{R}(x)$ , if  $c ? \vdash \exists x \mathcal{R}(x)$ , then there exists a finite set  $U \subseteq \mathbb{N}$  such that  $c ? \vdash \exists x \in U \mathcal{R}(x)$ . ◇

The compactness of a forcing question is sufficient to ensure preservation of hyperimmunity:

**Proposition 4.23.** Let  $? \vdash$  be a compact forcing question which preserves the arithmetic hierarchy. Then, for any  $n \geq 0$  and any hyperimmune function  $f : \mathbb{N} \rightarrow \mathbb{N}$  relative to  $\emptyset^{(n)}$ , for any set  $G$  sufficiently generic for  $(\mathbb{P}, \leq)$ ,  $f$  is hyperimmune relative to  $G^{(n)}$ . ★

PROOF. Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be a hyperimmune function relative to  $\emptyset^{(n)}$ . For all  $e \in \mathbb{N}$ , let

$$D_e = \{c \in \mathbb{P} : (\exists m \ c \Vdash \Phi_e(G^{(n)}, m) \uparrow) \text{ or } (\exists m \ c \Vdash \Phi_e(G^{(n)}, m) \downarrow < f(m))\}$$

Let us show that  $D_e$  is a dense set in  $(\mathbb{P}, \leq)$ . Let  $c \in \mathbb{P}$  and let  $g : \mathbb{N} \rightarrow \mathbb{N}$ , the partial function which for all  $m$ , searches for a finite set  $U \subseteq \mathbb{N}$  such that  $c \Vdash \Phi_e(G^{(n)}, m) \downarrow \in U$ . If such a set  $U$  is found,  $g(m) = 1 + \max U$ , otherwise  $g(m)$  is not defined. Knowing that the forcing question preserves the arithmetic hierarchy,  $g$  is partial  $\emptyset^{(n)}$ -computable. Two cases arise:

Case 1: There is an  $m$  such that  $g(m)$  is not defined. Then, by compactness of the forcing question,  $c \nVdash \Phi_e(G^{(n)}, m) \downarrow$ , so there exists a  $d \leq c$  such that  $d \Vdash \Phi_e(G^{(n)}, m) \uparrow$ . In particular,  $d \in D_e$ .

Case 2: The function  $g$  is total  $\emptyset^{(n)}$ -computable. By hyperimmunity of  $f$  relative to  $\emptyset^{(n)}$ , there is an  $m \in \mathbb{N}$  such that  $g(m) \leq f(m)$ . In particular,  $c \Vdash \Phi_e(G^{(n)}, m) \downarrow \in U$  for a finite set  $U$  such that  $\max U < f(m)$ . By definition of a forcing question, there is an extension  $d \leq c$  such that  $d \Vdash \Phi_e(G^{(n)}, m) \downarrow \in U$ , therefore  $\Phi_e(G^{(n)}, m) \downarrow < f(m)$ . It follows that  $d \in D_e$ .

In all cases, there is an extension of  $d$  in  $D_e$ , so the set  $D_e$  is dense. Let  $F$  be a sufficiently generic filter for  $(\mathbb{P}, \leq)$ . By density of  $D_e$ , we can assume that  $F$  intersects  $D_e$  for all  $e \in \mathbb{N}$ . Let  $G = \dot{F}$ . By definition of the forcing relation, for all  $e \in \mathbb{N}$ , either  $\Phi_e(G^{(n)})$  is a partial function, or  $\Phi_e(G^{(n)}, m) \downarrow < f(m)$  for an  $m \in \mathbb{N}$ , therefore  $f$  is hyperimmune relative to  $G^{(n)}$ . ■

The canonical forcing question of Cohen forcing is compact and preserves the arithmetic hierarchy, which implies that any set sufficiently generic for Cohen forcing preserves the hyperimmunity of any previously fixed function. It can be shown that the same goes for the forcings of Jockusch-Soare and Sacks.

#### 4.3.3. Preservation of non PA degrees

We will end the study of the properties of the forcing questions with a criterion for not computing a PA degree.

**Definition 4.24.** A forcing question  $? \vdash$  is  $\Pi$ -merging if for any  $c \in \mathbb{P}$ , any pair of  $\Pi_n^0$  requirements  $\mathcal{R}_0, \mathcal{R}_1$  such that  $c ? \vdash \mathcal{R}_0$  and  $c ? \vdash \mathcal{R}_1$ , there is an extension  $d \leq c$  which simultaneously forces  $\mathcal{R}_0$  and  $\mathcal{R}_1$ .  $\diamond$

**Proposition 4.25.** Let  $? \vdash$  be a  $\Pi$ -merging forcing question which preserves the arithmetic hierarchy. Then, for any  $n \geq 0$  for any set  $G$  sufficiently generic for  $(\mathbb{P}, \leq)$ ,  $G^{(n)}$  is not of PA degree relative to  $\emptyset^{(n)}$ .  $\star$

PROOF. According to Theorem 8-6.2, a Turing degree is PA iff it computes a  $\{0,1\}$ -valued DNC function. In what follows, we will assume that  $\Phi_0, \Phi_1, \dots$  is an enumeration of all  $\{0,1\}$ -valued Turing functionals. For all  $e \in \mathbb{N}$ , let

$$D_e = \left\{ c \in \mathbb{P} : \begin{array}{ll} (\exists m \ c \Vdash \Phi_e(G^{(n)}, m) \uparrow) & \\ \text{or} & (\exists m \ c \Vdash \Phi_e(G^{(n)}, m) \downarrow = \Phi_e(\emptyset^{(n)}, m)) \end{array} \right\}$$

Let us show that  $D_e$  is a dense set in  $(\mathbb{P}, \leq)$ . Let  $c \in \mathbb{P}$  and let  $g : \mathbb{N} \rightarrow \mathbb{N}$ , the partial function which for any  $m$ , searches for an integer  $v \in \{0,1\}$  such that  $c ? \vdash \Phi_e(G^{(n)}, m) \downarrow = v$ . If such a  $v$  is found,  $g(m) = v$ , otherwise  $g(m)$  is not defined. Knowing that the forcing question preserves the arithmetic hierarchy,  $g$  is partial  $\emptyset^{(n)}$ -computable. Two cases arise:

Case 1: There is an  $m$  such that  $g(m)$  is not defined. Then, for all  $v \in \{0,1\}$ ,  $c \not \vdash \Phi_e(G^{(n)}, m) \downarrow = v$ , in other words  $c ? \vdash \neg(\Phi_e(G^{(n)}, m) \downarrow = v)$ . As the relation is  $\Pi$ -mergeable, there exists a  $d \leq c$  such that  $d$  force at the same time  $\neg(\Phi_e(G^{(n)}, m) \downarrow = 0)$  and  $\neg(\Phi_e(G^{(n)}, m) \downarrow = 1)$ , however the functional being  $\{0,1\}$ -valued,  $d$  therefore forces  $\Phi_e(G^{(n)}, m) \uparrow$ . In particular,  $d \in D_e$ .

Case 2: The function  $g$  is total  $\emptyset^{(n)}$ -computable. Knowing that no degree is PA relative to itself, there exists an integer  $m \in \mathbb{N}$  such that  $g(m) = \Phi_m(\emptyset^{(n)}, m)$ . In particular,  $c ? \vdash \Phi_e(G^{(n)}, m) \downarrow = g(m)$ , so by definition of a forcing question, there exists an extension  $d \leq c$  such that  $d \Vdash \Phi_e(G^{(n)}, m) \downarrow = g(m) = \Phi_m(\emptyset^{(n)}, m)$ . It follows that  $d \in D_e$ .

In all cases, there is an extension of  $d$  in  $D_e$ , so the set  $D_e$  is dense. Let  $F$  be a sufficiently generic filter for  $(\mathbb{P}, \leq)$ . By density of  $D_e$ , we can assume that  $F$  intersects  $D_e$  for all  $e \in \mathbb{N}$ . Let  $G = \dot{F}$ . By definition of the forcing relation, for any  $e \in \mathbb{N}$ , either  $m \mapsto \Phi_e(G^{(n)}, m)$  is a partial function, or  $\Phi_e(G^{(n)}, m) \downarrow = \Phi_m(\emptyset^{(n)}, m)$  for an  $m \in \mathbb{N}$ , therefore  $G^{(n)}$  is not of PA degree relative to  $\emptyset^{(n)}$ .  $\blacksquare$

The forcing question for Cohen forcing is  $\Pi$ -merging, just like the forcing question for computable Sacks forcing. On the other hand, there is no  $\Pi$ -

merging forcing question for Jockusch-Soare forcing, because there exist non-empty  $\Pi_1^0$  classes containing only sets of PA degree.

**Exercise 4.26.** A forcing question  $? \vdash$  is  $\Pi$ - $\omega$ -merging if for any  $c \in \mathbb{P}$ , any sequence of  $\Pi_n^0$  requirements  $\mathcal{R}_0, \mathcal{R}_1, \dots$  such that  $c ? \vdash \mathcal{R}_i$  for all  $i \in \mathbb{N}$ , there exists an extension  $d \leq c$  which simultaneously forces  $\mathcal{R}_i$  for all  $i$ . Show that if  $? \vdash$  is a  $\Pi$ - $\omega$ -merging forcing question which preserves the arithmetic hierarchy, then for any set  $G$  sufficiently generic and all  $n$ , its iterated jump  $G^{(n)}$  is not of DNC degree relative to  $\emptyset^{(n)}$ .  $\diamond$

# Chapter 12

## Quest for natural degrees

The beginnings of Computability Theory enabled to observe that all computable enumerable sets originating from natural problems were either computable or as powerful as the halting problem, moreover via a many-one reduction (see Section 5-4). This led Post to ask the following question in 1944:

**Question (Post’s problem [180]).** Are there non-computable c.e. degrees which are strictly weaker than the halting problem? ★

Post’s problem remained open for almost a decade, before being solved in the affirmative by Muchnik [162] and Friedberg [62] via the priority method, which we will see in Section 13-3. Post’s problem has since seen many other different resolutions, not necessarily using the priority method. We can cite for example the construction of a non-computable K-trivial c.e. set that we will see with Theorem 16-4.5. However, all these constructions are based on a complex argument allowing the ad-hoc construction of an “artificial” set having the desired properties, and the only undecidable “natural” decision problems known to date are reduced to the halting problem<sup>1</sup>. Conversely, the halting problem seems to arise naturally all over the place. The question then arose of the properties that give it this naturalness.

---

<sup>1</sup>This statement should be taken with caution, however, and will be attenuated in the last section of this chapter.

## 1. Three emblematic undecidable problems

Before tackling Post's problem directly, let us see three emblematic examples of undecidable c.e. decision problems, all many-one equivalent to the halting problem.

### Post correspondence problem

We start with a problem defined by Post himself in 1946, which is called the Post correspondence problem, and which should not be confused with "Post's problem" which refers to the above question.

Given two finite lists of strings  $\sigma_0, \dots, \sigma_n \in 2^{<\mathbb{N}}$  and  $\tau_0, \dots, \tau_n \in 2^{<\mathbb{N}}$ , is there a sequence of indices  $(i_k)_{k \leq K}$  — possibly with repetition — such that the concatenations  $\sigma_{i_0} \sigma_{i_1} \dots \sigma_{i_K}$  and  $\tau_{i_0} \tau_{i_1} \dots \tau_{i_K}$  form the same string?

The question may seem simple on the surface, and one might even think at first that it is easy to create an algorithm to solve it. After all, the number of strings involved is finite. On further reflection, the problem shouldn't appear so obvious, and for good reason: although it may seem surprising, it is an undecidable question, and as difficult as whether a computer program halts or not.

To show it, Post finds a nifty way to encode the computation of a Turing machine via instances of the correspondence problem. Thus, an instance for which a correspondence exists will correspond to a computation which halts, and an instance for which no correspondence exists will correspond to an infinite computation.

### Domino problem

In 1961, Hao Wang imagines the following problem: given a finite set of *tiles*, that is to say of squares having a color on each of their sides, the objective is to make a tiling of the plan using only tiles in our finite set, requiring that two neighbour tiles share the same color on their common side. There are of course sets of tiles for which such a tiling of the plane is possible, and others for which it is not.

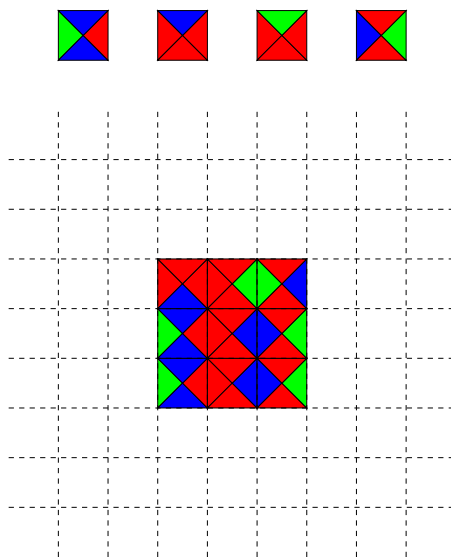


Figure 1.1: Start of a tiling of the plane using the four tiles above.

A lemma deriving from that of König applies to tilings of the plane using a finite number of tiles: if for all  $n$  there exists a tiling of  $n \times n$  tiles, then there exists an infinite tiling of the plane. We then deduce that if a finite set of tiles does not allow the plane to be paved, there exists  $n$  such that no tiling of size  $n \times n$  is possible. If a tiling is impossible, it suffices to look for the smallest  $n$  such that no tiling configuration of size  $n \times n$  will work. In other words, the finite sets of tiles which do not allow the plane to be paved can be enumerated by an algorithm.

Wang then conjectures that the same is true for finite sets of tiles making it possible to pave the plane, which would make the problem of tiling the plane decidable: there would exist an algorithm making it possible to decide, given a finite set of tiles if the latter may or may not pave the plane.

But in 1966, Robert Berger (a student of Wang) shows that the problem of paving the plane is not decidable, by reducing to it again the halting problem for Turing machines, in the manner of Post: Berger creates a paving system allowing to “simulate” the computation being carried out on a Turing machine, an impossible paving meaning that the machine halts, and a possible paving meaning that the computation continues indefinitely.

### Tenth Hilbert problem

A Diophantine equation is a polynomial equation with one or more unknowns, whose solutions are sought among integers — or possibly rational

— the coefficients being themselves also integers. For example  $a^2 + b^2 = c^2$  is a Diophantine equation having many solutions like  $a = 3, b = 4$  and  $c = 5$ . On the other hand, there are Diophantine equations having no solution. Some of them have asked for their resolution — to find the solutions or show that they do not have any — considerable efforts from many mathematicians over several centuries. The example par excellence is certainly the famous “Fermat’s last theorem” which states that for any integer  $n > 2$ , the equation  $a^n + b^n = c^n$  has no integer solutions.

Fermat states his theorem in the margin of a translation of “Arithmetic of Diophantus”, in which he writes: *“On the contrary, it is impossible to divide either a cube into two cubes, or a Quadruple in two quadruple, that is in general any power greater than the square in two powers of the same degree: I have discovered a truly marvelous demonstration of this, which this margin is too narrow to contain”*.

Many mathematicians have sought this wonderful demonstration for centuries without success. It is only after 357 years of efforts that the mathematician Andrew Wiles, helped by his student Richard Taylor, will provide a proof using mathematical tools obviously much more complex than those which existed at the time of Fermat.

In 1900, Hilbert places the question of solving Diophantine equations in tenth position in his famous list of 23 problems: *“Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.”*

Hilbert asks, before we have a formal definition, the existence of an *algorithm* allowing to know if any Diophantine equation admits a solution or not.

Here again, the set of Diophantine equations having a solution is computably enumerable; it suffices to search among all the potential candidates if one of them is a solution. Martin Davis, Hilary Putnam and Julia Robinson had the idea of showing that Hilbert’s tenth problem is undecidable by following a daring intuition, and which turns out to be correct: a Diophantine equation is the building block of a formula of arithmetic, in this case the equality between two terms. Gödel showed that computably enumerable sets are exactly those which can be defined by  $\Sigma_1^0$  formulas of arithmetic.

Would it not be possible to transform such a formula into an equivalent  $\Sigma_1^0$  formula  $\exists x_1 \dots \exists x_n F(x_1, \dots, x_n)$ , but where  $F$  is no more than a big Diophantine equation?

Consider for example the case of the formula

$$t_1(a_1, \dots, a_i) = q_1(b_1, \dots, b_j) \vee t_2(x_1, \dots, x_k) = q_2(y_1, \dots, y_l),$$

where  $t_1, t_2$  and  $q_1, q_2$  are terms. Then, this formula is true in  $\mathbb{N}$  iff the formula

$$t_1(a_1, \dots, a_i) - q_1(b_1, \dots, b_j) = 0 \vee t_2(x_1, \dots, x_k) - q_2(y_1, \dots, y_l) = 0$$

is true in  $\mathbb{Z}$ , or equivalently if the Diophantine equation

$$(t_1(a_1, \dots, a_i) - q_1(b_1, \dots, b_j)) \times (t_2(x_1, \dots, x_k) - q_2(y_1, \dots, y_l)) = 0$$

admits solutions. We easily show something similar for the connector  $\wedge$ , the remaining difficulty being in the deletion of existential and bounded universal quantifications. Davis, Putnam, and Robinson succeeded in removing bounded quantifications at the cost of using the exponential function in the resulting equations. The work will then be completed by Matiassevitch, who succeeded in encoding the exponential function in Diophantine equations, leading to the following theorem.

**Theorem 1.2 (MRDP theorem)**

*Let  $A \subseteq \mathbb{N}$  be a computably enumerable set. Then, there is a  $\Sigma_1^0$  formula of arithmetic  $F(x) = \exists y_1, \dots, \exists y_n G(x, y_1, \dots, y_n)$  where  $G$  has no quantifier, such that  $x \in A$  iff  $\mathbb{N} \models F(x)$ .*

The MRDP theorem is remarkable in that it illustrates the fact that the undecidability of Peano arithmetic is concealed in the very structure of addition and multiplication of integers, without any need to resort to bounded quantifications. It provides of course an answer to the tenth problem of Hilbert: if an algorithm makes it possible to know if a Diophantine equation has integer solutions, then one can create an algorithm computing the halting problem.

## 2. Natural Turing degrees

Let's go back to our original question: what is special about the halting problem, so that all natural c.e. decision problems are equivalent to it? What makes a Turing degree natural? Steel [216] suggests an answer: a natural Turing degree should be definable, and its definition should be relativizable to any degree. For example, the halting problem  $\emptyset'$ , initially defined as a particular set, namely  $\{e : \Phi_e(e) \downarrow\}$ , is relativized to any set  $X$ , by considering the set  $X' = \{e : \Phi_e(X, e) \downarrow\}$ . As we saw in Section 4-6, this is a notion on Turing degrees, in the sense that if  $X \equiv_T Y$ , then  $X' \equiv_T Y'$ .

### 2.1. Sacks question

Steel's idea echoes an old question from Sacks [187]: is there a solution to Post's problem that is invariant on Turing degrees? We will say that  $W$  is a *c.e. operator* if  $W$  corresponds to a Turing functional which, uniformly in a set  $X$ , enumerates a set which we will denote by  $W^X$ . Sacks asks if there is a c.e. operator  $W$  such that  $X <_T W^X <_T X'$  for all  $X$  and such that  $X_0 \equiv_T X_1$  implies  $W^{X_0} \equiv_T W^{X_1}$  for all  $X_0, X_1$ .

By working on it a little, the reader will be able to see that the priority method used in Theorem 13-3.1 to construct a c.e. set  $Y$  such that  $0 <_T Y <_T \emptyset'$ , can be relativized to any oracle  $X$  to obtain a set  $X$ -c.e.  $Y$  such that  $X <_T Y <_T X'$ . On the other hand, it is much more uncertain that this relativization is invariant in the Turing degrees, and in fact, it is not. The first result in this direction was obtained by Lachlan, who gave a negative answer to Sacks' question, in the particular case where invariance is expected to be uniform, i.e., that we ask for the existence of functions  $h_1, h_2$  such that if  $\Phi_{a_1}(X_1) = X_2$  and  $\Phi_{a_2}(X_2) = X_1$ , then  $\Phi_{h_1(a_1)}(W^{X_1}) = W^{X_2}$  and  $\Phi_{h_2(a_2)}(W^{X_2}) = W^{X_1}$ . Note that Lachlan does not require that the functions  $h_1, h_2$  are computable, but simply that they exist.

In the case where the operator is invariant in the Turing degrees, the notation  $W(\mathbf{a})$  for a Turing degree  $\mathbf{a}$  has a meaning: it is the Turing degree obtained by applying to  $W$  any set in the degree  $\mathbf{a}$ . Lachlan actually shows the following: for any c.e. operator uniformly invariant such that  $W(\mathbf{d}) \geq \mathbf{d}$  for any degree  $\mathbf{d}$ , then there exists a degree  $\mathbf{a}$  such that for any degree  $\mathbf{d} \geq \mathbf{a}$  we have  $W(\mathbf{d}) = \mathbf{d}$ , or then such that for any degree  $\mathbf{d} \geq \mathbf{a}$  we have  $W(\mathbf{d}) = \mathbf{d}'$ . In the first case we will say that  $W$  coincides with the identity *on a cone*, and in the second that  $W$  coincides with the Turing jump *on a cone*. The expression *on a cone* then signifying on a cone in the Turing degrees, that is to say on all the degrees greater than  $\mathbf{a}$  for a certain  $\mathbf{a}$ , the degree  $\mathbf{a}$  being *the base of the cone*. Lachlan therefore obtains the following result.

**Theorem 2.1 (Lachlan [131])**

*Let  $W$  be a c.e. operator uniformly invariant such that  $W(\mathbf{d}) \geq \mathbf{d}$  for any degree  $\mathbf{d}$ . Then,  $W$  is the jump operator on a cone or  $W$  is the identity operator on a cone.*

### 2.2. Sacks question for any degree

We have so far talked about natural c.e. degrees, arguing that  $\mathbf{0}$  and  $\mathbf{0}'$  are the only ones. There are, however, many natural decision problems that are strictly more powerful than the halting problem, and which are of course not c.e. We can then push the question of naturality to any degree. Let's see some canonical examples first:

1. P: The problem of determining if a polynomial of  $\mathbb{Z}[x, y_0, y_1, y_2, \dots]$  has solutions for any sufficiently large element  $x$ .
2. T: The problem of knowing if a statement of arithmetic is true in  $\mathbb{N}$ .
3. WF: The problem of knowing if a computable tree of  $\mathbb{N}^{<\mathbb{N}}$  has infinite paths.

The problem P is  $\Sigma_2^0$  and  $\emptyset'' \leq_m P$ . The problem T is such that  $\emptyset^{(n)} \leq_m T$  for all  $n \in \mathbb{N}$  uniformly in  $n$ . It is moreover many-one computable with the oracle  $\emptyset^{(\omega)} = \oplus_n \emptyset^{(n)}$  and therefore corresponds to the first transfinite level in the hierarchy of Turing jumps (we will see this formally in Part IV). The WF problem is even more complex, and corresponds to a higher transfinite Turing jump that can be noted  $\emptyset^{(\omega_1^{ck})}$  and which is commonly called *hyperjump* (we will also see this formally in Part IV).

Note that the iterations of the Turing jump, just like the simple Turing jump, are also relativized to any oracle invariantly on Turing degrees: for example if  $X \equiv_T Y$  then  $X^{(3)} \equiv_T Y^{(3)}$ . The iterated jump operators are therefore also natural, and one can find for each of them decision problems which correspond to them. At the same time, we can iterate Sacks' question: the solutions to Post's problem are not the only constructions of exotic degrees in Computability Theory, and this question of the invariance of the operators can be extended to any construction. Consider for example the construction of a computably dominated set of Theorem 7-5.6 or that of Theorem 8-4.5. In both cases, the construction requires  $\emptyset''$ . One verifies easily with one or the other construction that one can create a  $\emptyset''$ -computable operator  $W$  such that for all  $X$  the set  $W^X$  verifies  $X <_T W^X$  and is such that  $W^X$  is computably dominated relative to  $X$ . Can such an operator be invariant in the Turing degrees? If this is not the case, can we get one that can be computed in  $\emptyset'''$  or in a more powerful oracle?

### 2.3. Martin's conjecture

Inspired by these questions, and perhaps by Lachlan's result on the c.e. operators, Martin then proposes a rather daring conjecture which essentially says: the jump operator and its iterations, are the only definable and invariant operators in Turing degrees. More formally, the conjecture has two distinct parts:

**Conjecture 2.2 (Martin [1] p. 281).** Let  $f : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  be a Borel function invariant in the Turing degrees. Then, we can see  $f$  as a function on the Turing degrees, in which case we have:

- I. Either  $f$  is constant on a cone, or  $f$  is increasing on a cone.

- II. If  $f$  is strictly increasing on a cone, then it corresponds to the Turing jump or to one of its iterations (possibly transfinite). ★

What do we mean by Borel function? Before answering it, let us note that if we go completely out of the framework of Computability Theory, we can perfectly define functions  $f$  invariant in Turing degrees such that  $\mathbf{a} < f(\mathbf{a}) < \mathbf{a}'$  for all  $\mathbf{a}$ , simply using the axiom of choice in Set Theory. The objective of restricting the conjecture to Borel functions  $f$  essentially aims to prohibit the use of this axiom, and the conjecture is generally presented without the restriction for  $f$  to be Borel, and with additional assumptions on the axioms of Set Theory, essentially meaning: “for any function  $f$  that can be defined without using the axiom of choice”.

Martin’s conjecture essentially tells us that the only natural and non-constant operators in the Turing degrees are the identity operator, the jump operator and its iterations. If this conjecture is still open to this day, much progress has been made, by adding to it, as Lachlan did, the condition of uniformity in the invariance of the functions  $f$  considered. First, Steel [216] showed II of Martin’s conjecture, for uniformly invariant functions, thus generalising the Lachlan result considerably. Then, still for uniformly invariant functions, Slaman and Steel [204] have shown I, and also managed to get rid, via a very clever proof, of uniformity for the case of functions  $f$  such that  $f(\mathbf{a}) < \mathbf{a}$  for any degree  $\mathbf{a}$  on a cone: these functions are necessarily constant on a cone.

To date, not much is known about the conjecture in the general case, and it is also also valid for the potentially much simpler question of Sacks, which also remains open in the case of non-uniformity. This story of non-uniformity has however something to instill doubt: if we basically seek to construct functions invariant in the Turing degrees, we always obtain a uniform invariance. This finding led Steel to make the following conjecture:

**Conjecture 2.3 (Steel [216]).** If a Borel function is invariant in the Turing degrees, then it is uniformly invariant. ★

If Steel’s conjecture is true, then it will show Martin’s conjecture.

### 3. Mass problems

Faced with Post’s problem, we saw a first negative response, namely that under naturality assumptions, there are no c.e. sets of intermediary degree. There is another approach, complementary to the first, which consists in saying that if the iterations of the Turing jump are the only natural degrees, it is because of the too restrictive nature of a Turing degree: there are

computational powers which do not correspond to Turing degrees taken individually, but to classes of degrees.

Consider for example the completions of Peano arithmetic. We have seen with Theorem 9-3.10 a proof that any complete and consistent theory which extends Peano arithmetic is non-computable, but we did not do it by showing that the halting problem could be reduced to such a theory, and for good reason: it is not always the case. There exist  $\Delta_2^0$  PA degrees not computing the halting problem. This is not incompatible with Martin's conjecture, in the sense that if one seeks to define a very specific natural extension of Peano arithmetic which is complete and consistent, one will find one which computes the halting problem or more. This is the case for example of the set of true formulas in  $\mathbb{N}$ .

The PA degrees form a *natural* computational power as a class: for any computable infinite binary tree, there exists a procedure which takes a completion of Peano arithmetic as input, and computes a path of the tree in return. Here we go beyond the naturality of the degrees, to consider instead the naturality of the classes of degrees. As we have seen in this book, there is a wide variety of classes of degrees, all defined in a very natural way, and which do not correspond to iterations of the halting set: the high, hyperimmune, PA degrees ... The notion of class of degrees is generally approached via the principle of *mass problems*. These go back to Kolmogorov [118], who speaks informally of them as a formalization of Brouwer's principles of intuitionist logic, before being rigorously defined by Medvedev [153] and Muchnik [164]:

**Definition 3.1.** A *mass problem*  $\mathcal{P} \subseteq 2^{\mathbb{N}}$  is seen as the set of its possible solutions, identified, via an appropriate encoding, to elements of  $2^{\mathbb{N}}$ .  $\diamond$

For example, a problem will be solvable if it contains a computable element. Problems are studied in particular through the balance of power they maintain with one another. Muchnik suggests the following approach:

**Definition 3.2 (Muchnik reduction).** A mass problem  $\mathcal{P}$  is *Muchnik reducible* to a mass problem  $\mathcal{Q}$ , in which case one notes  $\mathcal{P} \leq_w \mathcal{Q}$  if any solution to  $\mathcal{Q}$  allows to compute a solution to  $\mathcal{P}$ .  $\diamond$

For example, any non-empty  $\Pi_1^0$  class reduces in Muchnik's sense to the problem consisting of complete and consistent extensions of PA. Medvedev proposes a more restrictive definition asking for uniformity:

**Definition 3.3 (Medvedev reduction).** A mass problem  $\mathcal{P}$  is *Medvedev*

*reducible* to mass problem  $\mathcal{Q}$ , in which case write  $\mathcal{P} \leq_s \mathcal{Q}$  if there exists a functional  $\Phi$  such that  $\Phi(X) \in \mathcal{P}$  for all  $X \in \mathcal{Q}$ .  $\diamond$

Any non-empty  $\Pi_1^0$  class is also reduced in Medvedev's sense to the class of PA sets. The two concepts do not, however, coincide in the general case. Jockusch [104] for example showed that the class of  $\text{DNC}_2$  functions was reduced in the sense of Muchnik to that of  $\text{DNC}_3$  functions, but not in the sense of Medvedev.

The equivalence classes of the relations  $\equiv_w$  and  $\equiv_s$  (whose definitions result from  $\leq_w$  and  $\leq_s$ ) are respectively called *Muchnik degrees* and *Medvedev degrees*, the structure of which has been extensively studied. There is a natural embedding of the Turing degrees towards the Muchnik and Medvedev degrees, by associating with a Turing degree  $\deg_T(X)$  the Muchnik or Medvedev degree of the problem  $\{X\}$ . This embedding respects the semi-lattice structure. We can therefore consider the Muchnik and Medvedev degrees as an extension of the Turing degrees. In particular, we can define  $\mathbf{0}_w$  and  $\mathbf{0}'_w$ , the Muchnik degrees of  $\{\emptyset\}$  and  $\{\emptyset'\}$ , respectively. So the following proposition is in some way related to Post's original question.

**Proposition 3.4.** Let PA be the Muchnik degree of the PA degrees. Then

$$\mathbf{0}_w <_w \text{PA} <_w \mathbf{0}'_w$$

This generalization of the Turing degrees comes at a cost: the Muchnik and Medvedev degrees are much more numerous. More precisely, the Turing degrees have the power of the continuum ( $|\mathbb{N}|$ ) while the Muchnik and Medvedev degrees have cardinality  $|2^{\mathbb{N}}|$  and have anti-chains of this size.

# Chapter 13

## Priority method and c.e. degrees

Among the non-computable sets, the computably enumerable sets play a particularly important role. These sets are “almost computable”, in the sense that if an integer  $n$  belongs to a c.e. set  $A$ , then this information will be known in a finite time. The class of computably enumerable sets is quite natural, for several reasons.

First, the c.e. sets have several very different characterizations, which makes it a relatively *robust* class. By definition, a set is c.e. if it is the domain of a partial computable function. The non-empty c.e. sets are also precisely those which are the image of a total computable function, the function being able to be injective if the set is infinite (see Proposition 3-7.2). Equivalently, a set is c.e. if and only if it is reducible to the halting problem by a many-one reduction, or if it is  $\Sigma_1^0$ .

Computably enumerable sets form a *syntactic class* unlike computable sets. Indeed, the c.e. sets are precisely the  $\Sigma_1^0$  sets, while the computable sets are the sets definable by both a  $\Sigma_1^0$  and  $\Pi_1^0$  predicate. This syntactic nature gives the c.e. sets better uniformity properties. Thus, the class of computably enumerable sets is uniformly c.e., because it suffices to list all the partial computable functions, or in an equivalent way all the  $\Sigma_1^0$  formulas. The computable sets cannot, on the other hand, be listed in a computable way (according to Theorem 7-6.2, the high degrees are exactly those allowing to list the computable sets).

Finally, and this is perhaps one of the most important arguments in favor of the naturality of c.e. sets, a number of non-decidable problems in mathematics happen to be computably enumerable. Among them, we will of course cite the halting problem, but also the set of theorems of arithmetic (see Theorem 9-3.7), or even the set of solutions of Diophantine equations (see, for this purpose, Theorem 12-1.2).

## 1. C.e. degrees

Being computably enumerable is a property of a set and not of a Turing degree. Indeed, we have seen that the Turing degrees are closed under complementation, or by Proposition 3-7.4, if a set and its complement are c.e., Then they are computable. In particular, the decision problem  $\emptyset'$  is c.e., but is Turing-equivalent to its complement which is not. However, we have defined a notion of c.e. degree at the start of Section 7-3, definition that we repeat here:

**Definition 1.1.** A Turing degree is *c.e.* if it contains a computably enumerable set. ◇

We saw that the Turing degrees are not bounded, because for any degree  $\mathbf{d}$ , its Turing jump  $\mathbf{d}'$  is strictly above. The c.e. degrees, on the other hand, are bounded by  $\mathbf{0}'$ .

**Proposition 1.2.** The c.e. degrees have for maximum degree  $\mathbf{0}'$ . ★

PROOF. By Post's theorem (see Proposition 5-4.3), a set is c.e. if and only if it is many-one reducible to  $\emptyset'$ . The many-one reductions being special cases of Turing reductions, any c.e. degree is Turing-reducible to  $\emptyset'$ . ■

The c.e. degrees forming a subset of the Turing degrees, questions about Turing degrees also apply to computably enumerable degrees. Are they linearly ordered? Do they form a well-founded order? And before all this, are there c.e. degrees other than  $\mathbf{0}$ , the Turing degree of computable sets, and  $\mathbf{0}'$ , the Turing degree of the halting problem?

The c.e. degrees are notoriously difficult to handle, due to the computability constraint of their enumeration. The finite extension method is no longer suitable, and it will be necessary to appeal to very elaborate techniques to prove results similar to those obtained in the Turing degrees.

## 2. Permitting method

The permitting method allows to compute a c.e. set  $A$  from another c.e. set  $B$ . It is based on the notion of *computation function* seen at Section 4-7. Recall that, given a c.e. approximation  $(A_s)_{s \in \mathbb{N}}$  of a c.e. set  $A$  (or more generally for any  $\Delta_2^0$  approximation), the computation function associated with this approximation is the function  $c_A : \mathbb{N} \rightarrow \mathbb{N}$  which to  $n$  associates the smallest time  $s > n$  such that  $A_s \upharpoonright_n = A \upharpoonright_n$ . In particular, this function can be computed in  $A$ . Moreover, by Theorem 4-7.9, any dominating function  $c_A$  recomputes  $A$ .

**Proposition 2.1 (Permitting method).** Let  $A$  and  $B$  be c.e. sets of c.e. approximations  $(A_s)_{s \in \mathbb{N}}$  and  $(B_s)_{s \in \mathbb{N}}$ , respectively. If there exists a  $B$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $n, s \in \mathbb{N}$

$$A_{s+1} \upharpoonright_n \neq A_s \upharpoonright_n \Rightarrow B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)},$$

then  $A \leq_T B$ . ★

PROOF. For all  $n$ ,  $c_A(n) \leq c_B(f(n))$ , but  $c_B \oplus f \leq_T B$ , so  $B$  computes a function dominating  $c_A$ . By Theorem 4-7.9,  $B$  computes  $A$ . ■

Most of the time, the function  $f$  will be computable and increasing, or even the identity function. Informally, the approximation of  $A$  is only allowed to add an element to  $A$  if at the same time,  $B$  adds an element smaller than  $f(x)$ . In other words, the set  $A$  waits for permission from  $B$  to add elements, which gives this method its name. The permitting method is often combined with other techniques, like the priority method, which we will see in the next section. The permitting method does not lose in generality, in the sense that we can prove the following reversal in the case where the set  $B$  is infinite.

**Proposition 2.2.** Let  $A$  and  $B$  be c.e. sets such that  $A \leq_T B$  and  $B$  is infinite. Then, there exist c.e. approximations  $(A_s)_{s \in \mathbb{N}}$  and  $(B_s)_{s \in \mathbb{N}}$  and a  $B$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for all  $n, s \in \mathbb{N}$

$$A_{s+1} \upharpoonright_n \neq A_s \upharpoonright_n \Rightarrow B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)} \quad \star$$

PROOF. Let  $(A_s)_{s \in \mathbb{N}}$  and  $(B_s)_{s \in \mathbb{N}}$  be c.e. approximations of  $A$  and  $B$ , respectively. Knowing that  $B$  is infinite, by accelerating its c.e. approximation, we can assume without loss of generality that  $B_{s+1} \neq B_s$  for all  $s$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the function which to  $n$  associates the smallest integer  $m$  such that  $B_{s+1} \upharpoonright_m \neq B_s \upharpoonright_m$  for all  $s \leq c_A(n)$ . The function  $f$  is  $c_A \oplus B$ -computable, or  $c_A \leq_T A \leq_T B$ , so  $f \leq_T B$ . For all  $n, s \in \mathbb{N}$ , if  $A_{s+1} \upharpoonright_n \neq A_s \upharpoonright_n$ , then  $c_A(n) \geq s + 1$ , therefore  $B_{s+1} \upharpoonright_{f(n)} \neq B_s \upharpoonright_{f(n)}$ . ■

### 3. $\Sigma_1^0$ priority method (finite injury)

Post asked the question in 1944 [180] whether there are computably enumerable sets which are both non-computable and Turing incomplete, that is to say which do not allow as an oracle to compute the halting problem. The question remained open for more than a decade, before being solved in the affirmative independently by Muchnik [162] and Friedberg [62], who introduced the famous *priority method*. This technique will subsequently find

many applications for the study of c.e. and  $\Delta_2^0$  sets, which turn out to have a very rich structure, as witnessed by Sacks' density theorem: let  $X, Y$  be c.e. sets such that  $X <_T Y$ . Then, there exists a c.e. set  $Z$  such that  $X <_T Z <_T Y$ .

In general, the priority method serves the same purpose as the finite extension method (see Section 4-8), i.e., to construct sets satisfying properties of strength and weakness, but this time by controlling the complexity of these sets in the arithmetic hierarchy. This additional constraint is at the origin of an explosion in the complexity of constructions. Indeed, like the finite extension method, it consists of satisfying an infinity of requirements simultaneously, but while for the finite extension method, the construction is omniscient, the priority method argument must be carried out with a limited computational power. It is therefore necessary to continue building the set without having full knowledge of the situation. The construction is therefore done by trial and error, with backtracking when an error is noticed. The strategies to satisfy the requirements come into conflict, and the whole difficulty of the construction lies in ensuring that these conflicts and backtracking do not prevent the overall goal: to build a set satisfying all the requirements.

We simply start here with the easiest use of the priority method, which was the first to be introduced, to solve Post's problem. This is a *finite injury* priority method, meaning that each strategy to fulfill a requirement will only have to backtrack a finite number of times before it can achieve its goal.

**Theorem 3.1 (Friedberg (1957), Muchnik (1956))**

*There are incomparable c.e. sets for the Turing reduction.*

The statement of Friedberg and Muchnik's theorem is similar to Kleene and Post's theorem (see Proposition 4-8.1), but imposes the additional constraint on sets to be computably enumerable.

PROOF. As for the theorem of Kleene and Post (see Proposition 4-8.1), we are going to construct two sets,  $A$  and  $B$ , each satisfying a strength property and a weakness property. These properties are each declined in the form of two sets of requirements:  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : W_e^A \neq \overline{B} \quad \mathcal{S}_e : W_e^B \neq \overline{A}.$$

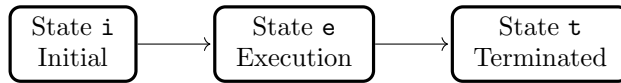
Recall the meaning of the notation  $W_e^A$  which designates the c.e. set relative to  $A$  of code  $e$  (ie  $\{n \in \mathbb{N} : \Phi_e(A, n) \downarrow\}$ ). The requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  ensure that  $A \not\geq_T B$  because the complement of  $B$  is not c.e. in  $A$ . Symmetrically, if the requirements  $(\mathcal{S}_e)_{e \in \mathbb{N}}$  are simultaneously satisfied, then  $B \not\geq_T A$ .

**Remark**

The sets  $A$  and  $B$  being c.e., it is equivalent to say that they are not computable, or that their complement is not c.e. Thus, the requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$  are without loss of generality. This formulation of requirements simplifies notations.

Since the sets  $A$  and  $B$  must be c.e., We are going to enumerate elements in  $A$  and in  $B$  during a computable process. More formally, we will define two uniformly computable sequences of finite sets  $A_0 \subseteq A_1 \subseteq \dots$  and  $B_0 \subseteq B_1 \subseteq \dots$  starting with  $A_0 = B_0 = \emptyset$ . For presentation reasons, we will omit the index  $s$  and will speak of  $A$  and  $B$  as sets under construction and evolving over time. We will use the indices when it is necessary to distinguish the sets at different time stages. For each  $\mathcal{R}_e$  or  $\mathcal{S}_e$  requirement, we will describe a process responsible for its satisfaction. The different processes will run in parallel. A process responsible for the satisfaction of a requirement  $\mathcal{R}_e$  (resp.  $\mathcal{S}_e$ ) is called *strategy for  $\mathcal{R}_e$*  (resp.  $\mathcal{S}_e$ ). In this construction, only one strategy will be needed per requirement. We will see priority arguments later which will associate several strategies with each requirement.

**Satisfaction of a requirement  $\mathcal{R}_e$ .** Here is a strategy for constructing two c.e. sets  $A$  and  $B$  satisfying a unique requirement  $\mathcal{R}_e$ . To gain in generality and prepare the satisfaction of several requirements, we will also assume that other processes or strategies run in parallel, and add elements to  $A$  and  $B$  over time. At any time  $t$ , the strategy for  $\mathcal{R}_e$  is found in one of the following three states:



*State i.* This is the initial state of the process. To get out of this state, the process goes through an initialization phase which consists in choosing a unique integer  $x_{\mathcal{R}_e} \notin B$ . This number exists because at any time, the sets  $A$  and  $B$  have a finite number of elements. Once  $x_{\mathcal{R}_e}$  is chosen, our process sets a *restraint* on  $x_{\mathcal{R}_e}$ , that is to say, it forbids other processes running in parallel to add  $x_{\mathcal{R}_e}$  in  $B$ . Only our process is the decision maker of the enumeration of  $x_{\mathcal{R}_e}$ . Note that once  $x_{\mathcal{R}_e}$  is added to  $B$ , it will no longer be able to leave it, because the set  $B$  is c.e. Once the restraint is set, the process enters State e, called execution.

*State e.* During this phase, the process executes  $\Phi_e^A(x_{\mathcal{R}_e})$  until it halts. If it never halts, the process remains in this state. Note that during this phase, the set  $A$  can evolve, because other processes can add elements to  $A$  in

parallel. The process must take this evolution into account, and therefore compute  $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t]$  at time  $t$ . If  $\Phi_e^{A_t}(x_{\mathcal{R}_e}) \downarrow$  at a time  $t$ , then the process poses a *restraint* on the use of this computation, that is to say on the bits of the oracle  $A_t$  used for this execution, thus preventing other processes from making a modification to it. We therefore make sure that  $\Phi_e^A(x_{\mathcal{R}_e}) \downarrow$ , in other words that  $x_{\mathcal{R}_e} \in W_e^A$ . The process then adds  $x_{\mathcal{R}_e}$  to  $B$ , so that  $W_e^A \neq \overline{B}$ , and ends up in State  $\mathfrak{t}$ .

**State  $\mathfrak{t}$ .** In this state, the process has completed its execution. He does leave this state.

**Outcomes.** Let us study the different outcomes of the strategy for  $\mathcal{R}_e$ . The process always goes from State  $\mathfrak{i}$  to State  $\mathfrak{e}$ , but may never reach State  $\mathfrak{t}$ . We therefore have two possible outcomes. Outcome  $\mathfrak{p}$ : it remains stuck in the execution state (Qstate  $\mathfrak{e}$ ). In this case,  $\Phi_e^A(x_{\mathcal{R}_e}) \uparrow$  and  $x_{\mathcal{R}_e} \notin B$ , so  $W_e^A \neq \overline{B}$ . We say that the requirement  $\mathcal{R}_e$  is *passively satisfied*. Outcome  $\mathfrak{a}$ : it enters the termination state (State  $\mathfrak{t}$ ). Then, by its actions of restraints and the enumeration of  $x_{\mathcal{R}_e}$  in  $B$ , it ensures that the requirement  $\mathcal{R}_e$  is satisfied by the second clause of the disjunction. We then say that the requirement  $\mathcal{R}_e$  is *actively satisfied*. In both cases, the requirement  $\mathcal{R}_e$  is satisfied.

#### State vs strategy outcome

It is important to distinguish between the state of a strategy and its outcome. The state of a strategy depends on each step, and is known information at that step. The outcome of the strategy is a limit behavior that is not known in finite time. We have only seen outcomes of the form “The strategy will remain in such state after a while”, but we will see other outcomes in the infinite injury priority method, such as “The strategy will go through all of its states successively.”

**Conflicts.** Complications arise when one wants to satisfy several requirements simultaneously. Indeed, the strategy of a requirement places restraints on finite segments of  $A$  and  $B$  over time, which can conflict with the needs of other strategies. We will therefore analyze to what extent the strategies can come into conflict, whether between strategies for requirements of the same type (for example  $\mathcal{R}_a$  and  $\mathcal{R}_b$ ), or between strategies for requirements of different type (for example  $\mathcal{R}_a$  and  $\mathcal{S}_b$ ).

**Satisfaction of all requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$ .** Generally speaking, in priority arguments, strategies for requirements of the same nature are relatively easy to satisfy simultaneously. The only possible conflicts between two requirements  $\mathcal{R}_e$  and  $\mathcal{R}_d$  would arise if the two strategies had chosen the same integer  $x$  (in other words  $x_{\mathcal{R}_e} = x_{\mathcal{R}_d}$ ) during their passage of State  $\mathfrak{i}$

in State **e**, and one of the two, say  $\mathcal{R}_e$ , sees its computation  $\Phi_e^A(x)$  terminate and seeks to add  $x$  in  $B$  to pass in the termination State **t**, while  $\mathcal{R}_d$  is still in the executing state (State **e**). To avoid these conflicts, it suffices to associate in State **i** a different integer  $x$  for each requirement  $\mathcal{R}_e$ . This is possible because,  $B$  being finite during the construction, there exists an infinity of integers outside  $B$ .

**Satisfaction of a requirement  $\mathcal{R}_e$  and  $\mathcal{S}_d$ .** Suppose now that we want to satisfy two requirements of opposite nature. By default, the requirement  $\mathcal{S}_d$  playing a symmetrical role, the strategy to satisfy it is obtained from that for  $\mathcal{R}_e$  by substituting  $A$  for  $B$  and vice versa:

- State **i**: Choose an integer  $x_{\mathcal{S}_d} \notin A$  and restrain  $x_{\mathcal{S}_d}$ .
- State **e**: Execute  $\Phi_d^B(x_{\mathcal{S}_d})$ . If execution stops at time  $t$ , restrain the use of  $\Phi_d^{B_t}(x_{\mathcal{S}_d})$  and add  $x_{\mathcal{S}_d}$  to  $A$ , then switch to State **t**.
- State **t**: The process is complete.

A new conflict can arise between the strategy for  $\mathcal{R}_e$  and that for  $\mathcal{S}_d$ . In State **e**, the strategy for  $\mathcal{R}_e$  may see execution of  $\Phi_e^A(x)$  terminate with use of  $s$  oracle bits, for  $s > x_{\mathcal{S}_d}$ . It then restrains  $A_t \upharpoonright_s$ , preventing other processes from modifying these values. If, later, the strategy for  $\mathcal{S}_d$  sees its execution of  $\Phi_d^B(x_{\mathcal{S}_d})$  terminate, it will not be able to add  $x_{\mathcal{S}_d}$  to  $A$  because of the previous restraint. The strategy for  $\mathcal{S}_d$  is *injured* and will have to return to its State **i**, choose a new integer  $x_{\mathcal{S}_d}$  large enough not to be restrained, and start the procedure again. The strategy for  $\mathcal{R}_e$  being in the termination state (State **t**), it will no longer act and will therefore no longer pose a new restraint which risks injuring the strategy for  $\mathcal{S}_d$ .

In general, one can satisfy a finite number of requirements  $\mathcal{R}_e$  and  $\mathcal{S}_d$  simultaneously with the same method. As soon as a strategy restrains the use of its computation when this one passes to State **t** and ends, it injures all the strategies which have not reached State **t** yet, and which then return to State **i**. The strategies thus injured will then choose new elements free from any restraint. Note that when a strategy enters State **t**, it no longer leaves it. The injuries being caused only by the a strategy arriving in State **t**, each strategy is only injured a finite number of times, and will end up entering State **t**, or will remain stuck in State **e**.

**Satisfaction of all requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ .** A new problem arises when we want to satisfy an infinity of requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ . Suppose the strategy for  $\mathcal{R}_e$  chooses in State **i** an integer  $x_{\mathcal{R}_e}$  and starts the execution of  $\Phi_e^A(x_{\mathcal{R}_e})$ , but does not have time to reach the end of the execution, because the strategy for a requirement  $\mathcal{S}_{d_0}$  sees its

own execution terminate before  $\mathcal{R}_e$ , and places a restraint on  $x_{\mathcal{R}_e}$  to preserve the use of its computation. The strategy for  $\mathcal{R}_e$  is then injured, and goes back to State **i** and chooses a new integer  $x_{\mathcal{R}_e}$  and starts the execution of  $\Phi_e^A(x_{\mathcal{R}_e})$  again with its new  $x_{\mathcal{R}_e}$ . Before reaching the end of its computation, for lack of luck, another strategy  $\mathcal{S}_{d_1}$  reaches State **t**, and again injures the strategy for  $\mathcal{R}_e$ , and so on infinitely often. The strategy for  $\mathcal{R}_e$  will then change infinitely often integers  $x_{\mathcal{R}_e}$ , and the requirement  $\mathcal{R}_e$  will never be satisfied.

To solve this problem, we will order the strategies. Let  $R_e$  be the strategy for  $\mathcal{R}_e$  and  $S_e$  the strategy for  $\mathcal{S}_e$ . We order the strategies as follows:

$$R_0 > S_0 > R_1 > S_1 > R_2 > S_2 > \dots$$

considering that a strategy has lower priority than the strategies to its left, but higher priority than those to its right. For example, the strategy for  $S_1$  has lower priority than the strategies for  $\mathcal{R}_0$  and  $S_0$ , but has higher priority than the strategies for  $\mathcal{R}_2, S_2$  and so on. Thus, each strategy is “below” a finite number of strategies, and “above” the rest.

#### Remark

In the proof of Friedberg and Muchnik’s theorem, each requirement has exactly one strategy, which makes the distinction between strategy  $R_e$  and requirement  $\mathcal{R}_e$  useless. The order given on the strategies thus induces an order on the requirements. However, in the following constructs, when requirements will be assigned more than one strategy, it will be essential to give total priority order to the strategies and not to the requirements.

The golden rule that we establish is then the following.

The restraints imposed by a strategy only apply to strategies of lower priority. Thus, a strategy can only be injured by the higher priority strategies, and can injure all lower priority strategies.

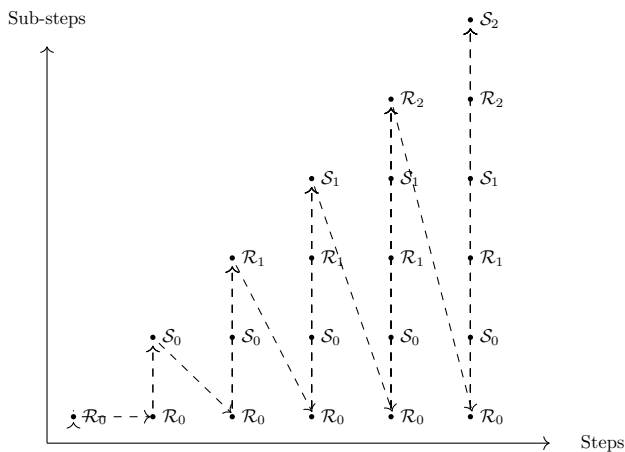
Assuming that a strategy poses only a finite number of restraints before reaching its terminal state, and ceases to act after a while if it is not injured, a simple induction shows that each strategy gets injured a finite number of times. Contrary to the satisfaction of a finite number of requirements simultaneously, it is possible that a process arrives at State **t**, but is then injured by a strategy of higher priority which will have ignored the imposed restraint. Fortunately, after a while, the higher priority strategy will stop injuring the weaker one, which will eventually stabilize.

**Construction.** Formally, the construction is done in stages  $t = 0, 1, \dots$  and each stage is divided into sub-stages  $s < t$ . Initially, all strategies are in

State **i**. In step  $t \geq 0$  and sub-step  $s < t$ , we consider the requirement  $\mathcal{R}_e$  if  $s = 2e$  and  $\mathcal{S}_e$  if  $s = 2e + 1$ . At the start of sub-step  $s = 2e$ , the strategy for  $\mathcal{R}_e$  has 3 possible states:

- (i) The strategy chooses an integer  $x_{\mathcal{R}_e} \notin B_t$  that has not been restrained by a higher priority strategy, and restrains it. It is then found in State **e**.
- (e) The strategy executes  $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t]$ . If  $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t] \downarrow$ , then it restrains all the bits used for the computation of  $\Phi_s^{A_t}(x_{\mathcal{R}_e})[t]$ , and injures all the lower priority strategies by returning them to State **i**. It is then found in State **t**. If  $\Phi_e^{A_t}(x_{\mathcal{R}_e})[t] \uparrow$ , the strategy remains in State **e**.
- (t) The strategy does not act and remains in this state.

In the sub-step  $s = 2e + 1$ , we apply the same procedure for  $\mathcal{S}_e$  *mutatis mutandis*, then we go to the next sub-step, until reaching  $t$ , in which case we go to step  $t + 1$ , and so on. This concludes the construction.



**Verification:** First of all, let us notice that the construction described above is indeed a computable process which enumerates two sets  $A$  and  $B$ . In particular, once an element is listed in  $A$  or  $B$ , we do not change our mind and it stays there. Note also that if a strategy for a requirement is no longer injured after a step  $r$ , then the associated requirement will be satisfied passively or actively at the end of the construction. The difficulty lies in showing that for any requirement there is indeed such an  $r$ .

Let us show by induction on  $e \in \mathbb{N}$  that the strategies for the requirements  $\mathcal{R}_e$  and  $\mathcal{S}_e$  injure only finitely often strategies of lower priority. Suppose by induction hypothesis that there is a step  $t$  after which none of the

strategies for the requirements  $\mathcal{R}_d$  and  $\mathcal{S}_d$  with  $d < e$  injure lower priority strategies. In particular, from step  $t$ , the strategy for  $\mathcal{R}_e$  will no longer be injured, and either now remains in the execution state (State  $\mathfrak{e}$ ), or goes to after a while in the termination state (State  $\mathfrak{t}$ ), only injuring lower priority strategies one last time. The same reasoning applies to the strategy for the requirement  $\mathcal{S}_e$ . So each strategy injures finitely often lower priority strategies, and each requirement will end up being satisfied, passively or actively. This concludes the proof of Theorem 3.1. ■

### Remark

The previous proof was very detailed in order to give intuitions of the priority method. We will now go more directly to the final construction for the other proofs. However, it is often useful, when trying to prove a new result using the priority method, to proceed step by step, seeking to satisfy one requirement, then several simultaneously, to end up satisfying all of them, in order to better understand their interactions.

### Corollary 3.2

*There exists a non-computable c.e. set  $A$  such that  $A \not\leq_T \emptyset'$ .*

PROOF. Let  $A$  and  $B$  be two c.e. sets such that neither computes the other. Then, in particular, they are both non-computable. Moreover if  $A$  could compute the halting set, it would also compute  $B$  since any c.e. set is many-one reducible to the halting problem. ■

Before tackling more elaborate constructions based on the technique of finite injury priority methods of Friedberg and Muchnik, we see here another simple application.

### Theorem 3.3

*There exists a non-computable c.e. set of low degree.*

PROOF. We are going to construct a c.e. set  $A$  by the finite injury priority method, together with a stable total computable function  $\Gamma : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  satisfying the requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : W_e \neq \overline{A} \quad \mathcal{S}_e : A'(e) = \lim_t \Gamma(e, t).$$

Satisfying all the requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  ensures that the set  $A$  is not computable, while the requirements  $(\mathcal{S}_e)_{e \in \mathbb{N}}$  cause  $A'$  (the halting problem relative to  $A$ ) to admit a  $\Delta_2^0$  approximation, which, by the Shoenfield limit

lemma (see Lemma 4-7.2), ensures that  $A' \leq_T \emptyset'$ , and therefore that  $A$  is of low degree.

**Satisfaction of a requirement  $\mathcal{R}_e$ .** Here is a strategy for satisfying a requirement  $\mathcal{R}_e$ , independently of other requirements. Our strategy has 3 states: initial (i), in execution (e) and terminated (t). The strategy takes the following steps depending on its condition.

- State i: Choose an integer  $x_{\mathcal{R}_e} \notin A$  and restrain  $x_{\mathcal{R}_e}$ .
- State e: Execute  $\Phi_e(x_{\mathcal{R}_e})$ . If the execution stops at time  $t$ , add  $x_{\mathcal{R}_e}$  to  $A$ , then switch to State t.
- State t: The process has completed its execution and remains in this state.

Assuming that the strategy is only injured a finite number of times, it could have two possible outcomes. Issue p: it will end up staying in State e, in which case  $\Phi_e(x_{\mathcal{R}_e}) \uparrow$  and  $x_{\mathcal{R}_e} \notin A$ , so  $W_e \neq \bar{A}$ . In this case the requirement  $\mathcal{R}_e$  is said to be passively satisfied. Issue a: The strategy will reach State t and stop. In this case,  $\Phi_e(x_{\mathcal{R}_e}) \downarrow$ ,  $x_{\mathcal{R}_e} \in A$ , and  $\mathcal{R}_e$  is said to be actively satisfied.

**Satisfaction of a requirement  $\mathcal{S}_e$ .** In step  $t$ , we will define the value of  $\Gamma(x, t)$  for all  $x < t$ . The strategy has 2 states: in execution (e) and completed (t). Here is the procedure to follow according to each state of the strategy.

- State e: Execute  $\Phi_e^A(e)$ . While  $\Phi_e^A(e)[t] \uparrow$ , hold  $\Gamma(e, t) = 0$ . If the execution stops at time  $t$ , restrains the use of  $\Phi_e^A(e)$  then go to State t.
- State t: Define  $\Gamma(e, t) = 1$  for any new step  $t$ .

The two possible outcomes of the strategy are as follows. Issue p: it will eventually remain in execution state (State e), in which case  $\Phi_e^A(e) \uparrow$  and  $\lim_t \Gamma(e, t) = 0$ . Thus,  $A'(e) = 0 = \lim_t \Gamma(e, t)$ . In this case the requirement  $\mathcal{S}_e$  is said to be passively satisfied. Issue a: The strategy will reach State t and stop. In this case,  $\Phi_e^A(e) \downarrow$  and  $\lim_t \Gamma(e, t) = 1$ . So  $A'(e) = 1 = \lim_t \Gamma(e, t)$  and  $\mathcal{S}_e$  is said to be actively satisfied.

**Construction.** The construction is globally the same as that of Theorem 3.1. The strategies are ordered by decreasing priority  $R_0, S_0, R_1, S_1, \dots$ . The construction is divided into stages  $t = 0, 1, \dots$  and each stage is itself divided into sub-stages  $s < t$ . Initially, all strategies are in State i. In step  $t \geq 0$  and sub-step  $s < t$ , we consider the requirement  $\mathcal{R}_e$  if  $s = 2e$  and  $\mathcal{S}_e$  if  $s = 2e + 1$ . In sub-step  $s = 2e$ , we execute the strategy for  $\mathcal{R}_e$  as described above depending on its state. When it reaches State t, all lower

priority strategies are injured and either return to State **i** in the case of strategies for requirements of type  $\mathcal{R}$ , or in State **e** in the case of strategies for requirements of type  $\mathcal{S}$ . In the same way, in the sub-step  $s = 2e + 1$ , we execute the strategy  $\mathcal{S}_e$  as described above, injuring all the lower priority strategies if we reach the termination state **t**.

**Verification:** The set  $A$  produced is indeed c.e. because the process is computable, and does not remove any element from  $A$  once added. We easily prove by induction on  $e \in \mathbb{N}$  that the strategies for requirements  $\mathcal{R}_e$  and  $\mathcal{S}_e$  injure only finitely often lower priority strategies. Thus, each strategy is only finitely often injured, and will end up having limit behavior. It also follows that  $\lim_t \Gamma(e, t)$  exists, and by construction, is equal to  $A'(e)$ . This concludes the proof of Theorem 3.3. ■

## 4. $\Sigma_2^0$ priority method

In the previous constructions using the priority method, finite injuries are structurally ensured, that is, in the very structure of the construction, the strategies pose only a finite number of restraints when they are not injured, and are ensured by construction that they will only be injured a finite number of times. The number of injuries can even be computably bounded: In Friedberg and Muchnik's construction, the  $e$ -th process is injured at most  $2^e - 1$  times.

We will now see an elaboration of the previous method, which could structurally result in an infinite number of injuries of a strategy, but construction assumptions will ensure that this never happens. Technically, this is therefore a finite injury priority method, as each strategy will only be injured a finite number of times. However, this elaboration can be seen as a degenerate version of the infinite injury priority method, shown in the next section.

### Theorem 4.1 (Sacks)

*For any non-computable c.e. set  $B$ , there exists a c.e. non-computable set  $A$  which does not compute  $B$ .*

PROOF. We are going to build a c.e. set  $A$  satisfying the following requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : W_e \neq \bar{A} \quad \mathcal{S}_e : \Phi_e^A = B \Rightarrow B \text{ is computable.}$$

Satisfying all requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  ensures that the set  $A$  is not computable, while requirements  $(\mathcal{S}_e)_{e \in \mathbb{N}}$  ensure that  $B \not\leq_T A$ . The require-

ment  $\mathcal{S}_e$  could also have been noted  $W_e^A \neq \overline{B}$ , but its current formulation better represents the form of the argument used to satisfy it.

**Satisfaction of a requirement  $\mathcal{R}_e$ .** The satisfaction of a requirement  $\mathcal{R}_e$  is exactly the same as for Theorem 3.3. We recall the actions of the strategy according to its three states:

- State **i**: Choose an integer  $x_{\mathcal{R}_e} \notin A$  and restrain  $x_{\mathcal{R}_e}$ .
- State **e**: Execute  $\Phi_e(x_{\mathcal{R}_e})$ . If the execution stops at time  $t$ , add  $x_{\mathcal{R}_e}$  to  $A$ , then switch to State **t**.
- State **t**: The process is complete.

The two outcomes of the strategy are still **p** (passive satisfaction) and **a** (active satisfaction).

**Satisfaction of a requirement  $\mathcal{S}_e$ .** The strategy to satisfy  $\mathcal{S}_e$  is more complex and less intuitive. It consists in trying to make longer and longer initial segments of  $\Phi^A$  and  $B$  coincide, in a computable way, by setting each time larger and larger restraints on  $A$  to preserve the use of  $\Phi^A$ . At first glance, this strategy will therefore cause infinite injury to lower priority strategies. Fortunately, the set  $B$  not being computable, the procedure will stop finding new initial segments coinciding, and will thus satisfy the requirement  $\mathcal{S}_e$ . More precisely, the strategy has an initial state (State **i**), and an infinity of states  $(\mathbf{w}_n)_{n \in \mathbb{N}}$ . During the execution of the process, we will define a computable function  $\Delta : \mathbb{N} \rightarrow \{0, 1\}$  supposed to coincide with the characteristic function of  $B$ . Let  $(B_t)_{t \in \mathbb{N}}$  be a c.e. approximation of  $B$ .

- State **i**: Define  $\Delta$  as the empty domain function, and change to State  $\mathbf{w}_0$ .
- State  $\mathbf{w}_n$ : Wait for a step  $t \geq n$  where  $\Phi_e^{A_t}[t] \upharpoonright_{n+1} = B_t \upharpoonright_{n+1}$ . If this happens, restrain the use of  $\Phi_e^{A_t}[t] \upharpoonright_{n+1}$ , define  $\Delta(n) = B_t(n)$ , and switch to State  $\mathbf{w}_{n+1}$ .

Here,  $\Phi_e^{A_t}[t] \upharpoonright_{n+1} = B_t \upharpoonright_{n+1}$  means that for all  $x \leq n$ ,  $\Phi_e^{A_t}(x)[t] \downarrow \in \{0, 1\}$ , and  $\Phi_e^{A_t}(x)[t] = 1$  iff  $x \in B_t$  for  $x \leq n$ . The strategy does not return to State **i** unless injured. At this time, the  $\Delta$  function is also reset. However, assuming that the strategy is injured a finite number of times, the function will only be reset finitely often, and therefore will be defined in a computable way.

The strategy has an infinite number of outcomes: for all  $n$ , the outcome  $\mathbf{p}_n$  consists of remaining stuck in State  $\mathbf{w}_n$ . If the strategy never goes out of this state, then  $\Phi_e^{A_t}[t] \upharpoonright_{n+1} \neq B_t \upharpoonright_{n+1}$  for all  $t$ , hence  $\Phi_e^A \neq B$ , and the requirement  $\mathcal{S}_e$  is satisfied because the premise of the implication is false. The strategy has a last possible outcome, of an infinite nature, which

consists in going through all the states  $\mathbf{w}_n$ . Let us name this issue  $\infty$ . By using the hypothesis according to which  $B$  is not computable, we will now show that this outcome cannot happen.

**Lemma 4.2.** If the outcome  $\infty$  occurs, then  $B$  is computable. ★

PROOF. Let  $e$  be the smallest integer such that the strategy for  $\mathcal{S}_e$  goes through all states  $(\mathbf{w}_n)_{n \in \mathbb{N}}$ . By induction, all higher priority strategies are finitely injured, and will reach a limit state where they will no longer injure the strategy for  $\mathcal{S}_e$ . From this moment, the strategy for  $\mathcal{S}_e$  will go through all the states  $(\mathbf{w}_n)_{n \in \mathbb{N}}$  successively. The function  $\Delta$  defined by the process is then total computable. Let us show that  $\Delta$  is the characteristic function of  $B$ . Let us assume absurdly that  $\Delta(n) \neq B(n)$  for an  $n \in \mathbb{N}$ . By definition of  $\Delta$ ,  $\Delta(n) = B_t(n) = \Phi_e^{A_t}(n)$  for a  $t \in \mathbb{N}$ . Since  $B$  is c.e., This difference comes from an element that appears in  $B$ , because no element can come out of it. Thus,  $\Delta(n) = B_t(n) = \Phi_e^{A_t}(n) = 0$  and  $n \in B$ . Let  $m$  be large enough such that  $n \in B_m$ . Then, at state  $\mathbf{w}_m$ ,  $\Phi_e^{A_t}(n)[t] = 0 \neq B_t(n)$  for all  $t \geq m$ , so  $\Phi_e^{A_t}[t] \upharpoonright_{n+1} \neq B_t \upharpoonright_{n+1}$  for all  $t \geq m$ , and the strategy will never reach State  $\mathbf{w}_{m+1}$ . Contradiction. The function  $\Delta$  is therefore the characteristic function of  $B$ , which proves that  $B$  is computable. ■

**Construction.** The overall construction is that of a standard finite injury priority method. The strategies are ordered by decreasing priority  $R_0, S_0, R_1, S_1, \dots$ . The construction is divided into stages  $t = 0, 1, \dots$  and each stage is itself divided into sub-stages  $s < t$ . Initially, all strategies are in State **i**. In step  $t \geq 0$  and sub-step  $s < t$ , we consider the requirement  $\mathcal{R}_e$  if  $s = 2e$  and  $\mathcal{S}_e$  if  $s = 2e + 1$ . In the sub-step  $s = 2e$ , the strategy for  $\mathcal{R}_e$  is executed as described above depending on its state. When it reaches State **e**, all lower priority strategies are injured and revert to State **i**. Likewise, in the sub-step  $s = 2e + 1$ , the strategy for  $\mathcal{S}_e$  is executed according to the steps described above. Each time it transitions to a next state  $\mathbf{w}_{n+1}$ , all lower priority strategies are injured and revert to State **i**.

Verification is left to the reader. This concludes the proof of Theorem 4.1. ■

### Sacks preservation strategy

The strategy to satisfy  $\mathcal{S}_e$  consists in not trying to actively differentiate two sets, but on the contrary, to preserve increasingly long common initial segments in a computable process, then to use the hypothesis of incomputability of one of the sets to deduce that this process should fail. This strategy is frequently found in this type of construction. It is sometimes referred to as *Sacks preservation strategy*, in honor of its

author.

## 5. $\Pi_2^0$ priority method (infinite injury)

We are now going to approach a new elaboration of the priority method, known as the infinite injury method. As the name suggests, some strategies will act infinitely often by placing larger and larger restraints, causing infinite injury to lower priority strategies. We will therefore start to have conditional strategies, adopting different behaviors depending on the outcomes of the higher priority strategies, thus taking full advantage of the formalism of the “strategy tree” that we will see soon.

Our illustration of the infinite injury priority method concerns the existence of minimal pairs of c.e. degrees. It improves the Friedberg-Muchnik theorem by combining it with Sacks’ preservation strategy.

**Definition 5.1.** Two non-computable degrees  $\mathbf{a}$  and  $\mathbf{b}$  form a *minimal pair* if their lower bound is  $\mathbf{0}$ , in other words if for any set  $A$  such that  $A \leq_T \mathbf{a}$  and  $A \leq_T \mathbf{b}$ , then  $A$  is computable.  $\diamond$

The existence of minimal pairs of c.e. degrees has been independently proven by Lachlan [128] and Yates [233].

### Theorem 5.2 (Lachlan 1966, Yates 1966)

*There is a minimum pair of c.e. degrees*

PROOF. We are going to construct two c.e. sets  $A$  and  $B$  satisfying the following requirements  $(\mathcal{R}_e, \mathcal{S}_e, \mathcal{N}_e)_{e \in \mathbb{N}}$ :

$$\mathcal{R}_e : W_e \neq \bar{A} \quad \mathcal{S}_e : W_e \neq \bar{B} \quad \mathcal{N}_e : \Phi_e^A = \Phi_e^B \Rightarrow \Phi_e^A \text{ is computable.}$$

The requirements  $(\mathcal{R}_e)_{e \in \mathbb{N}}$  and  $(\mathcal{S}_e)_{e \in \mathbb{N}}$  ensure that  $A$  and  $B$  are not computable. The  $(\mathcal{N}_e)_{e \in \mathbb{N}}$  requirements force the lower bound of the degrees of  $A$  and  $B$  to be  $\mathbf{0}$ .

### Posner’s trick

At first glance, to impose that the lower bound  $\deg_T A$  and  $\deg_T B$  is  $\mathbf{0}$ , one would expect to have to satisfy requirements of the form  $(\mathcal{N}_{i,j})_{i,j \in \mathbb{N}}$  with

$$\mathcal{N}_{i,j} : \Phi_i^A = \Phi_j^B \Rightarrow \Phi_i^A \text{ is computable.}$$

However, if  $\Phi_i^A = \Phi_j^B$ , then it is possible to create a new functional  $\Phi_e$  which will hardcode an integer  $n$  such that  $A(n) \neq B(n)$ , and execute  $\Phi_i$

or  $\Phi_j$  in function of the value of its oracle at position  $n$ . Thus,  $\Phi_e^A = \Phi_i^A$  and  $\Phi_e^B = \Phi_j^B$ . This trick, due to Posner, makes it possible to simplify the notations by using a single index.

**Satisfaction of a requirement  $\mathcal{R}_e$  or  $\mathcal{S}_e$ .** The satisfaction of a requirement  $\mathcal{R}_e$  or  $\mathcal{S}_e$  is exactly the same as for Theorem 3.3. We recall, in the case of  $\mathcal{R}_e$ , the actions of the strategy according to its three states:

- State **i**: Choose an integer  $x_{\mathcal{R}_e} \notin A$  and restrain  $x_{\mathcal{R}_e}$ .
- State **e**: Run  $\Phi_e(x_{\mathcal{R}_e})$ . If the execution stops at time  $t$ , add  $x_{\mathcal{R}_e}$  to  $A$ , then enter State **t**.
- State **t**: The process is complete.

So far, we have considered that this strategy has two outcomes, depending on the state in which it stabilizes. These two outcomes are of the same finite nature, in that they pose only a finite number of restraints when they are injured finitely often. We will therefore consider them as a single finitary outcome **f**.

**Satisfaction of a requirement  $\mathcal{N}_e$**  The satisfaction of a requirement  $\mathcal{N}_e$  will follow Sacks' preservation strategy to preserve increasingly long common initial segments to  $\Phi_e^A$  and  $\Phi_e^B$ . However, unlike Theorem 4.1, the process will not necessarily fail after a finite number of steps, because nothing in the assumptions prevents this equality. We are therefore in a case where the infinite outcome will be able to occur, with increasingly long restraints, resulting in an infinite injury. As in the case of Theorem 4.1, the strategy has an initial state **i**, and an infinity of states  $(\mathbf{w}_n)_{n \in \mathbb{N}}$ . In what follows, we will call *use* of  $\Phi_e^{A_t}[t] \upharpoonright_{n+1}$  the maximum of uses of  $\{\Phi_e^{A_t}(x)[t] : x \leq n\}$ . During the execution of the strategy, we will define a computable function  $\Delta : \mathbb{N} \rightarrow \{0, 1\}$  such that if  $\Phi_e^A$  and  $\Phi_e^B$  are total and equal, then they are both equal to  $\Delta$ .

- State **i**: Define  $\Delta$  as the empty domain function, and go to State  $\mathbf{w}_0$ .
- State  $\mathbf{w}_n$ : Wait for a step  $t \geq n$  where  $\Phi_e^{A_t}[t] \upharpoonright_{n+1} = \Phi_e^{B_t}[t] \upharpoonright_{n+1}$ . If this happens, release its previous restraint, and place a restraint on the use of  $\Phi_e^{A_t}[t] \upharpoonright_{n+1}$  if  $n$  is even, and on the use of  $\Phi_e^{B_t}[t] \upharpoonright_{n+1}$  if  $n$  is odd. Next, define  $\Delta(n) = \Phi_e^{A_t}(n)[t]$ , and go to State  $\mathbf{w}_{n+1}$ .

The strategy has two possible outcomes. Issue **f** (finitary): it gets stuck in state  $\mathbf{w}_n$  for a given  $n$ . In this case, either  $\Phi_e^A$  or  $\Phi_e^B$  is partial, or  $\Phi_e^A \neq \Phi_e^B$ . Issue  $\infty$  (infinitary): the strategy goes through all states  $(\mathbf{w}_n)_{n \in \mathbb{N}}$ . In this case, the two sets coincide, and infinitely often, the restraint changes sides.

We still have to prove that  $\Delta = \Phi_e^A = \Phi_e^B$  to deduce that this set is computable. The underlying idea of the proof is very simple, but it is a bit cumbersome to formalize. We will therefore illustrate it with a figure (see Figure 5.3) before formally proving the result through Lemma 5.4. Whatever the outcome, the requirement  $\mathcal{N}_e$  is therefore satisfied.

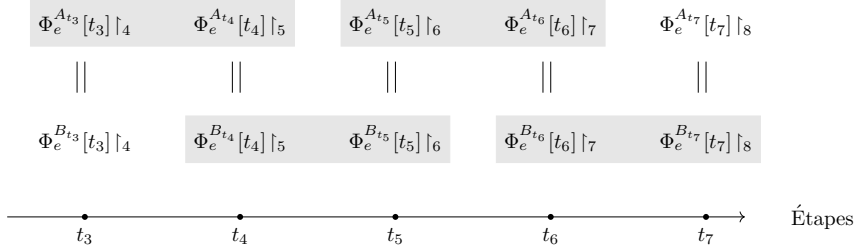


Figure 5.3: Let  $t_n$  be the step at which we reach State  $\mathbf{w}_{n+1}$ . The grey rectangle between step  $t_3$  and step  $t_4$  means that the use of  $\Phi_e^{A_{t_3}}[t_3] \upharpoonright_4$  is restraint, hence is preserved up to step  $t_4$ . Thus,  $\Phi_e^{A_{t_3}}[t_3] \upharpoonright_4 = \Phi_e^{A_{t_4}}[t_4] \upharpoonright_4$ . At each step  $t_n$ , the initial segments of length  $n + 1$  of both functionals coincide. Thus,  $\Phi_e^{A_{t_4}}[t_4] \upharpoonright_5 = \Phi_e^{B_{t_4}}[t_4] \upharpoonright_5$ . We therefore have  $\Phi_e^{B_{t_3}}[t_3] \upharpoonright_4 = \Phi_e^{A_{t_3}}[t_3] \upharpoonright_4 = \Phi_e^{A_{t_4}}[t_4] \upharpoonright_4 = \Phi_e^{B_{t_4}}[t_4] \upharpoonright_4$ . Even if the use of  $\Phi_e^{B_{t_3}}[t_3] \upharpoonright_4$  can differ from that of  $\Phi_e^{B_{t_4}}[t_4] \upharpoonright_4$ , since no restraint is posed on  $B$  between steps  $t_3$  and  $t_4$ , the first 4 output values of the functional are preserved.

### Remark

The choice of the restrained side ( $A$  if the strategy goes from a state  $\mathbf{w}_n$  to  $\mathbf{w}_{n+1}$  with  $n$  even, and  $B$  if  $n$  is odd) does not intervene in the proof of the validity of a strategy  $\mathcal{N}_e$  independently of the others. We could just as well always have kept the same side, or even restrained both sides, which would have considerably simplified the proof of validity. However, this alternation of sides becomes necessary when one seeks to satisfy a requirement of type  $\mathcal{R}$  or  $\mathcal{S}$  under a strategy for  $\mathcal{N}_e$ , as we will see hereafter.

**Lemma 5.4.** If the outcome  $\infty$  happens, then  $\Delta = \Phi_e^A = \Phi_e^B$ . ★

PROOF. Let  $P(n, s)$  be the proposition “Either (1)  $\Delta \upharpoonright_{n+1} = \Phi_e^{A_s}[s] \upharpoonright_{n+1}$  with a restraint on its use, or (2)  $\Delta \upharpoonright_{n+1} = \Phi_e^{B_s}[s] \upharpoonright_{n+1}$  with a restraint on its use.” For all  $n$ , let  $t_n$  be the step at which the strategy changes to state  $\mathbf{w}_{n+1}$ . We will show by induction on  $n$  and  $s$  that for all  $n \geq 0$  and  $s \geq t_n$ , the proposition  $P(n, s)$  is true. By convention,  $t_{-1} = 0$  and for all  $s \geq t_{-1}$ ,  $P(-1, s)$  is true.

Let  $n \geq 0$ . Let us show that if for all  $s \geq t_{n-1}$ ,  $P(n-1, s)$  is true, then  $P(n, t_n)$  is true. In step  $t_n$ ,  $\Delta(n) = \Phi_e^{A_{t_n}}(n)[t_n] = \Phi_e^{B_{t_n}}(n)[t_n]$ , and  $\Phi_e^{A_{t_n}}[t_n] \upharpoonright_{n+1} = \Phi_e^{B_{t_n}}[t_n] \upharpoonright_{n+1}$ . As  $t_n \geq t_{n-1}$ , by induction hypothesis,  $P(n-1, t_n)$  is true, so either  $\Delta \upharpoonright_n = \Phi_e^{A_{t_n}}[t_n] \upharpoonright_n$  or  $\Delta \upharpoonright_n = \Phi_e^{B_{t_n}}[t_n] \upharpoonright_n$ . So  $\Delta \upharpoonright_{n+1} = \Phi_e^{A_{t_n}}[t_n] \upharpoonright_{n+1} = \Phi_e^{B_{t_n}}[t_n] \upharpoonright_{n+1}$ . At this step, the strategy restrains  $\Phi_e^{A_{t_n}}[t_n] \upharpoonright_{n+1}$  or  $\Phi_e^{B_{t_n}}[t_n] \upharpoonright_{n+1}$ , so  $P(n, t_n)$  is true.

Let  $s > t_n$ . Let us show that if  $P(n, s-1)$  is true, then  $P(n, s)$  is true. If the strategy does not change state at step  $s$ , then it keeps its restraint, and by the use property,  $P(n, s)$  remains true. If the strategy changes to a  $\mathbf{w}_{p+1}$  state, then by definition,  $\Phi_e^{A_s}[s] \upharpoonright_{p+1} = \Phi_e^{B_s}[s] \upharpoonright_{p+1}$ , or  $p \geq n$ , so  $\Phi_e^{A_s}[s] \upharpoonright_{n+1} = \Phi_e^{B_s}[s] \upharpoonright_{n+1}$ . By  $P(n, s-1)$ , either  $\Delta \upharpoonright_{n+1} = \Phi_e^{A_{s-1}}[s-1] \upharpoonright_{n+1}$  with a restraint on its use, or (2)  $\Delta \upharpoonright_{n+1} = \Phi_e^{B_{s-1}}(n)[s-1] \upharpoonright_{n+1}$  with a restraint on its use. By the restraint to step  $s-1$  and the use property, either  $\Delta \upharpoonright_{n+1} = \Phi_e^{A_s}[s] \upharpoonright_{n+1}$ , or  $\Delta \upharpoonright_{n+1} = \Phi_e^{B_s}[s] \upharpoonright_{n+1}$ . It follows that  $\Delta \upharpoonright_{n+1} = \Phi_e^{A_s}[s] \upharpoonright_{n+1} = \Phi_e^{B_s}[s] \upharpoonright_{n+1}$ . The strategy restrains the use of  $\Phi_e^{A_s}[s] \upharpoonright_{p+1}$  or  $\Phi_e^{B_s}[s] \upharpoonright_{p+1}$ , so  $P(n, s)$  is true. This ends the proof of the lemma. ■

Note that unlike Theorem 4.1, the infinite outcome will really occur with the strategy for  $\mathcal{N}_e$ . It therefore does not combine that well with the strategies for  $\mathcal{R}_d$  and  $\mathcal{S}_d$ . Indeed, when this outcome occurs, the strategy for  $\mathcal{N}_e$  will impose increasingly long restraints, causing infinite injury. We will therefore have to adapt the construction to allow the other requirements to be satisfied.

### Execution step

It is not necessary to execute the strategies for  $\mathcal{R}_e$ ,  $\mathcal{S}_e$  and  $\mathcal{N}_e$  at each step to satisfy their respective constraints. It suffices to perform them each for an infinite number of steps, while maintaining their restraints during the intermediate steps.

Let us take the example of the strategy for  $\mathcal{R}_e$ . If it is only executed at times  $t_0 < t_1 < \dots$ , it may be that it “misses” the first step  $t$  between  $t_0$  and  $t_1$  where  $\Phi_e(x_{\mathcal{R}_e})[t]$  stops, which prevents it from entering State  $\mathbf{t}$  at this step. However, in step  $t_1$ ,  $\Phi_e(x_{\mathcal{R}_e})[t_1]$  will also stop, and the transition to State  $\mathbf{t}$  will still take place. The limit behavior of the strategy for  $\mathcal{R}_e$  therefore does not depend on the choice of steps.

The case of the strategy for  $\mathcal{N}_e$  is a bit more subtle. It may be that the default strategy for  $\mathcal{N}_e$  has an infinite outcome, but that when it is executed only at times  $t_0 < t_1 < \dots$ , we have  $\Phi_e^{A_{t_i}}[t_i] \upharpoonright_{n+1} \neq \Phi_e^{B_{t_i}}[t_i] \upharpoonright_{n+1}$ , so that the strategy does not will never change to state  $\mathbf{w}_{n+1}$ , which is the finite outcome. Fortunately, even in this case  $\mathcal{N}_e$  will be satisfied,

as long as the enumeration of steps  $(t_i)_{i \in \mathbb{N}}$  is computable so that the function  $\Delta$  is also computable.

**Satisfaction of a requirement  $\mathcal{R}_d$  or  $\mathcal{S}_d$  under  $\mathcal{N}_e$ .** Suppose we want to satisfy a requirement  $\mathcal{R}_d$  under the strategy for  $\mathcal{N}_e$ , i.e., with the strategy for  $\mathcal{N}_e$  of higher priority than that for  $\mathcal{R}_d$ . Several solutions arise, depending on the outcome of the strategy for  $\mathcal{N}_e$ :

- **Issue  $\mathbf{f}$  (finitary).** In this case, it suffices to use the standard strategy for  $\mathcal{R}_e$  presented above. Indeed, the strategy for  $\mathcal{N}_e$  will set a finite number of restraints, so that the strategy for  $\mathcal{R}_e$  will be injured and reset a finite number of times before being satisfied. This strategy does not work if the outcome of the strategy for  $\mathcal{N}_e$  is infinite (Issue  $\infty$ ). Indeed, in this case, the strategy for  $\mathcal{R}_e$  will be injured infinitely often, and might never satisfy  $\mathcal{R}_e$ .
- **Issue  $\infty$  (infinitary).** Note that in this case, the restraint posed by the strategy for  $\mathcal{N}_e$  will infinitely often alternate sideways, and will therefore free the other side, allowing the strategy for  $\mathcal{R}_d$  to be satisfied. We will therefore only execute the strategy for  $\mathcal{R}_d$  at the stages where the restraint of the strategy for  $\mathcal{N}_e$  is removed from the  $A$  side. The outcome of the strategy for  $\mathcal{N}_e$  being infinite, the strategy for  $\mathcal{R}_d$  will be executed for an infinite number of steps, and as explained above, an infinite subset of steps is sufficient to satisfy  $\mathcal{R}_d$ . This strategy for  $\mathcal{R}_d$  does not work, however, if the outcome of the strategy for  $\mathcal{N}_e$  is finite, because it may never remove its restraint and the strategy for  $\mathcal{R}_d$  therefore waits forever.

We therefore have two different strategies to satisfy  $\mathcal{R}_d$  under  $\mathcal{N}_e$ , depending on the outcome of the strategy for  $\mathcal{N}_e$ . This case analysis poses a difficulty: to produce a c.e. set, The construction must be a computable process, but the outcome of  $\mathcal{N}_e$  cannot be decided in a finite time. We therefore cannot know which strategy to choose for  $\mathcal{R}_d$ . The solution is to run the two strategies for  $\mathcal{N}_e$  in parallel, each making a guess about the outcome. The one making the correct assumption will then be able to satisfy  $\mathcal{R}_d$  under  $\mathcal{N}_e$ . We will therefore end up with a tree of strategies, induced by the successive case analysis on the outcomes of strategies of type  $\mathcal{N}$ . We will detail this tree structure below.

**Satisfaction of a requirement  $\mathcal{N}_d$  under  $\mathcal{N}_e$ .** Similar requirements are often easy to satisfy simultaneously because their strategies generally do not conflict. In the case of the satisfaction of a requirement  $\mathcal{N}_d$  under  $\mathcal{N}_e$ , the difficulty is not to satisfy these two requirements simultaneously, but

then to leave room for a requirement of the type  $\mathcal{R}$  or  $\mathcal{S}$  to be satisfied under  $\mathcal{N}_d$ . Indeed, in the worst case, the strategies for  $\mathcal{N}_e$  and  $\mathcal{N}_d$  will both be infinitary, and each time the strategy for  $\mathcal{N}_e$  releases its restraints on the  $A$  side, the strategy for  $\mathcal{N}_d$  will set its own, so that the  $A$  side will have increasingly larger restraints at all times, not allowing strategies of type  $\mathcal{R}$  to be satisfied below. The solution is to “synchronize” the strategies for  $\mathcal{N}_d$  and  $\mathcal{N}_e$ . More precisely, the strategy for  $\mathcal{N}_d$  will be defined by case analysis depending on the outcome of  $\mathcal{N}_e$ :

- Issue  $\mathfrak{f}$  (finitary). In this case, the strategy for  $\mathcal{N}_d$  is the standard strategy presented above. Indeed, the restraints of the strategy for  $\mathcal{N}_e$  will be finitary, and the other requirements will have the possibility of being satisfied under  $\mathcal{N}_d$  by being injured finitely often by the strategy of  $\mathcal{N}_e$ .
- Issue  $\infty$  (infinitary). Let’s modify the strategy for  $\mathcal{N}_d$ , as follows: This strategy will only be executed during stages where the strategy for  $\mathcal{N}_e$  changes the side of its restraints, in other words changes from state  $\mathfrak{w}_n$  to  $\mathfrak{w}_{n+1}$ . This makes sure that when the strategy for  $\mathcal{N}_d$  changes its restraints side, at the same step, the strategy for  $\mathcal{N}_e$  will change its side restraints. Finally, we must ensure that these changes go on the same side. For this, if the strategy for  $\mathcal{N}_d$  is in a state  $\mathfrak{w}_n$  with  $n$  an even integer, in other words if its restraints are on the  $B$  side, the strategy for  $\mathcal{N}_d$  will not be executed only during the steps where the strategy for  $\mathcal{N}_e$  changes from a state  $\mathfrak{w}_m$  to  $\mathfrak{w}_{m+1}$  with  $m$  even, so that both strategies put their restraints on the side  $A$  simultaneously. Likewise, if the strategy for  $\mathcal{N}_d$  is in a state  $\mathfrak{w}_n$  with  $n$  an odd integer, it will be executed during the steps where the strategy for  $\mathcal{N}_e$  changes from a state  $\mathfrak{w}_m$  to  $\mathfrak{w}_{m+1}$  with  $m$  odd.

As in the case of the satisfaction of a requirement  $\mathcal{R}_d$  under a requirement  $\mathcal{N}_e$ , we therefore have different strategies for  $\mathcal{N}_d$  for each outcome of the strategy for  $\mathcal{N}_e$ . We are therefore going to run two strategies in parallel, each assuming that its hypothesis is correct. We will now describe the strategy tree.

**Strategy tree.** The requirements are listed as follows.

$$\mathcal{N}_0, \mathcal{R}_0, \mathcal{S}_0, \mathcal{N}_1, \mathcal{R}_1, \mathcal{S}_1, \dots$$

In the previous constructions, each requirement was assigned a unique strategy, and the priority order of the strategies followed the enumeration of requirements. We now have a tree structure of strategies based on the outcomes of the previous strategies, as follows: the requirement  $\mathcal{N}_0$  has a single strategy  $\mathcal{N}_\epsilon$  (where  $\epsilon$  is the empty string). The requirement  $\mathcal{R}_0$

has two strategies  $R_\infty$  and  $R_f$ , depending on the outcomes  $\infty$  and  $f$  of the strategy  $N_0$ . The requirement  $S_0$  also has two strategies  $S_{\infty f}$  and  $S_{ff}$ , depending on the outcomes of the higher strategies. For example,  $S_{\infty f}$  must satisfy the requirement  $S_0$  under the assumption that the outcome of  $N_\epsilon$  is  $\infty$  and the outcome of  $R_\infty$  is  $f$ .

In general, we can define a tree  $\mathcal{T} \subseteq \{f, \infty\}^{<\mathbb{N}}$  of strings in the alphabet  $\{f, \infty\}$ , induced by the prefix relation, such that for all  $\sigma \in \mathcal{T}$  and  $i < |\sigma|$  such that  $i \not\equiv 0 \pmod 3$ ,  $\sigma(i) = f$ . To each string  $\sigma \in \mathcal{T}$ , if  $|\sigma| = 3e$  we associate a strategy  $N_\sigma$  for  $\mathcal{N}_e$ , if  $|\sigma| = 3e + 1$  we associate a strategy  $R_\sigma$  for  $\mathcal{R}_e$  and if  $|\sigma| = 3e + 2$ , we associate a strategy  $S_\sigma$  for  $\mathcal{S}_e$  (see Figure 5.5). To simplify the notations, we will denote by  $C_\sigma$  the strategy whose index is  $\sigma$ .

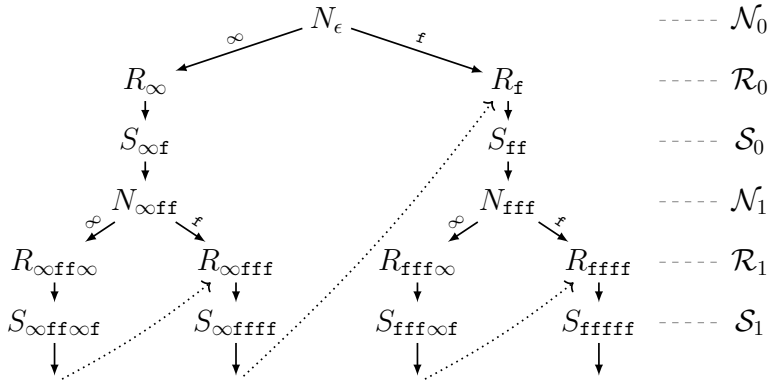


Figure 5.5: Strategy tree for the construction of a minimal pair. Full arrows indicate the tree structure, but also a partial order of priority between strategies. Dotted arrows enable to linearize the partial order to obtain a total order between strategies. For example, all the strategies on the left sub-tree of the strategy  $N_{\infty f f}$  are weaker than  $N_\epsilon$ ,  $R_\infty$ ,  $S_{\infty f}$  et  $N_{\infty f f}$ , but are stronger than all the other strategies.

**Construction.** At the start of the construction, each strategy is in the initial state (State i). At each step  $t$ , we will define a *current node*, which is a string  $\sigma_t \in \mathcal{T}$  of length  $t$ , representing an approximation of the outcomes at the end of step  $t$ . More precisely,  $\sigma_t(i)$  represents the approximation at step  $t$  of the outcome of the strategy  $C_{\sigma_t \upharpoonright i}$ . The current node at step  $t$  is defined by induction on the substeps  $s < t$  as follows: at the start of the substep  $i$ , we assume  $\sigma_t \upharpoonright i$  defined. We then execute the strategy  $C_{\sigma_t \upharpoonright i}$  at time  $t$  as described above. Recall that we may not execute the strategy  $C_{\sigma_t \upharpoonright i}$  for reasons of synchronization with the strategies for requirements

of type  $\mathcal{N}$ . In this case, the strategy remains in its state and keeps its constraints unchanged. At the end of sub-step  $t$ , we define  $\sigma_t(i) = \infty$  if  $i \equiv 0 \pmod 3$  and the strategy  $N_{\sigma_t \upharpoonright i}$  has changed state at step  $t$ . In all the other cases,  $\sigma_t(i) = \mathbf{f}$ , because the strategies of type  $\mathcal{R}$  and  $\mathcal{S}$  only have the finite possible outcome, and if  $i \equiv 0 \pmod 3$  and  $N_{\sigma_t \upharpoonright i}$  does not change state, we assume that its outcome is finite.

**True path.** We can now define the *true path* of  $\mathcal{T}$ , which is the path along which the outcomes are true. More precisely, the true path of  $\mathcal{T}$  is the infinite sequence  $P \in \Lambda^{\mathbb{N}}$  (with  $\Lambda = \{\infty, \mathbf{f}\}$  where  $\infty$  is less than  $\mathbf{f}$ ) defined inductively on  $i$  by

$$P(i) = \liminf \{o \in \Lambda : \exists^\infty t (P \upharpoonright i)^\frown o \preceq \sigma_t\}.$$

Intuitively, the strategies along the real path are going to be the ones that make the correct assumptions about the outcomes of the previous strategies. They will be finitely injured by a higher priority strategies, and will succeed in fulfilling their requirement. We are now going to prioritize the strategies so that the strategies along the true path are finitely injured.

**Priority order.** Let  $\Lambda = \{\infty, \mathbf{f}\}$  be the set of possible outcomes. Let us provide this set with an order of priority  $(\Lambda, <_p)$  by considering that  $\infty <_p \mathbf{f}$ , which means that the outcome  $\infty$  has priority over  $\mathbf{f}$ . This order will induce a total order  $(\mathcal{T}, <_p)$  (and therefore a total order of priority on the strategies) as follows:  $\sigma <_p \tau$  if  $\sigma \preceq \tau$ , or if  $i$  is the first position where the two strings differ, and  $\sigma(i) <_p \tau(i)$ . A visual example of this order is given in Figure 5.5. Note that, unlike finite injury priority methods, the strategies are generally below an infinite number of higher priority strategies. For example, the strategy  $R_{\infty \mathbf{f} \mathbf{f} \mathbf{f}}$  is below  $N_\epsilon$ ,  $R_\infty$ ,  $S_{\infty \mathbf{f}}$ ,  $N_{\infty \mathbf{f} \mathbf{f}}$ ,  $R_{\infty \mathbf{f} \mathbf{f} \infty}$ ,  $S_{\infty \mathbf{f} \mathbf{f} \infty \mathbf{f}}, \dots$

### Remark

One would be tempted to define a priority order such that each strategy only has a finite number of higher priority strategies, for example by defining the priorities by traversing the nodes of the tree. However, there is a problem:

Suppose that the strategy  $N_\epsilon$  has the outcome  $\infty$ . The strategy  $R_\mathbf{f}$  making the wrong assumption according to which the outcome of  $N_\epsilon$  has a finite outcome, it will be injured and reinitialized at each change of state of  $N_\epsilon$ . It will therefore potentially pose an infinite number of restraints, and will therefore prevent all lower priority strategies from functioning normally. It is therefore essential that all strategies along the true path of the tree take precedence over  $R_\mathbf{f}$ , in order to ignore its restraints. The strategy for  $R_\mathbf{f}$  must therefore be under an infinity of higher priority strategies.

**Verification.** Note first that by definition, the strategies along the real path  $P$  (strategies  $C_\sigma$  for  $\sigma \prec P$ ) are executed at an infinite number of steps. Although a strategy is generally below an infinite number of strategies, we will show that the strategies along the real path are injured finitely often, which is the necessary condition to satisfy them. The following lemma is a property that we generally expect from an argument with  $\Pi_2^0$  priority:

**Lemma 5.6.** Let  $P$  be the true path, and  $\alpha \prec P$ . Then,  $\alpha \leq_p \sigma_t$  for a co-finite number of steps  $t$ . ★

PROOF. By induction on the length  $n$  of  $\alpha$ . If  $n = 0$ , then  $\alpha = \epsilon$ , and by definition,  $\epsilon \leq_p \sigma_t$  for all  $t$ . Let  $n > 0$ . By induction hypothesis, there exists a threshold  $t_0$  such that  $\alpha \upharpoonright_{n-1} \leq_p \sigma_t$  for all  $t \geq t_0$ . Let  $S = \{t \geq t_0 : \sigma_t <_p \alpha\}$ . Suppose absurdly that  $S$  is infinite. Note that for all  $t \in S$ , as  $\alpha \upharpoonright_{n-1} \leq_p \sigma_t$  and  $\sigma_t <_p \alpha$  we necessarily have  $\alpha \upharpoonright_{n-1} \preceq \sigma_t$  with  $\sigma_t(n-1) <_p \alpha(n-1)$ . In other words,  $\sigma_t(n-1) = \infty$  and  $\alpha(n-1) = \mathbf{f}$  and  $\forall t \in S (\alpha \upharpoonright_{n-1})^\infty \preceq \sigma_t$ . Thus,  $(\alpha \upharpoonright_{n-1})^\infty \preceq P$ , contradicting the hypothesis  $\alpha \preceq P$ . This concludes the proof of the lemma. ■

Only the strategies along  $\sigma_t$  are executed in step  $t$ . Thus, for any strategy  $C_\alpha$  along the real path  $P$ , there is a threshold  $t_0$  after which only the lower priority strategies or strategies along the real path will be executed. We can therefore prove by induction on  $n$  that the strategy  $C_{P \upharpoonright_n}$  will only be injured finitely often, and will have the outcome  $P(n)$ . Any requirement being represented by a strategy along the real path, the requirements will all be satisfied. This concludes the proof of Theorem 5.2. ■

## Cappable degrees

We have seen two ways of creating incomparable Turing degrees. The first (see Proposition 4-8.1) consists in creating two sets simultaneously, while the second (see Proposition 4-8.2) starts from a non-computable set, and creates a second set of incomparable degree with the first. It is natural to ask whether it is possible, in the case of minimal pairs of c.e. degrees, to start from an arbitrary non-computable c.e. set, and complete it with another c.e. set to form a minimum pair. This is not the case, as proved by Yates [233] and Lachlan [128].

**Definition 5.7.** A c.e. degree  $\mathbf{a} > \mathbf{0}$  is *cappable* if there exists a degree  $\mathbf{b} > \mathbf{0}$  such that  $\mathbf{a}$  and  $\mathbf{b}$  form a minimal pair. Otherwise,  $\mathbf{a}$  is said to be *non-cappable*. ◇

Ambos-Pies et al. [5] obtained a surprising characterization of non-cappable degrees, using promptly simple sets.

**Definition 5.8.** A co-infinite c.e. set  $A$  is *promptly simple* if there exists a computable enumeration  $A_0 \subseteq A_1 \subseteq \dots$  and a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that

$$W_e \text{ infinite} \Rightarrow \exists^\infty x, s \ (x \in W_e[s] \setminus W_e[s-1] \wedge x \in A_{f(s)}).$$

In other words, a co-infinite set  $A$  is promptly simple if it is not only co-immune, but even more, this co-immunity must be achieved by making infinitely many elements enter  $A$  few time after they appear in  $W_e$ . A c.e. degree is *promptly simple* if it contains a promptly simple set.

**Theorem 5.9 (Ambos-Spies, Jockusch, Shore, et Soare [5])**

*The non-cappable degrees are precisely the promptly simple degrees.*

In particular, the proof showing that if a degree is not promptly simple, then it is cappable, is obtained via a variation of Theorem 5.2.

# Chapter 14

## Structure of the Turing degrees

The study of Turing degrees has been carried out in conjunction with that of its *structure*, as a partial order. Gerald E. Sacks is undoubtedly one of the main protagonists of this adventure.

Sacks began studying engineering at Cornell University in his youth, which he interrupted mid-term to enlist in the army for three years. It was at this time that he got his hands on a copy of *Introduction to Metamathematics* by Kleene, which fascinated him [34]. On his return to civilian life, he then oriented the rest of his studies towards mathematics. Barkley Rosser, an eminent logician who studied with Kleene and Church, agrees to take him on as a doctoral stu-



Gerald Sacks, 1933–2019

dent. Sacks will become in the sixties one of the pioneers of modern Computability Theory. He will participate in particular as we will see in the first studies on the structure of Turing degrees, and besides his work, he will become famous for the large number of his students who will become leading logicians, among whom we can quote Harvey and Sy Friedman, Léo Harrington, Richard Shore, Théodore Slaman and Stephen Simpson. We will see that the last three members of this list took a very active part in the study of the structure of Turing degrees.

Let's pose without further ado the vocabulary we will be using.

### Notation

$(\mathcal{D}, \leq)$  denotes the partial order structure of Turing degrees.

We will write  $\mathbf{a} \leq \mathbf{b}$  for two degrees  $\mathbf{a}, \mathbf{b} \in \mathcal{D}$  if  $A \leq_T B$  for any element  $A \in \mathbf{a}$  and any element  $B \in \mathbf{b}$ , and we will write  $\mathbf{a} < \mathbf{b}$  if  $\mathbf{a} \leq \mathbf{b}$  and  $\mathbf{b} \not\leq \mathbf{a}$ . In order to be completely clear, let us recall the vocabulary of use of partial orders: given a partially ordered set  $A$  and a subset  $B \subseteq A$ , an *upper bound* (resp. *lower bound*) of  $B$  is an element of  $A$  greater (resp. smaller) than all the elements of  $B$ . An upper bound (resp. lower bound) of  $B$  is *minimal* (resp. *maximal*) if no other upper bound of  $B$  is smaller than it (resp. no other lower bound of  $B$  is greater than it). Finally a least upper bound (resp. greatest lower bound) of  $B$  is a *upper bound* (resp. *lower bound*) if it is smaller than any upper bound of  $B$  (resp. greater than any lower bound of  $B$ ). It is easy to show that a least upper bound (resp. greatest lower bound) when it exists is unique.

## 1. Minimal degrees

One of the first results obtained on the structure of Turing degrees concerns the initial segments of  $\mathcal{D}$ , and in particular the existence of so-called *minimal* degrees:

**Definition 1.1.** A Turing degree  $\mathbf{d}$  is *minimal* if it is different from  $\mathbf{0}$  — the computable degree — and if there is no degree  $\mathbf{e}$  such that  $\mathbf{0} < \mathbf{e} < \mathbf{d}$ .  $\diamond$

In other words a set  $X \in 2^{\mathbb{N}}$  is of minimal degree if it is not computable and if for any functional  $\Phi$  such that  $\Phi(X, n) \downarrow \in \{0, 1\}$  for all  $n$ , either the set  $\{n : \Phi(X, n) \downarrow = 1\}$  is computable, or it is able to compute back  $X$ .

The proof of the existence of a minimal degree is one of the first uses of forcing in Computability Theory, via Sacks forcing, which we saw in Section 11-3.

### 1.1. Existence

The existence of minimal degrees, due to Spector [214], was initially made by forcing with *uniform* computable f-trees, that is to say f-trees  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that for any size  $n$  there exists a unique string  $\tau_n$  such that for any string  $\sigma$  of size  $n$  and any  $i \in \{0, 1\}$  we have  $T(\sigma i) = T(\sigma) i \tau_n$ . We can also see the paths of these f-trees as being all the possibilities of completion of a set  $X$  on which an infinity of bits are not specified.

Spector's restriction is of interest for the more general study of initial segments in Turing degrees, but for minimal degrees (i.e., initial segments of size 2), Shoenfield [197] noticed that computable Sacks forcing, easier to handle, is sufficient.

**Definition 1.2.** Let  $\Gamma$  be a Turing functional. Two strings  $\sigma, \tau \in 2^{<\mathbb{N}}$  form a  $\Gamma$ -split if there is an integer  $n \in \mathbb{N}$  such that  $\Gamma^\sigma(n) \downarrow \neq \Gamma^\tau(n) \downarrow$ . An f-tree  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  is  $\Gamma$ -splitting if for all  $\sigma \in 2^{<\mathbb{N}}$ , the strings  $T(\sigma 0)$  and  $T(\sigma 1)$  form a  $\Gamma$ -split. An f-tree  $T$  avoids  $\Gamma$ -splitting if no pair of strings  $\sigma, \tau \in \text{Im } T$  forms a  $\Gamma$ -split.  $\diamond$

For this section, we will consider the functionals  $\Gamma$  as being partial functions from  $2^{\mathbb{N}}$  to  $2^{\mathbb{N}}$ . In this particular context we will then denote by  $\text{dom } \Gamma$  — the domain of definition of  $\Gamma$  — the class  $\{X \in 2^{\mathbb{N}} : \forall n \Gamma(X, n) \downarrow \in \{0, 1\}\}$ . Given  $X \in \text{dom } \Gamma$ , we will write  $\Gamma(X)$  for the set  $\{n \in \mathbb{N} : \Gamma(X, n) \downarrow = 1\}$ , and we will speak of the totality of  $\Gamma$  with respect to  $2^{\mathbb{N}}$ , and not with respect to its inputs for a fixed oracle.

The key lemma in the construction of a minimal degree says that for any functional  $\Gamma$  and any computable f-tree, there exists a computable sub-tree on which  $\Gamma$  is everywhere defined and injective, or on which  $\Gamma$  is a constant function, restricted to its domain of definition in the sub-f-tree.

**Lemma 1.3.** For every computable f-tree  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  and every Turing functional  $\Gamma$ , there exists a computable sub-f-tree  $S$  of  $T$  which is either  $\Gamma$ -splitting, or avoids  $\Gamma$ -splittings.  $\star$

PROOF. Two cases arise.

Case 1: there exists a string  $\sigma \in 2^{<\mathbb{N}}$  such for all  $\rho, \tau \succeq \sigma$ ,  $T(\rho)$  and  $T(\tau)$  do not form a  $\Gamma$ -split. Let  $S$  be the computable sub-tree defined by  $S(\mu) = T(\sigma\mu)$ . Then,  $S$  is sub-tree of  $T$  avoiding  $\Gamma$ -splittings.

Case 2: for any string  $\sigma \in 2^{<\mathbb{N}}$ , there are extensions  $\rho, \tau \succeq \sigma$  such that  $T(\rho)$  and  $T(\tau)$  form a  $\Gamma$ -split. We then compute  $S$  as follows. We define  $S(\epsilon) = T(\epsilon)$ . Suppose we have computed  $S(\sigma) = T(\mu)$  for strings  $\sigma, \mu \in 2^{<\mathbb{N}}$ . We then look for strings  $\rho_0, \rho_1 \succeq \mu$  such that  $T(\rho_0)$  and  $T(\rho_1)$  form a  $\Gamma$ -split. By hypothesis, the search always succeeds. We then define  $S(\sigma 0) = T(\rho_0)$  and  $S(\sigma 1) = T(\rho_1)$ . Note that since  $T(\rho_0)$  and  $T(\rho_1)$  form a  $\Gamma$ -split, in particular, these two strings are incomparable. Thus,  $S$  is indeed an f-tree, and  $\text{Im } S \subseteq \text{Im } T$ . By construction,  $S$  is  $\Gamma$ -splitting.  $\blacksquare$

The interest of obtaining  $\Gamma$ -splitting or avoiding  $\Gamma$ -splitting f-trees is found in the following lemma.

**Lemma 1.4.** Let  $T$  be a computable f-tree and  $\Gamma$  a functional.

- (1) If  $T$  avoids  $\Gamma$ -splittings then for all  $X \in \text{dom } \Gamma \cap [T]$  the element  $\Gamma(X)$  is computable.
- (2) If  $T$  is  $\Gamma$ -splitting then for all  $X \in \text{dom } \Gamma \cap [T]$  we have  $X \leq_T \Gamma(X)$ .

★

PROOF. (1) Suppose that  $T$  avoids  $\Gamma$ -splittings. Let  $X \in \text{dom } \Gamma \cap [T]$ .

Then, to know the  $n$ -th bit of  $\Gamma(X)$  it suffices to search  $\sigma \in \text{Im } T$  such that  $\Gamma(\sigma, n) \downarrow = i$  for  $i \in \{0, 1\}$ . The  $n$ -th bit of  $\Gamma(X)$  is then equal to  $i$ .

- (2) Suppose that  $T$  is  $\Gamma$ -splitting. Let  $X \in \text{dom } \Gamma \cap [T]$ . Let  $Y = \Gamma(X)$ . To compute  $X$  from  $Y$  we proceed as follows: as  $T(0)$  and  $T(1)$  form a  $\Gamma$ -split, there exists  $i \in \{0, 1\}$  such that  $\Gamma(T(i), n) \neq Y(n)$  for some  $n$ . We can find  $i$  in a computable way in  $Y$ . The correct prefix of  $X$  is then necessarily  $T(1 - i)$ . Suppose we have computed a prefix  $\sigma \prec X$  and a string  $\tau$  such that  $\sigma = T(\tau)$ . As  $T(\tau 0)$  and  $T(\tau 1)$  form a  $\Gamma$ -split, there exists  $i \in \{0, 1\}$  such that  $\Gamma(T(\tau i), n) \neq Y(n)$  for some  $n$ . We can find  $i$  in a computable way in  $Y$ . The correct prefix of  $X$  is then necessarily  $T(\tau(1 - i))$ . By proceeding in this way, we then compute larger and larger prefixes of  $X$  from  $Y = \Gamma(X)$ . ■

We now have all the necessary ingredients to prove the existence of minimal degrees.

**Theorem 1.5 (Spector [214])**

*Any sufficiently generic set for Sacks forcing is of minimal degree.*

PROOF. Let  $(\mathbb{P}, \leq)$  be computable Sacks forcing. Let us note first that, according to Exercize 11-3.3, if  $G \in 2^{\mathbb{N}}$  is sufficiently generic for this forcing then it is not computable.

Let  $\Gamma$  be a Turing functional. According to Lemma 1.3 the set of conditions  $c \in \mathbb{P}$  such that  $c$  is  $\Gamma$ -splitting or  $c$  avoids  $\Gamma$ -splittings is dense. According to Lemma 1.4, in the first case, for all  $G \in [c]$  the functional  $\Gamma$  is defined on  $G$  and  $\Gamma(G) \geq_T G$ . In the second case, according to Lemma 1.4, for all  $G \in [c]$ , if the functional  $\Gamma$  is defined on  $G$  then  $\Gamma(G)$  is computable.

So if  $G$  is sufficiently generic for Sacks forcing, it is of minimal degree. ■

**Corollary 1.6**

*There is a perfect class of sets, all of distinct minimal degrees.*

PROOF. We can first show that there exists a perfect tree of minimal degrees. It suffices to proceed as in the proof of Theorem 8-5.1, in order to “duplicate” the construction of a minimal degree. With the forcing formalism, let  $(D_n)_{n \in \mathbb{N}}$  be a sequence of dense sets of Sacks forcing conditions, sufficient to force a set to be of minimal degree. We choose  $c_\epsilon \in D_0$ , then for any string  $\sigma$  of size  $n$ , assuming  $c_\sigma \in D_n$  defined, we define  $c_{\sigma 0} \leq c_\sigma$  and  $c_{\sigma 1} \leq c_\sigma$  in such a way that  $[c_{\sigma 0}] \cap [c_{\sigma 1}] = \emptyset$ . We can also make sure that the first branching node of  $c_{\sigma i}$  strictly extends the first branching node of  $c_\sigma$  for  $i \in \{0, 1\}$ . The final tree is given by the set of strings  $\tau$  such that there is  $X \in 2^\mathbb{N}$  for which  $\tau \in \bigcap_{\sigma \prec X} c_\sigma$ .

Once we have a perfect tree containing only minimal degrees, we can refer to Exercize 8-5.4 to extract from it a perfect subtree containing only pairwise incomparable degrees. ■

## 1.2. Computation of a minimal degree

A fine analysis of the level of effectiveness necessary to carry out Theorem 1.5 shows that there exists a  $\emptyset''$ -computable minimal degree. It is in fact possible to considerably improve this result, by noting that the use of computable  $f$ -trees is not absolutely necessary:

### C.e. tree

A c.e.  $f$ -tree is a partial function  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that for any  $\sigma$  such that  $T(\sigma) \downarrow$ , either  $T(\sigma 0) \uparrow$  and  $T(\sigma 1) \uparrow$ , or  $T(\sigma 0) \downarrow \succeq T(\sigma)$  and  $T(\sigma 1) \downarrow \succeq T(\sigma)$  with  $T(\sigma 0)$  and  $T(\sigma 1)$  incomparable.

The use of c.e.  $f$ -trees which are  $\Gamma$ -splitting or avoid  $\Gamma$ -splittings, allows to make constructions of minimal degrees requiring less computational power. It is then a question of effective constructions which do not fall strictly speaking any more into the scope of forcing. We list, without proving them, three important results which use this new type of tree to more easily compute sets of minimal degrees. By “more easily” it is necessary to understand “with little computational power”, the proofs being on the contrary more complex ...

### Theorem 1.7 (Sacks [184])

*There is a minimal degree under  $\emptyset'$ .*

Note that according to Proposition 7-4.7, Sacks' result implies the existence of a minimal hyperimmune degree. This result was subsequently improved by Yates [234], and independently by Cooper [39]:

**Theorem 1.8 (Yates [234], Cooper [39])**

*Let  $X$  be a non-computable c.e. set, then  $X$  computes a set  $G$  of minimal degree.*

Note that a minimal degree can never be c.e.: a sophisticated use of the famous priority method which we saw in Chapter 13, makes it possible to show that any non-computable c.e. set  $A$  compute another non-computable c.e. set  $B$  which does not compute it  $A$  (see Theorem 5.1 for the result in all its generality).

Finally, Groszek and Slaman showed that any PA degree could compute a minimal degree, via the following remarkable result.

**Theorem 1.9 (Groszek et Slaman [77])**

*Any PA degree computes a minimal degree. More precisely, there exists a non-empty  $\Pi_1^0$  class whose members are either of minimal degree, or compute a non-computable c.e. degree.*

**Exercise 1.10. (★)** Show that any non-empty  $\Pi_1^0$  class contains a set of the same degree as a c.e. set. Deduce that there is no non-empty  $\Pi_1^0$  class containing only elements of minimal degrees.  $\diamond$

The possibility of constructing “complex” minimal degrees has also been well studied. The main result in this direction is as follows.

**Theorem 1.11 (Kumabe [124])**

*There is a minimal degree which is also DNC.*

Kumabe first showed the existence of a minimal DNC degree, in a very complex article, which was never published. The proof was then reworked by Kumabe and Lewis [124], and the presentation was subsequently simplified by Khan and Miller [110], who rewrote it under the formalism of forcing, via *Forcing with Bushy Trees*. The following exercise shows that a minimal degree on the other hand can never be  $\text{DNC}_2$ .

**Exercise 1.12. (★)** Show that there exists a non-empty  $\Pi_1^0$  class of sets  $X \oplus Y$  such that  $X$  is of PA degree and  $Y$  of PA degree relative to  $X$ . Deduce that no PA set is of minimal degree (the reader can consult Proposition 24-2.4 for an iteration of this result).  $\diamond$

**1.3. Relativization: minimal cover**

The construction of a minimal degree with the forcing on f-trees is relativized to any Turing degree in the following sense.

**Definition 1.13.** Let  $\mathbf{a}$  and  $\mathbf{b}$  be Turing degrees. We say that  $\mathbf{b}$  is a *minimal cover* of  $\mathbf{a}$  if  $\mathbf{b} > \mathbf{a}$  and if there is no degree  $\mathbf{c}$  such that  $\mathbf{a} < \mathbf{c} < \mathbf{b}$ .  $\diamond$

In other words,  $\mathbf{b}$  is a minimal cover of  $\mathbf{a}$  if it is a minimal element in the cone of Turing degrees strictly above  $\mathbf{a}$ . The relativization of Theorem 1.5 shows the following theorem.

**Theorem 1.14**

*Any Turing degree has minimal cover.*

PROOF. Let  $A \subseteq \mathbb{N}$  be any set. The objective is to build a set  $B >_T A$  such that any set  $C$  computable by  $B$  is either  $A$ -computable, or such that  $A \oplus C \geq_T B$ . Thus, if  $B \geq_T C >_T A$  we will have  $C \geq_T B$ .

It suffices to consider a variant of Sacks forcing, for which our  $f$ -trees are this time  $A$ -computable, and such that each of their paths computes  $A$ . We could, for example, restrict ourselves to  $A$ -computable  $f$ -trees such that  $A$  is encoded in the even bits of each path of the  $f$ -tree.

It suffices then to repeat the proof of Theorem 1.5 with this new partial order, noting the following difference: given a  $\Gamma$ -splitting  $f$ -tree  $T$ , for all  $X \in [T]$  we now need  $A$  in order to find  $X$  starting from  $\Gamma(X)$ : indeed we need the knowledge of  $T$ . This is the reason why we will have  $A \oplus \Gamma(X) \geq_T X$  and not  $\Gamma(X) \geq_T X$ .  $\blacksquare$

Note that a minimal cover  $\mathbf{b}$  of  $\mathbf{a}$  does not exclude the existence of degrees  $\mathbf{c} < \mathbf{b}$  incomparable with  $\mathbf{a}$ . This leads us to define a stronger notion of cover:

**Definition 1.15.** Let  $\mathbf{a}$  and  $\mathbf{b}$  be Turing degrees. We say that  $\mathbf{b}$  is a *strong minimal cover* of  $\mathbf{a}$  if  $\mathbf{b} > \mathbf{a}$  and if for any Turing degree  $\mathbf{c} < \mathbf{b}$ , we have  $\mathbf{c} \leq \mathbf{a}$ .  $\diamond$

As noted in the proof of Theorem 1.14, the relativized version of Theorem 1.5 does not prove the existence of a strong minimal cover for any Turing degree, and for good reason: some degrees do not admit any strong minimal cover, although many will.

Ishmukhametov [94] has established an elegant characterization of the c.e. degrees admitting a strong minimal cover.

**Theorem 1.16 (Ishmukhametov [94])**

*A c.e. set  $A \subseteq \mathbb{N}$  admits a strong minimal cover iff any  $A$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is bounded for  $n$  sufficiently large by the function  $n \mapsto$*

$$\min \{s \in \mathbb{N} : \emptyset'[s] \upharpoonright_n = \emptyset' \upharpoonright_n\}.$$

A general characterization of the degrees admitting a strong minimal cover is for the moment unknown, although many partial results have been established (see Lewis [142]).

## 2. Nature of $\mathcal{D}$

What does the partial order  $(\mathcal{D}, \leq)$  look like? Regarding its size first, we have seen in this book several constructions of perfect trees where each path is in a different Turing degree (see for example Exercise 7-5.8, Exercise 8-5.3 or Exercise 8-5.4). This gives us an injection of  $2^{\mathbb{N}}$  into  $\mathcal{D}$ . Using the axiom of choice we can choose a representative in each Turing degree, which gives an injection of  $\mathcal{D}$  into  $2^{\mathbb{N}}$ . The cardinality of  $\mathcal{D}$  is therefore  $|2^{\mathbb{N}}|$ , that of  $2^{\mathbb{N}}$ . Note that we cannot necessarily choose a representative in each Turing degree if we do not have the axiom of choice. The fact remains that “morally”,  $\mathcal{D}$  is of cardinality  $|2^{\mathbb{N}}|$ .

Given an element  $\mathbf{a} \in \mathcal{D}$ , the set of elements below  $\mathbf{a}$  is at most countable since a set can only compute a countably many elements. On the other hand, the cardinality of the elements above  $\mathbf{a}$  is that of  $2^{\mathbb{N}}$ : given  $A \in \mathbf{a}$  we can easily create a perfect tree whose paths are all of the form  $A \oplus X$  for  $X \in 2^{\mathbb{N}}$ , and all in different Turing degrees.

The use of the Turing join leads us to the following consideration: given two sets  $A, B$ , the set  $A \oplus B$  computes both  $A$  and  $B$ , and any set computing at the same time  $A$  and  $B$  computes  $A \oplus B$ . In terms of degrees, this implies that every pair of degrees  $\mathbf{a}, \mathbf{b}$  has a least upper bound. There is therefore a minimum of structure in this partial order, for which we introduce the following concept.

**Definition 2.1.** A *lattice* is a partially ordered set in which any pair of elements  $a, b$  has a least upper bound noted  $a \cup b$  and a greatest lower bound noted  $a \cap b$ . An *upper (resp. lower) semilattice* is an ordered set for which every pair of elements has a least upper bound (resp. greatest lower bound).  $\diamond$

The paragraph preceding this definition leads to the following theorem, appearing in the founding article of the study of the structure of Turing degrees.

**Theorem 2.2 (Kleene et Post [117])**

$(\mathcal{D}, \leq)$  is an upper semilattice of cardinality  $|2^{\mathbb{N}}|$ , with a smallest but no largest element, such that each element has a most countably many

elements below it, and a set of cardinality  $|2^{\mathbb{N}}|$  of elements above it.

We sometimes base the vocabulary of Turing degrees on that of the sets they contain: given two degrees  $\mathbf{a}, \mathbf{b}$  we say that the least upper bound  $\mathbf{a} \cup \mathbf{b}$  of  $\mathbf{a}$  and  $\mathbf{b}$  is the *join* of  $\mathbf{a}, \mathbf{b}$ . Note that in an upper semilattice, any finite sequence of elements also admits a least upper bound. In particular for a finite set of degrees  $\mathbf{a}_1, \dots, \mathbf{a}_n$  we will denote it  $\mathbf{a}_1 \cup \dots \cup \mathbf{a}_n$  and it will be the degree of the join  $A_1 \oplus \dots \oplus A_n$  for any representatives  $A_i \in \mathbf{a}_i$ .

What happens for countable sets of degrees? The following theorem implies that if such a set is closed under join — i.e.  $\mathbf{a} \cup \mathbf{b}$  is in our set for all  $\mathbf{a}, \mathbf{b}$  in our set — and has no maximal element, then it does not have any least upper bound.

**Theorem 2.3 (Sacks [187])**

*Any countable set of degrees closed under join and without maximal element has minimal upper bounds in quantity  $|2^{\mathbb{N}}|$ .*

PROOF SKETCH. First note that an upper bound of a set of integers of degrees  $(\mathbf{a}_n)_{n \in \mathbb{N}}$ , is also an upper bound of  $\mathbf{a}_0 \leq \mathbf{a}_0 \cup \mathbf{a}_1 \leq \mathbf{a}_0 \cup \mathbf{a}_1 \cup \mathbf{a}_2 \leq \dots$ . As  $(\mathbf{a}_n)_{n \in \mathbb{N}}$  has no maximal element and is closed under join, we can therefore consider without loss of generality that our set of degrees is such that  $\mathbf{a}_n < \mathbf{a}_{n+1}$ . For all  $n$ , let  $A_n$  be a representative of  $\mathbf{a}_n$ .

It is now enough to elaborate on Sacks forcing (see Section 1) allowing to create the minimal cover of a degree. We start with an  $A_0$ -computable f-tree whose paths all compute  $A_0$ . A forcing condition will be an  $A_n$ -computable f-tree whose paths all compute  $A_n$  (for a certain  $n$ ). We extend such a forcing condition  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  to the f-tree  $Q$  such that  $\text{Im } Q \subseteq \text{Im } T$  consists of the paths which encode  $A_{n+1} \oplus X$  for all  $X \in 2^{\mathbb{N}}$ . Formally,  $Q(\sigma i) = T(A_{n+1} \upharpoonright_{|\sigma i|} \oplus \sigma i)$  for any string  $\sigma$  and any  $i \in \{0, 1\}$ . As  $A_{n+1} \geq_T A_n$ , then  $A_{n+1}$  can compute  $T$  and thus find  $A_{n+1} \oplus X$  from the path of  $Q$  which encodes  $A_{n+1} \oplus X$  in  $T$ .

The way of doing a minimal cover does not change, and the technique described in Section 1 applies in the same way. The resulting generic set  $G$  will compute each set  $A_n$ , and will be such that anything that is computed by  $G$  and which can compute each set  $A_n$ , can also compute  $G$ .

To obtain minimal upper bound in quantity  $|2^{\mathbb{N}}|$ , one can build a perfect tree of minimal upper bound by subdividing the construction in two, then each substructure in two, etc., as in the proof of Theorem 8-5.1. ■

A countable set of degrees closed under join and with no maximum element therefore always has two distinct minimal upper bounds. We can therefore deduce the following corollary.

**Corollary 2.4**

*A countable set of degrees closed under join and with no maximal element never has a least upper bound.*

Note that for a sequence of sets  $(A_n)_{n \in \mathbb{N}}$ , the set  $\bigoplus_{n \in \mathbb{N}} A_n$  (see Definition 10-3.24) is not in general a minimal upper bound, as shown by the following elegant result.

**Theorem 2.5 (Enderton et Putnam [55], Sacks [189])**

*There exists a minimal upper bound of  $(\emptyset^{(n)})_{n \in \mathbb{N}}$  whose double jump is in the same Turing degree as that of  $\bigoplus_n \emptyset^{(n)}$ .*

PROOF SKETCH. For a direction, it suffices to notice that the double jump of any upper bound of  $(\emptyset^{(n)})_{n \in \mathbb{N}}$  allows to compute  $\bigoplus_n \emptyset^{(n)}$ . Let  $B$  be an upper bound and let  $f$  be a computable function such that  $f(X') = X$  for any  $X$  (see Exercize 4-6.4 for more details on such a function). Using the double jump of  $B$  we look for a functional code  $e_1$  such that  $f(\Phi_{e_1}(B)) = \emptyset$ , then a functional code  $e_2$  such that  $f(\Phi_{e_2}(B)) = \Phi_{e_1}(B)$ , etc.

For the other direction, it suffices to see that the construction of an upper bound of  $(\emptyset^{(n)})_{n \in \mathbb{N}}$  is effective using  $\bigoplus_n \emptyset^{(n)}$ , and forces at each step a functional  $\Phi_e$  to be partial or else total on all the elements of the tree considered: one thus does not only compute the resulting generic  $G$ , but one can also determine the set of the codes of total functionals on  $G$ . The double jump of  $G$  reduces to this set (see Exercize 5-7.2). ■

To end this section, we answer the question that may have tormented the reader since the beginning of this chapter: is the structure  $(\mathcal{D}, \leq)$  a lattice? We will see that it is not, and for this we introduce the notion of exact pair.

**Definition 2.6.** The degrees  $\mathbf{a}, \mathbf{b}$  form a *exact pair* for a set of degrees  $C \subseteq \mathcal{D}$  if  $\mathbf{a}$  and  $\mathbf{b}$  each bound all the degrees of  $C$ , and if each degree below both  $\mathbf{a}$  and  $\mathbf{b}$  is also bounded by a degree of  $C$ . ♦

**Theorem 2.7**

*Any countable set of degrees  $C$  closed under join admits an exact pair.*

PROOF. Let  $(\mathbf{a}_n)_{n \in \mathbb{N}}$  be a set of Turing degrees closed under join. Let  $A_n$  be a representative of  $\mathbf{a}_n$ . The idea is to construct two sets  $G_0 = \bigoplus_n X_n^0$  and  $G_1 = \bigoplus_n X_n^1$  such that each column  $X_n^i$  for  $i \in \{0, 1\}$  is equal to the set  $A_n$ , except for a finite number of bits. It is clear that such sets  $G_0, G_1$  allow us to compute all the  $A_n$ . We must now build them via an adapted

forcing in such a way that if  $G_0$  and  $G_1$  compute the same set, then this set is computable by  $A_0 \oplus A_1 \oplus \cdots \oplus A_m$  for a certain  $m$ .

Our forcing conditions are made up of triplets  $(\sigma_0, \sigma_1, n)$  where  $\sigma_0, \sigma_1 \in 2^{<\mathbb{N}}$  and  $n \in \mathbb{N}$ . The parameter  $n$  is used to control the possible extensions of our conditions. We have  $(\sigma_0, \sigma_1, n) \succeq (\tau_0, \tau_1, m)$  for two forcing conditions if  $\sigma_0 \preceq \tau_0$ , if  $\sigma_1 \preceq \tau_1$ , if  $n \leq m$  with the restriction that for all  $\langle k, a \rangle$  such that  $|\sigma_i| \leq \langle k, a \rangle < |\tau_i|$  for  $k \leq n$ , we must have  $\tau_i(\langle k, a \rangle) = A_k(a)$ . Given a set of conditions  $(\sigma_0^0, \sigma_1^0, n_0) \succeq (\sigma_0^1, \sigma_1^1, n_1) \succeq (\sigma_0^2, \sigma_1^2, n_2) \succeq \dots$ , the generic set  $G_0$  will be the limit of  $\sigma_0^0 \preceq \sigma_0^1 \preceq \sigma_0^2 \preceq \dots$  and the generic set  $G_1$  will be the limit of  $\sigma_1^0 \preceq \sigma_1^1 \preceq \sigma_1^2 \preceq \dots$ . Note that the restriction on the possible extensions guarantees that as long as the sequence  $n_0 \leq n_1 \leq n_2 \leq \dots$  is unbounded, each  $n$ -th column of  $G_i$  will indeed be equal to  $A_n$  except for a finite number of bits.

For any pair of functionals  $\Phi_{e_0}, \Phi_{e_1}$ , we will force  $\Phi_{e_0}(G_0) = \Phi_{e_1}(G_1) = X$  implies  $X \leq_T A_0 \oplus \cdots \oplus A_n$  for some  $n$ . Note that if  $G_0$  and  $G_1$  bound the same degree, there necessarily exist two functionals such that  $G_0$  and  $G_1$  compute the same set in this degree. Such a construction therefore achieves our objectives.

Let  $(\sigma_0, \sigma_1, n)$  be a forcing condition and  $\Phi_{e_0}, \Phi_{e_1}$  be two functionals. We are looking for two strings  $\tau_0, \tau_1$  such that  $(\tau_0, \tau_1, n)$  is a valid extension of  $(\sigma_0, \sigma_1, n)$ , and such that  $\Phi_{e_0}(\tau_0, x) \downarrow \neq \Phi_{e_1}(\tau_1, x) \downarrow$  for some  $x$ . If this search is successful, then we take  $(\tau_0, \tau_1, n)$  as an extension of  $(\sigma_0, \sigma_1, n)$ . Otherwise it means that on all  $x$ , the computations  $\Phi_{e_0}(\tau_0, x)$  and  $\Phi_{e_1}(\tau_1, x)$ , when they halt, return the same value for any valid extension  $(\tau_0, \tau_1, n) \succeq (\sigma_0, \sigma_1, n)$ .

Note that by definition of what is a valid extension of  $(\sigma_0, \sigma_1, n)$ , it is possible to enumerate them using  $A_0 \oplus \cdots \oplus A_n$ . If for all  $x$ , there exists a valid extension  $(\tau_0, \tau_1, n)$  such that  $\Phi_{e_0}(\tau_0, x) \downarrow$  or  $\Phi_{e_1}(\tau_1, x) \downarrow$  then we can compute via  $A_0 \oplus \cdots \oplus A_n$  the unique element  $Z$  thus potentially computable by any generic  $G_0$  via  $\Phi_{e_0}$  or any generic  $G_1$  via  $\Phi_{e_1}$ . Otherwise at least one of the two computations  $\Phi_{e_0}(G_0, x)$  or  $\Phi_{e_1}(G_1, x)$  will be partial over a certain  $x$ .

So if  $G_0, G_1$  are sufficiently generic for this forcing, for any functional  $\Phi_{e_0}, \Phi_{e_1}$  we will have  $\Phi_{e_0}(G_0) = \Phi_{e_1}(G_1) = X$  implies  $X \leq_T A_0 \oplus \cdots \oplus A_n$  for some  $n$ . It follows that  $G_0, G_1$  is an exact pair for degrees  $(\mathbf{a}_n)_{n \in \mathbb{N}}$ . ■

We can now deduce that  $(\mathcal{D}, \leq)$  is not a lattice.

**Theorem 2.8 (Kleene et Post [117])**

*The upper semilattice  $\mathcal{D}$  is not a lattice. There are in particular degrees  $\mathbf{a}, \mathbf{b}$  which do not have a greatest lower bound.*

PROOF. It suffices to consider a set of degrees  $\mathbf{c}_0 < \mathbf{c}_1 < \mathbf{c}_2 < \dots$  closed under join. This set of degrees therefore admits an exact pair  $\mathbf{a}, \mathbf{b}$ . Such an exact pair has no greatest lower bound since any degree under both  $\mathbf{a}$  and  $\mathbf{b}$  is also under a degree  $\mathbf{c}_n$  for some  $n$ . ■

### 3. Universality of $\mathcal{D}$

We see in this section that  $(\mathcal{D}, \leq)$  presents a certain universality, in the sense that *all partial orders can be embedded in  $\mathcal{D}$* , except those which cannot claim it for reasons of cardinality.

**Definition 3.1.** A partial order  $(A, \leq)$  *embeds* into  $(\mathcal{D}, \leq)$  if there is an injection  $f : A \rightarrow \mathcal{D}$  such that  $a \leq b$  iff  $f(a) \leq f(b)$ . ◇

The structure  $(\mathcal{D}, \leq)$  therefore contains in it all the partial orders which are not larger than it. This claim, which will be made precise, is actually a bit wrong: there is an open question on this subject, which we will mention shortly. Note that this does not necessarily inform us about the computational complexity of  $\mathcal{D}$ . Consider for example the partial computable order  $\leq_R \subseteq \mathbb{Q}^2 \times \mathbb{Q}^2$  defined by  $(p_1, p_2) \leq_R (q_1, q_2)$  if  $p_1 \leq q_1$  and  $p_2 \leq q_2$  for  $p_1, p_2, q_1, q_2 \in \mathbb{Q}$ . It is not very difficult to show that any countable partial order embeds into  $(\mathbb{Q}^2, \leq_R)$ . The construction of an embedding is done without difficulty, by constructing the injection in a greedy way, element by element, without ever violating at each finite stage the constraints of an embedding. The structure  $(\mathbb{Q}^2, \leq_R)$  is not computably complex, but it is sufficiently rich in terms of possibilities to contain all the partial orders.

We will use several times the existence of sets called *computably independent*.

**Definition 3.2.** A countable collection of sets  $(X_n)_{n \in \mathbb{N}}$  is *computably independent* if  $X_i \not\leq_T \bigoplus_{j \neq i} X_j$  for all  $i \in \mathbb{N}$ . ◇

The existence of computably independent sets does not present any particular difficulties, and we can refer to Exercize 10-3.25 to see that if  $G = \bigoplus_n G_n$  is a 1-generic set, then the sets  $(G_n)_{n \in \mathbb{N}}$  are computably independent.

### 3.1. Embeddings in $\mathcal{D}$

Let us immediately see what was announced, in the form of a first theorem.

**Theorem 3.3 (Sacks [187])**

*Any countable partial order embeds into the Turing degrees.*

PROOF. We saw in the introduction of this section that any countable partial order can be embedded in the structure  $(\mathbb{Q}^2, \leq_R)$  defined by  $(p_1, p_2) \leq_R (q_1, q_2)$  if  $p_1 \leq q_1$  and  $p_2 \leq q_2$ .

It suffices then to show that  $(\mathbb{Q}^2, \leq_R)$  embeds into  $(\mathcal{D}, \leq)$ . Let  $(X_n)_{n \in \mathbb{N}}$  be a sequence of computably independent sets, and let  $(a_n)_{n \in \mathbb{N}}$  be an enumeration of the elements of  $\mathbb{Q}^2$ . The embedding  $f$  assigns to the element  $a_n$  the Turing degree of the set  $\bigoplus_{a_m \leq_R a_n} X_m$ . We can easily verify  $a_n \leq_R a_m$  iff  $f(a_n) \leq f(a_m)$ . ■

Sacks subsequently sought to extend his result to larger partial orders. After all,  $(\mathcal{D}, \leq)$  admits  $2^{\mathbb{N}}$  for cardinality. We cannot of course expect any partial order of cardinality  $2^{\mathbb{N}}$  to be embedded in  $(\mathcal{D}, \leq)$ : if an element in a partial order has an uncountable quantity of predecessors, there is no hope of constructing an embedding of this order to  $\mathcal{D}$  since each element of  $\mathcal{D}$  has only a countable quantity below it. We must therefore respect this restriction, but are there others? We need here to anticipate a little on the ordinals which will be introduced in Chapter 27, and in particular on the ordinal  $\omega_1$ , the smallest uncountable infinite ordinal. Sacks obtained the following results.

**Theorem 3.4 (Sacks [185])**

*Any partial order with one of the following properties can be embedded in Turing degrees:*

1. *the order is of cardinality  $2^{\mathbb{N}}$  and each element has a finitely many predecessors;*
2. *the order is of cardinality  $|\omega_1|$  and each element has at most countably many predecessors;*
3. *the order is of cardinality  $2^{\mathbb{N}}$ , each element has at most countably many predecessors, as well as at most  $\omega_1$  successors.*

In particular, if we assume the continuum hypothesis, namely  $|\omega_1| = 2^{\mathbb{N}}$ , Sacks' theorem is optimal: any partial order of cardinality  $2^{\mathbb{N}}$ , and where each element has at most a countable quantity of predecessors, can embed into the Turing degrees. But, if we do not assume the continuum hypothesis, the question is still open.

**Question 3.5.** Can we embed into  $(\mathcal{D}, \leq)$  any partial order of cardinality  $|2^{\mathbb{N}}|$  where each element has at most countably many predecessors? ★

If we do not present here the proof of Sacks, we nevertheless see two ingredients, via the notions of chain and anti-chain.

**Definition 3.6.** A *chain* is a linearly ordered set of Turing degrees. An *anti-chain* is a set of pairwise incomparable Turing degrees. ◇

**Proposition 3.7 (Sacks [187]).**

- (1) Each countable chain can be extended in the Turing degrees. In particular, any maximal chain is of cardinality  $\omega_1$ .
- (2) Each anti-chain of cardinality less than  $|2^{\mathbb{N}}|$  can be extended in the Turing degrees. In particular, any maximal anti-chain is of cardinality  $|2^{\mathbb{N}}|$ . ★

PROOF.

- (1) If the chain has a largest element, we can consider its Turing jump. Otherwise, we can consider the degree of the Turing join of a representative of each of its elements.
- (2) Let  $D$  be the set of minimal degrees. By Corollary 1.6,  $D$  is of cardinality  $|2^{\mathbb{N}}|$ . Let  $C$  be an anti-chain of Turing degrees with cardinality less than  $|2^{\mathbb{N}}|$ . Each element of  $C$  has at most countably many elements below it. Thus, the downward closure of  $C$  has the same cardinality as  $C$ . There must therefore exist an element of  $\mathbf{d} \in D$  which is not computed by any element of  $C$ . Since  $\mathbf{d}$  is a minimal degree, it cannot bound any element of  $C$ . We deduce that  $C \cup \{\mathbf{d}\}$  is an anti-chain. ■

### 3.2. Extension of embeddings of $\mathcal{D}$

The notion of embedding can be considered weak, in particular because it does not say anything about the relations that the degrees maintain in the image of an embedding, with the degrees which are not in this image. One way to overcome this weakness is to consider an already existing embedding of a structure  $(C, \leq)$  towards Turing degrees, and to try to see to what extent this embedding can extend to an extension of the partial order on  $C$ . Such a thing is of course not always possible: if elements  $a_0, a_1, b \in C$  with  $a_0 < b, a_1 < b$  and  $a_0, a_1$  incomparable, are sent to degrees  $\mathbf{a}_0, \mathbf{a}_1$  and  $\mathbf{a}_0 \cup \mathbf{a}_1$ , then such an embedding cannot be extended to any upper bound  $c < b$  of  $a_0, a_1$ . For this, we introduce the notion of consistent extension.

**Definition 3.8.** Let  $C$  be an upper semilattice and let  $D \supseteq C$ . Then,  $D$  is a *consistent extension* of  $C$  if:

- (1) for  $a, b < d$  with  $a, b \in C$  and  $d \in D \setminus C$  we have  $a \cup b < d$ ;
- (2) no element of  $D \setminus C$  is under an element of  $C$ .

Note that since  $C$  is an upper semilattice,  $a \cup b \in C$  for all  $a, b \in C$ .  $\diamond$

**Theorem 3.9 (Kleene et Post [117])**

Let  $C$  be a finite upper semilattice and let  $D$  be a finite consistent extension of  $C$ . Then, any embedding  $f$  from  $(C, \leq)$  into  $(\mathcal{D}, \leq)$  can be extended to an embedding of  $(D, \leq)$  into  $(\mathcal{D}, \leq)$ .

PROOF. Let  $f$  be an embedding of  $(C, \leq)$  into  $(\mathcal{D}, \leq)$ . We denote by  $\mathbf{a}$  the image of  $a \in C$  by  $f$ . Let  $a \in D \setminus C$  be minimal in  $D \setminus C$ . Since  $D$  is a consistent extension then  $C$  can be partitioned into a list of elements  $(b_i)_{i \leq n}$  and  $(c_i)_{i \leq m}$  such that  $b_0 \cup \dots \cup b_n < a$ , such that  $a$  is incomparable with each  $c_i$  and such that  $b_0 \cup \dots \cup b_n$  is not above any  $c_i$ . It suffices then to construct a Turing degree  $\mathbf{a}$  such that  $\mathbf{b}_0 \cup \dots \cup \mathbf{b}_n < \mathbf{a}$  and such that  $\mathbf{a}$  is incomparable with each  $\mathbf{c}_i$ .

By using the fact that  $\mathbf{b}_0 \cup \dots \cup \mathbf{b}_n$  is not above any  $\mathbf{c}_i$ , we easily construct by finite extensions a degree  $\mathbf{d}$  such that  $\mathbf{d} \cup \mathbf{b}_0 \cup \dots \cup \mathbf{b}_n$  is not above any  $\mathbf{c}_i$  and such that  $\mathbf{b}_0 \cup \dots \cup \mathbf{b}_n$  is not above  $\mathbf{d}$ . The embedding is then extended by sending  $a$  to  $\mathbf{d} \cup \mathbf{b}_0 \cup \dots \cup \mathbf{b}_n$ . By minimality of the choice of  $a$  the set  $D - \{a\}$  is now a consistent extension of the closure of  $C \cup \{a\}$  in the upper semilattice. We can therefore start again until each element of  $D$  is assigned.  $\blacksquare$

We will see with Lemma 4.2 that the converse of the theorem works: it is necessary to be a consistent extension so that any embedding is extendible. The previous theorem can be extended:

**Theorem 3.10 (Sacks [185])**

Let  $C$  be a countable upper semilattice and let  $f$  be an embedding of  $(C, \leq)$  in  $(\mathcal{D}, \leq)$ . Let  $D$  be a consistent and countable extension of  $C$ . Then,  $f$  can be extended by an embedding of  $(D, \leq)$  into  $(\mathcal{D}, \leq)$ .

### 3.3. Initial segments of $\mathcal{D}$

Another way to strengthen the study of possible embeddings is to consider embeddings on *initial segments* of  $\mathcal{D}$

**Definition 3.11.** A *initial segment* of  $\mathcal{D}$  is a downward-closed set of degrees. A *final segment* is an upward-closed set of degrees.  $\diamond$

An embedding on an initial segment gives us complete information on all the degrees which are below those of the image of the embedding. For example the construction of a minimal degree indicates to us that the order  $a < b$  of two elements  $a, b$  can be embedded into an initial segment of  $\mathcal{D}$ . The existence of a minimal degree with a strong minimal cover (see Definition 1.15) indicates to us that the order  $a < b < c$  of three elements  $a, b, c$  can be embedded into an initial segment of  $\mathcal{D}$ . By elaborating on the construction of minimal degrees, Lachlan and Lebeuf obtained the following remarkable result.

**Theorem 3.12 (Lachlan et Lebeuf [132])**

*A countable partial order embeds into an initial segment of  $(\mathcal{D}, \leq)$  iff it is an upper semilattice with a smallest element.*

The proof of Lachlan and Lebeuf is done little by little, the most difficult step being to show it for any finite upper semilattice with a smallest element. This is a non-trivial modification of the construction of minimal degrees. Consider for example the partial diamond order given by  $a \leq b_1, a \leq b_2, b_1, b_2 \leq c$  and  $b_1, b_2$  incomparable. We must then construct two minimal degrees  $\mathbf{b}_1, \mathbf{b}_2$ , such that  $\mathbf{b}_1 \cup \mathbf{b}_2 = \mathbf{c}$ , and such that  $\mathbf{b}_1, \mathbf{b}_2$  are *the only non-computable degrees* found under  $\mathbf{c}$ . Such a construction is based on a forcing with computable *uniform*  $f$ -trees, as explained in Section 1.1. One can for example build with such a forcing a set  $X = X_0 \oplus X_1$  such that  $X_0$  and  $X_1$  are incomparable and minimal, and such that everything which is computed by  $X$  is either under  $X_0$ , either under  $X_1$ , or can recompute  $X$ . The detailed proof can be consulted in [168] or [137].

Note that the theorem of Lachlan and Lebeuf gives a complete characterization of Turing ideals of the form  $\mathcal{D}(\leq \mathbf{a})$  for a certain degree  $\mathbf{a}$  (i.e., the set of elements which are found under  $\mathbf{a}$ ): these are the upper semilattices which are at most countable, having a smallest and a largest element.

We can push the theorem of Lachlan and Lebeuf a little further:

**Theorem 3.13 (Abraham et Shore [2])**

*A partial order of cardinality  $\omega_1$  is isomorphic to an initial segment of  $(\mathcal{D}, \leq)$  iff it is an upper semilattice with a smallest element, and in which each element has a most countably many predecessors.*

In particular if one makes the assumption of the continuum hypothesis, that completely characterizes exactly the possible initial segments of the

partial order  $(\mathcal{D}, \leq)$ . If we do not make the assumption of the continuum hypothesis, then things get complicated:

**Theorem 3.14 (Groszek et Slaman [76])**

*There is a model of ZFC in which there is an uncountable partial order with a smallest element and for which each element has a finite number of predecessors, and which be embedded into an initial segment of  $(\mathcal{D}, \leq)$ .*

## 4. First-order theory of $\mathcal{D}$

We now discuss the complexity of  $\mathcal{D}$ . The question that interests us is the following: given a first-order statement which concerns the Turing degrees, can we decide whether the latter is satisfied or not? The language that we can use consists only of the relations  $\leq$ ,  $<$  or  $=$ , but it will be possible to extend this language to whatever is definable with  $\leq$ ,  $<$  or  $=$ . For example  $\mathbf{0}$ , the minimal degree, is definable as being the only degree which satisfies the formula  $F(\mathbf{x}) = \forall \mathbf{y} \mathbf{x} \leq \mathbf{y}$ . We can then for example express the existence of a minimal degree as follows:  $\exists \mathbf{x} (\mathbf{0} < \mathbf{x} \wedge \forall \mathbf{y} \mathbf{x} \leq \mathbf{y} (\mathbf{y} = \mathbf{0} \vee \mathbf{y} = \mathbf{x}))$ . The fact that  $\mathbf{0}$  can be defined by the formula  $F(\mathbf{x})$  makes it possible to get rid of it with an equivalent formula:

$$\exists \mathbf{z} (F(\mathbf{z}) \wedge \exists \mathbf{x} (\mathbf{z} < \mathbf{x} \wedge \forall \mathbf{y} \mathbf{x} \leq \mathbf{y} (\mathbf{y} = \mathbf{z} \vee \mathbf{y} = \mathbf{x}))).$$

It is of course longer, and we will therefore allow ourselves these language extensions. We will use in particular the function with two arguments  $\cup$  which gives us the join of two degrees, and which is also definable by the formula

$$F(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{x} \leq \mathbf{z} \wedge \mathbf{y} \leq \mathbf{z} \wedge \forall \mathbf{a} ((\mathbf{x} \leq \mathbf{a} \wedge \mathbf{y} \leq \mathbf{a}) \rightarrow \mathbf{z} \leq \mathbf{a}).$$

We easily verify that for all  $\mathbf{a}, \mathbf{b} \in \mathcal{D}$ , the degree  $\mathbf{a} \cup \mathbf{b}$  is the unique degree  $\mathbf{z}$  such that  $F(\mathbf{a}, \mathbf{b}, \mathbf{z})$ .

Now let's come back to our question: given a statement about Turing degrees, can we decide whether the latter is true or false? The question is *a priori* of great complexity if we approach it directly: a statement of the type  $\exists \mathbf{a}$  amounts to a statement of the type “does there exist a set  $X$  such that”. This is a second-order quantification, that is to say, relating not to integers, but to sets of integers. This kind of quantification will be discussed in detail in Part IV. The statements with quantifications on the integers being already undecidable, it is a safe bet that this is also the case for statements on Turing degrees. We can nevertheless show that if the formula is of the type  $\forall \exists$  or of the type  $\exists \forall$ , we can then decide whether it is true or false. This constitutes our first theorem on the subject:

**Theorem 4.1 (Lerman [137] (théorème 4.4) et Shore [198])**

*The  $\Pi_2^0$  theory of  $(\mathcal{D}, \leq)$ , i.e., the set of  $\Pi_2^0$  statements true in  $(\mathcal{D}, \leq)$ , is decidable.*

To prove the previous theorem, it is essentially enough to see that the converse of Theorem 3.9 is true:

**Lemma 4.2.** Let  $C$  be a finite upper semilattice and let  $D$  be a finite extension of  $C$ . Then,  $D$  is a consistent extension iff any embedding  $f$  of  $(C, \leq)$  into  $(\mathcal{D}, \leq)$  can be extended to an embedding of  $(D, \leq)$  into  $(\mathcal{D}, \leq)$ . ★

PROOF. Theorem 3.9 gives us a direction of the lemma. Suppose now that  $D$  is not a consistent extension of  $C$ . If an element of  $d \in D$  is such that  $a, b \leq d$  but  $a \cup b \not\leq d$ , it suffices to consider an embedding which associates  $\mathbf{a} \cup \mathbf{b}$  with  $a \cup b$ . It will then be impossible to extend such an embedding to  $D$ . Suppose now that a degree of  $D$  is below a degree of  $C$ . By Theorem 3.12, there exists an embedding of  $C$  on an initial segment of  $\mathcal{D}$ . Here again, such an embedding cannot be extended to a degree lower than a degree of  $C$ . ■

We now have the necessary ingredient to show Theorem 4.1:

PROOF OF THEOREM 4.1. Consider a statement of the form

$$\forall c_1, \dots, \forall c_n \exists d_1, \dots, d_m F(c_1, \dots, c_n, d_1, \dots, d_m),$$

where  $F$  is a Boolean combination of atomic formulas. Let  $C$  be the finite set of the possible models of upper semilattices generated by  $c_i$  and compatible with the conditions given by  $F$ . If  $C$  is empty, then the formula is not satisfied without even considering the part of  $F$  mentioning the  $d_i$ . Otherwise, for all the models of  $C$ , we check if this model can be completed in a way compatible with the conditions given by  $F$ , and in such a way that the  $d_i$  are a consistent extension. If this is the case the formula is true, otherwise it is false. ■

This is the limit of what is decidable. Lachlan [129] showed that the  $\Pi_3^0$  theory was no longer so.

**Theorem 4.3 (Schmerl (see Corollary 4.6 of [137]))**

*The  $\Pi_3^0$  theory of  $(\mathcal{D}, \leq)$  is undecidable.*

How complex is the theory of  $(\mathcal{D}, \leq)$ ? Can it be decided, for example, using the Turing jump, or even using the disjoint union of all the finite iterations of the Turing jump? We will see that it is not the case: the theory

of  $(\mathcal{D}, \leq)$  is of *maximal* complexity. What do we mean by this? Consider  $T_2$ , the second-order theory of  $(\mathbb{N}, \times, +, 0, 1)$ , i.e., the set of second-order formulas which are true in  $\mathbb{N}$ . Recall that a second-order formula is of the form  $\forall X \exists Y \dots F(X, Y, \dots)$  where the variables  $X, Y, \dots$  are sets of integers, and where  $F$  is a first-order formula, parameterized by these sets.

The theory  $T_2$  is therefore the set of second-order formulas which are true in  $\mathbb{N}$ . If we have access to  $T_2$ , we can know if a formula of the first-order theory of  $(\mathcal{D}, \leq)$  is true: the quantifications  $\exists \mathbf{a}$  and  $\forall \mathbf{a}$  can be replaced by quantifications on the elements of  $2^{\mathbb{N}}$  and using Theorem 9-3.4 relativized to the parameters of the second-order, which allows to transform a  $\Sigma_n^0(X, Y, \dots)$  predicate into a  $\Sigma_n$  formula of arithmetic, we can transform a formula of the first-order  $F$  of  $(\mathcal{D}, \leq)$  into an equivalent second-order formula  $F^*$  of  $(\mathbb{N}, \times, +, 0, 1)$ . Simpson showed that the reverse was also true: through ingenious coding, it is possible to transform a formula of second-order arithmetic into an equivalent formula of  $(\mathcal{D}, \leq)$ .

Let us insist, before going further, on the *extreme* complexity of  $T_2$ . We will study in Part IV all the details of the complexity of  $T_2$  restricted to  $\Pi_1^1$  formulas, that is to say restricted to formulas within which the second-order quantifications are all universal. We will see that this theory already has a considerably high Turing degree compared to  $\emptyset'$  or even all finite iterations of  $\emptyset'$ . This Turing degree is nevertheless well defined and it is *absolute* in the sense that the truth value of a  $\Pi_1^1$  formula will be the same in the transitive models of Set Theory sharing the same computable ordinals (see Part IV for a formal definition). From the level of  $\Pi_2^1$  complexity of the formulas, this meaning becomes more vague. The truth value of this kind of formula will however remain unchanged in all the transitive models of Set Theory, which this time share not only the same computable ordinals, but the same countable ordinals. Subject to accepting the absoluteness of countable ordinals, the truth of the  $\Pi_2^1$  formulas is also absolute. The Turing degree of the  $\Pi_2^1$  level of  $T_2$  is itself higher than all the Turing degrees discussed in this book (it is in a way the supremum of all the  $\Pi_1^1$  singletons, of which we will see the definition in Section 30-4). The truth value of a  $\Pi_3^1$  formula may differ between two ZFC models which share the same ordinals, and the meaning that there is to say, that such a formula is *true* or *false* vanishes here a little more. As for the Turing degree of the  $\Pi_3^1$  theory of  $T_2$ , from where we are, that is to say the world of computable things, from which we observe the structure of the universe, even the best telescopes do not allow us to see it: it is simply too far away from us.

Here then is the complexity of the theory of Turing degrees! We deliver here the modern proof of Simpson's theorem, which differs from that which

was originally produced, and which presents its own interest. Simpson's proof follows from the following theorem.

**Theorem 4.4 (Slaman et Woodin [205])**

Let  $R \subseteq \mathcal{D}^n$  for  $n \in \mathbb{N}$  be a countable set of  $n$ -tuples of Turing degrees. Then,  $R$  is definable in  $(\mathcal{D}, \leq)$  with a finite number of parameters. Formally, there is a formula  $F(x_1, \dots, x_n, y_1, \dots, y_m)$  and parameters  $\mathbf{p}_1, \dots, \mathbf{p}_m \in \mathcal{D}$  such that  $(\mathbf{a}_1, \dots, \mathbf{a}_n) \in R$  iff  $F(\mathbf{a}_1, \dots, \mathbf{a}_n, \mathbf{p}_1, \dots, \mathbf{p}_m)$  is true in  $(\mathcal{D}, \leq)$ .

Let us see immediately how to use Theorem 4.4 to show Simpson's theorem: it suffices to code a standard model of arithmetics in the Turing degrees.

**Theorem 4.5 (Simpson [200])**

The first-order theory of the Turing degrees is many-one equivalent to that of second-order arithmetic.

PROOF. We have already seen in the previous paragraphs how to transform a statement of  $(\mathcal{D}, \leq)$  into an equivalent statement of second-order arithmetic. Now let's see how to do the reverse.

The idea is to code a standard model of  $(\mathbb{N}, +, \times, 0, 1)$  in the Turing degrees. Such a model will be coded by a finite set of parameters coding a set  $N$  of Turing degrees which represent  $\mathbb{N}$ , with a specific degree representing 0 and another representing 1. The relations  $+$  and  $\times$  are also coded by a finite set of parameters.

It is possible to create a formula of  $\mathcal{D}$  which checks whether a finite set of parameters encodes well the standard model of arithmetic: it is simply necessary to check the axioms of Robinson arithmetic (see Section 9-2.3), which are in finite number, and then check that any subset of the model has a smallest element. We can refer to Theorem 9-3.13 to see that these conditions are necessary and sufficient to verify that we are indeed dealing with the standard model of integers. The universal quantification "any subset of the model has a smallest element" can be replaced by a universal quantification on the Turing degrees used as parameters to encode subsets of  $N$  (our set of degrees which represents  $\mathbb{N}$ ).

Given a formula  $F$  of second-order arithmetic, we can finally transform it into an equivalent formula  $F^*$  in  $\mathcal{D}$ , by replacing the quantifications on the sets by quantifications on the parameters encoding these sets. The second-order formula of arithmetic will therefore be interpreted in  $\mathcal{D}$  by the formula: there are parameters encoding a standard model of arithmetic, such that  $F^*$  is verified in this model. ■

Now let's move on to the Slaman and Woodin coding. The following lemma constitutes the difficult part of the proof. It is based on a forcing which may seem relatively simple in principle, but the execution of which is subtle and requires a lot of trickery to be completed.

**Lemma 4.6 (Slaman et Woodin [205]).** Any countable anti-chain in the Turing degrees is definable with three parameters. ★

PROOF. Let  $(\mathbf{a}_n)_{n \in \mathbb{N}}$  be an anti-chain in the Turing degrees and let  $\mathbf{b}$  be an upper bound of this anti-chain. We are going to define two degrees  $\mathbf{g}_0, \mathbf{g}_1$  such that for any degree  $\mathbf{y} \leq \mathbf{b}$  not bounding any  $\mathbf{a}_i$  then  $\mathbf{g}_0 \cup \mathbf{y}$  and  $\mathbf{g}_1 \cup \mathbf{y}$  have a greatest lower bound and this greatest lower bound is  $\mathbf{y}$ . In other words, any degree both below  $\mathbf{g}_0 \cup \mathbf{y}$  and below  $\mathbf{g}_1 \cup \mathbf{y}$  must also be below  $\mathbf{y}$ . Conversely there will exist for every  $i$  a degree both below  $\mathbf{g}_0 \cup \mathbf{a}_i$  and below  $\mathbf{g}_1 \cup \mathbf{a}_i$  which will not be below  $\mathbf{a}_i$ . It follows that each  $\mathbf{a}_i$  will be a minimal element satisfying the formula

$$F(\mathbf{x}) = \mathbf{x} \leq \mathbf{b} \wedge \exists \mathbf{c} (\mathbf{c} \not\leq \mathbf{x} \wedge \mathbf{c} \leq \mathbf{g}_0 \cup \mathbf{x} \wedge \mathbf{c} \leq \mathbf{g}_1 \cup \mathbf{x}).$$

In particular, the degrees  $\mathbf{a}_i$  will be exactly the degrees  $\mathbf{x}$  satisfying the formula  $F(\mathbf{x}) \wedge \forall \mathbf{y} (\mathbf{x} \leq \mathbf{y} \rightarrow F(\mathbf{y}))$ . The fact that the  $\mathbf{a}_i$  form an anti-chain is used only to define them as minimal solutions of  $F$ , but no longer occurs thereafter.

We will use for the construction of  $\mathbf{g}_0$  and  $\mathbf{g}_1$  the following fact: every Turing degree contains a set  $X$  computable in any infinite subset of  $X$ . We can see it as follows: given any set  $Y$  we define  $X$  as being the set of prefixes  $\sigma \prec Y$ , via an encoding of the finite strings by integers.

Let  $B$  be a representative of  $\mathbf{b}$ . Let  $A_n$  be a representative of  $\mathbf{a}_n$  computable in any of its infinite subsets. We will define two sets  $G_0, G_1$  such that for all  $i$ , there exists  $C \not\leq_T A_i$  such that  $C \leq_T G_0 \oplus A_i$  and  $C \leq_T G_1 \oplus A_i$ , and such that for all  $Y \leq_T B$  and all  $D$  such that  $D \leq_T G_0 \oplus Y$  and  $D \leq_T G_1 \oplus Y$ , then  $Y \geq_T D$  or  $Y \geq_T A_j$  for some  $j$ .

We proceed via a forcing which presents similarities with that of Theorem 2.7. Let  $\mathbb{P}$  be the set of conditions of the form  $(\sigma_0, \sigma_1, n)$  for  $\sigma_0, \sigma_1 \in 2^{<\mathbb{N}}$  with  $|\sigma_0| = |\sigma_1|$  and  $n \in \mathbb{N}$ . The integer  $n$  is used to restrict the possible extensions, the string  $\sigma_0$  is used for the first generic  $G_0$  and the string  $\sigma_1$  for the second generic  $G_1$ . As in the proof of Theorem 2.7, we can see  $G_0$  and  $G_1$  as being constructed by columns. The integer  $n$  indicates that the construction will henceforth be restricted on the first  $n$  columns: for a column  $k \leq n$ , if  $a \notin A_k$  then there is no restriction for the bit  $\langle k, a \rangle$  of the two generics. If on the other hand  $a \in A_k$ , then the bit  $\langle k, a \rangle$  of the two generics must be identical (without necessarily being equal to  $A_k(a)$ ). Formally we have  $(\sigma_0, \sigma_1, n) \succeq (\tau_0, \tau_1, m)$  if  $\sigma_0 \preceq \tau_0$ , if  $\sigma_1 \preceq \tau_1$ , if  $n \leq m$ ,

and if moreover the following condition is satisfied: for all  $k \leq n$ , then for all  $a \in A_k$  such that  $|\sigma_i| \leq \langle k, a \rangle < |\tau_i|$ , the values  $\tau_0(\langle k, a \rangle)$  and  $\tau_1(\langle k, a \rangle)$  must be the same.

Consider two sufficiently generic sets  $G_0, G_1$  for this forcing. For each  $A_n$  and for  $a \in A_n$  sufficiently large, we will have  $G_0(\langle n, a \rangle) = G_1(\langle n, a \rangle)$ , by definition of what is a valid extension in this forcing. In particular, the sets  $X_0^n, X_1^n \subseteq A_n$  defined by  $X_i^n(a) = 0$  if  $a \notin A_n$  and  $X_i^n(a) = G_i(\langle n, a \rangle)$  otherwise are the same except for a finite number of bits and are therefore both computed by  $G_0 \oplus A_n$  and  $G_1 \oplus A_n$ .

Let us show that if  $G_0, G_1$  are sufficiently generic, then no  $A_n$  can compute the sets  $X_0^n, X_1^n$  thus defined. Given a condition  $\langle \sigma_0, \sigma_1, n \rangle$ , we can take any extension for the side  $\sigma_0$ —this then forces some bits of the extension for the other side. By considering the fact that there are necessarily infinite subsets of  $A_n$  not computable in  $A_n$ , we can necessarily find an extension  $\tau_0 \succeq \sigma_0$  such that for a given functional  $\Phi_e$ ,  $\Phi_e(A_n)$  never produces the restriction of  $\tau_0$  which will be used to make it a prefix of  $X_n^0$ —either because  $\Phi_e(A_n)$  will be partial, or because it will produce a string incompatible with the prefix of  $X_n^0$  thus forced. Since  $X_n^1$  matches  $X_n^0$  except over a finite number of bits, then  $A_n$  will not compute  $X_n^1$  either. This gives us the first part of what we are trying to show: for any  $A_n$  there exists a computable set in  $G_0 \oplus A_n$  and in  $G_1 \oplus A_n$ , but not in  $A_n$ .

It remains to show that for any  $Y \leq_T B$  such that  $Y$  does not compute any  $A_n$ , if  $G_0 \oplus Y$  and  $G_1 \oplus Y$  compute the same set  $C$ , then  $Y \geq_T C$ . Let  $p = \langle \sigma_0, \sigma_1, n \rangle$  be a condition and let  $\Phi_{e_0}, \Phi_{e_1}$  be a pair of functionals. We first separate the chain  $\sigma_0$  from our condition  $p$ . If there exists  $x$  and  $\tau_0 \succeq \sigma_0$  such that for all  $\rho_0 \succeq \tau_0$  we have  $\Phi_{e_0}(Y \oplus \rho_0, x) \uparrow$ , then we consider a string  $\tau_1$  such that  $(\tau_0, \tau_1, n)$  forms a valid extension, for which we will have forced the partiality of  $\Phi_{e_0}(Y \oplus G_0)$ . Suppose now that for all  $x$  and for all  $\tau_0 \succeq \sigma_0$  there exists  $\rho_0 \succeq \tau_0$  such that  $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow$ . Suppose first that there exists an extension  $\tau \succeq \sigma_0$  such that for any extension  $\rho_0, \rho_1 \succeq \tau$  and for all  $x$  we have  $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow = a$  and  $\Phi_{e_0}(Y \oplus \rho_1, x) \downarrow = b$  implies  $a = b$ . Then, we can only produce a unique set via  $\Phi_{e_0}(Y \oplus G_0)$  for any generic  $G_0$ , and this set is then computable in  $Y$ . We therefore force  $\Phi_{e_0}(Y \oplus G_0)$  to compute something which is already computable in  $Y$ . Suppose finally that for all  $\tau \succeq \sigma_0$  there exists  $\rho_0, \rho_1 \succeq \tau$  and  $x$  such that  $\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow \neq \Phi_{e_0}(Y \oplus \rho_1, x) \downarrow$ . We then use the following lemma.

**Lemma 4.7.** For all  $\tau \succeq \sigma$  there are  $x$  and two extensions  $\rho_0, \rho_1 \succeq \tau$  which differ on only one bit and such that

$$\Phi_{e_0}(Y \oplus \rho_0, x) \downarrow = a \neq \Phi_{e_0}(Y \oplus \rho_1, x) \downarrow = b.$$

PROOF. It suffices for that to find two extensions of  $\tau$  of the same size and incompatible on a certain  $x$ . Let  $i_0, \dots, i_k$  be the bits on which these extensions differ. We invert the  $i_0$  bit in the first extension and expand it to get an  $a_0$  value for  $x$ . If  $a_0 \neq a$  we are done. Otherwise we in turn reverse  $i_1$  in this new extension, which we extend further to obtain a value  $a_1$ , and so on. If each value  $a_1 = a_2 = \dots = a_{k-1}$  then  $a_{k-1} \neq b$ , and our string now differs by only one bit from the one that produced  $b$ . ■

Using the lemma, we compute with the help of  $Y$  a sequence of triples  $(\tau_{0,m}, \tau_{1,m}, i_m, x_m)_{m \in \mathbb{N}}$  with  $i_m < i_{m+1}$  such that each  $\tau_{0,m}, \tau_{1,m}$  differ on exactly the bit  $i_m$  and are incompatible on  $x_n$ . The sequence of bits  $(i_m)_{m \in \mathbb{N}}$  is an infinite  $Y$ -computable sequence. Recall that our forcing condition is of the form  $(\sigma_0, \sigma_1, n)$ . If there exists  $i_m$  such that  $i_m = \langle k, a \rangle$  for  $k > m$ , then for any extension  $\tau'$  of  $\sigma_1$  such that  $\langle \tau_{0,m}, \tau', n \rangle$  is a valid extension, then  $\langle \tau_{1,m}, \tau', n \rangle$  forms also one, because there is no constraint on the  $i_m$  bit. We can therefore find an extension of  $\tau'$  of  $\sigma_1$  which forces a value for  $x_n$  (assuming that we cannot force the partiality on that side), and we take the extension  $\tau_{0,m}$  or  $\tau_{1,m}$  of  $\sigma_0$  which forces a different value. We therefore force  $\Phi_{e_0}(Y \oplus G_0)$  and  $\Phi_{e_0}(Y \oplus G_1)$  to be different. If at present there does not exist  $i_m = \langle k, a \rangle$  for  $k > m$ , then by the pigeonhole principle, there must exist a certain  $k \leq m$  for which an infinity of  $i_m$  is of the form  $\langle k, a \rangle$ . Also it is not possible to have  $A_k(a) = 1$  for each of these  $i_m$ , because we would then have an infinite  $Y$ -computable subset of  $A_k$ , or by hypothesis all infinite subset of  $A_k$  computes  $A_k$ , and  $Y$  does not compute  $A_k$ . There must therefore exist  $\tau_{0,m}, \tau_{1,m}$  and  $i_m = \langle k, a \rangle$  such that  $A_k(a) = 0$ . Again, any extension  $\tau'$  of  $\sigma_1$  that is compatible with  $\tau_{0,m}$  will also be compatible with  $\tau_{1,m}$ , because there is no constraint on the bit  $i_m$ . We can then find an extension  $\tau'$  of  $\sigma_1$  which forces a value on  $x_m$  — unless we can force partiality on that side — and choose an extension from  $\tau_{0,m}$  and  $\tau_{1,m}$  which forces another value on  $x_m$ . This concludes the proof. ■

And there you go. The proof of the previous lemma was not without difficulty, but we are now almost out of the woods. Finally, we show that any countable subset of  $\mathcal{D}^n$  can be encoded in the Turing degrees.

PROOF OF THEOREM 4.4. In what follows, the capitalized variables denote sets or sequences of Turing degrees. Let  $R \subseteq \mathcal{D}^n$  be a countable set of  $n$ -tuples. Let  $\mathbf{b}$  be an upper bound on the set of degrees concerned by  $R$  (i.e. on the union of the projections of  $R$  on each coordinate). Let  $(\mathbf{x}_i)_{i \in \mathbb{N}}$  be a list of all the degrees below  $\mathbf{b}$ . Note that  $\mathbf{b}$  is only an upper bound and therefore that some  $\mathbf{x}_i$  may not be degrees of  $R$ .

We then find an anti-chain  $(\mathbf{c}_i^k)_{k \leq n, i \in \mathbb{N}}$  such that  $B = (\mathbf{c}_i^k \cup \mathbf{x}_i)_{k \leq n, i \in \mathbb{N}}$  forms a set of computably independent degrees (we can easily show that such an anti-chain exists, by finite extensions). For  $k \leq n$  fixed, let  $C_k = (\mathbf{c}_i^k)_{i \in \mathbb{N}}$ . We finally define

$$S = \{\mathbf{c}_{i_1}^1 \cup \mathbf{x}_{i_1} \cup \dots \cup \mathbf{c}_{i_n}^n \cup \mathbf{x}_{i_n} : (\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_n}) \in R\}.$$

Note that as  $B$  is computably independent, for an element  $\mathbf{a} \in S$ , there exists an  $n$ -tuple of degrees  $\mathbf{b}_1, \dots, \mathbf{b}_n \in B$ , unique except for the order, such that  $\mathbf{a} = \mathbf{b}_1 \cup \dots \cup \mathbf{b}_n$ . Moreover, the computational independence of  $B$  also guarantees that for  $\mathbf{b}_1$  there exists a unique  $i \leq n$  such that  $\mathbf{b}_1 = \mathbf{c}_m^i \cup \mathbf{x}_m$  for a certain  $m$ , and this  $m$  is also unique. The same goes for  $\mathbf{b}_2, \mathbf{b}_3, \dots$ , which allows to define  $R$  as follows:  $(\mathbf{x}_1, \dots, \mathbf{x}_n) \in R$  if

$$\mathbf{x}_1 \leq \mathbf{b} \wedge \dots \wedge \mathbf{x}_n \leq \mathbf{b} \wedge \exists \mathbf{y}_1 \in C_1 \dots \exists \mathbf{y}_n \in C_n (\mathbf{x}_1 \cup \mathbf{y}_1) \cup \dots \cup (\mathbf{x}_n \cup \mathbf{y}_n) \in S.$$

As each  $C_i$  and as  $S$  are anti-chains, they are definable according to Lemma 4.6. Such a formula is therefore definable in the Turing degrees. ■

Slaman and Woodin used their coding technique as a starting point for a complex study of the *rigidity* of the Turing degrees. A structure is said to be *rigid* if it does not admit any automorphism other than identity, that is to say in the case of degrees, of bijections  $f : \mathcal{D} \rightarrow \mathcal{D}$  such that  $\mathbf{a} \leq \mathbf{b} \leftrightarrow f(\mathbf{a}) \leq f(\mathbf{b})$ . For example, the structure  $(\mathbb{R}, +, \times, \leq)$  of the real numbers is a rigid structure. Given an automorphism  $f : \mathbb{R} \rightarrow \mathbb{R}$  we must have  $f(0) + f(1) = f(1)$  and therefore  $f(0) = 0$ , then  $f(1) = f(1) \times f(1)$  and therefore  $f(1) = 1$ . Using  $f(n+1) = f(n) + 1$  we show that  $f$  is necessarily the identity on  $\mathbb{N}$ , and we show the same thing on  $\mathbb{Q}$  by playing with the multiplication. To finish, by using the order and the fact that  $\mathbb{Q}$  is dense in  $\mathbb{R}$  we then show that  $f$  is necessarily the identity on  $\mathbb{R}$ .

Can we do something similar in the Turing degrees or can we “swap” two degrees  $\mathbf{a}$  and  $\mathbf{b}$ , and extend this swap consistently into an automorphism over  $\mathcal{D}$ ? The question remains open for the moment, even if Slaman and Woodin have considerably reduced the possibilities:

**Theorem 4.8 (Slaman et Woodin [206])**

*Any automorphism over Turing degrees is the identity over  $\mathcal{D}(\geq 0'')$ .*

Slaman and Woodin use this result to show in the same article that the double jump function is definable in the Turing degrees. This result will be extended later by Shore and Slaman:

**Theorem 4.9 (Shore et Slaman [199])**

*The Turing jump is definable in the Turing degrees, without parameters.*

This result can then in turn be used to show that any automorphism over the Turing degrees is the identity over  $\mathcal{D}(\geq \mathbf{0}')$ . The following question remains open, however.

**Question 4.10.** Is there an automorphism other than identity on the Turing degrees? ★

## 5. Structure of the c.e. degrees

Another important area of research in Turing degrees is to restrict their study to a well-chosen subset. In this vein, the study of the structure  $(\mathcal{R}, \leq)$  of the c.e. degrees is certainly the most developed. We make here a quick summary of the main results concerning them.

Let's start right away with Sacks' impressive density theorem, which shows itself at the end of what is certainly one of the most complex infinite injury priority constructions:

**Theorem 5.1 (Sacks (1964))**

*The order  $(\mathcal{R}, \leq)$  is dense: given c.e. sets  $A <_T B$ , there exists a c.e. set  $C$  such that  $A <_T C <_T B$ .*

The structure of the c.e. degrees therefore appears to be quite different from that of the Turing degrees. There are still some similarities: one thus verifies without problem that any pair of elements admits an upper bound, the set  $A \oplus B$  being c.e. if  $A$  and  $B$  are both c.e. Just like in the Turing degrees, we can show — by working out on the construction of two incomparable degrees — that there exists a countable set of computably independent c.e. degrees [163]. As for the Turing degrees, we can deduce that any countable partial order embeds into the c.e. degrees, which shows, as Sacks noticed, that the  $\Pi_1^0$  theory of the c.e. degrees is decidable.

**Theorem 5.2 (Sacks [187])**

*The  $\Pi_1^0$  theory of the c.e. degrees is decidable.*

Given an existential formula on the c.e. degrees, it suffices to check whether it is compatible with the axioms of a partial order. If so, it is true, otherwise it is false.

As with Turing degrees, the question of embedding is not really satisfactory in itself. A more interesting question is that of lattice embedding, such that the least upper bounds are sent to the least upper bounds, and the greatest lower bounds to the greatest lower bounds. The lattice embeddings mentioned henceforth will be considered as satisfying this constraint. The existence of a minimal pair of c.e. degrees shows for example that the diamond lattice generated by incomparable  $\mathbf{c}_0, \mathbf{c}_1$  (that is to say with  $\mathbf{c}_0 \cap \mathbf{c}_1 < \mathbf{c}_0, \mathbf{c}_1 < \mathbf{c}_0 \cup \mathbf{c}_1$ ) can be embedded in this way in Turing degrees, the greatest lower bound being the degree  $\mathbf{0}$ .

The construction of a minimal pair of c.e. degrees (see Theorem 13-5.2) has been exploited to show much stronger results. A lattice is said to be *distributive* if  $a \cap (b \cup c) = (a \cup b) \cap (a \cup c)$ .

**Theorem 5.3 (Thomason [222] Lachlan and Lerman)**

*Any countable distributive lattice embeds into the c.e. degrees*

Regarding non-distributive lattices, some can embed into the c.e. degrees [130] and others can't [133]. No characterization is known to date.

We have seen that the theory of Turing degrees is of maximum complexity, the same goes for that of the c.e. degrees, which has the same complexity as the theory consisting of the true formulas of first-order arithmetic. The first result in this direction was obtained by Harrington and Shelah [82] who showed that the first-order theory of the c.e. degrees was undecidable. The proof was then simplified and improved by Harrington and Slaman, then by Nies, Shore and Slaman [167], who proved the following theorem.

**Theorem 5.4 (Nies, Shore et Slaman [167])**

*We can effectively transform a statement  $F$  of first-order arithmetic into a statement  $F^*$  on the c.e. degrees, such that  $F$  is true in  $(\mathbb{N}, +, \times, 0, 1)$  iff  $F^*$  is true in  $(\mathcal{R}, \leq)$ .*

Finally, Lempp, Nies and Slaman [136] showed that the  $\Pi_3^0$  theory of the c.e. degrees was already undecidable, which leaves open the following question:

**Question 5.5.** Is the  $\Pi_2^0$  theory of the c.e. degrees decidable? ★

## Part II

# Algorithmic Randomness



# Chapter 15

## Introduction

We have approached in Section 10-2 a notion of typicality for sets which directly follows from Baire's work: being *generic*. It is an understatement to say these underlying notions of meager and co-meager sets have proved themselves to be of great use in many fields: Analysis, Set Theory, equivalence relations, Computability Theory, etc.

And yet, if René Baire's work fruitfulness is exceptional, that of his colleague, Henri Lebesgue, is still above. The two mathematicians knew each other very well and studied with the same teacher — Émile Borel — whose brilliant career echoes that of his two students.

Lebesgue works on measure theory, which will lead to a notion of typicality orthogonal to that of Baire: to be random. Measure theory can be used in a similar way to Baire's categories, for common problems: for example the points of discontinuity of a limit of continuous functions form a meager class. In Cantor space, this is a consequence of the proof of Theorem 10-3.20: if  $G$  is 1-generic, then the Turing jump is continuous in  $G$ . We show something similar for measure: let  $f$  be a limit of continuous functions, then there exist closed sets of arbitrarily large measure, on which the restriction of  $f$  is continuous. This will be a consequence, in Cantor space, of



Henri Lebesgue, 1875–1941

the proof of Theorem 19-3.8. Finally, measure theory owes its fame to its use as a core concept in the axiomatization of probability theory. In that respect, a typical set for measure theory can be informally seen as “random”, that is, satisfying all the properties a set satisfies with probability 1.

The study of random numbers is also a way of studying numbers individually “inaccessible”, but which reveal a lot of secrets when we consider rather the groups to which they belong. This intuition is already in Borel’s work in the 1950s, as witnessed by this quote from “The inaccessible numbers”:

*“All the theories which relate to the measure of sets can therefore be considered as a contribution to the theory of inaccessible numbers; if we cannot study any of these numbers individually, we can study probability problems that are relative to them as a set, or to some of their subsets. The answer to certain questions is in that respect found to be a coefficient of probability. Such an answer can often be of great interest to many scientific questions.”*

In Computability Theory, this brings us to what we now call *Algorithmic Randomness*, a field that arose out of a seemingly simple philosophical question: what is a random sequence of 0 and 1 ? More precisely, given a sequence of 0 and 1, is it “reasonable” to think that it could have been obtained by a fair coin tossing?

We have of course an intuition on the matter: for instance the sequence 000 0000000000 seems less random than the sequence 01101001010010. But how to give a precise meaning to this intuition? In particular, when one think about it, each of the two sequences above has exactly the same probability of occurrence, that is,  $2^{-14}$ . However, getting the second out of repeatedly tossing a coin will seem perfectly normal, while getting the first will seem extraordinary. We indeed have the intuition that a random number — here by “number” we mean a real  $r \in ]0, 1[$ , seen as an infinite sequence of 0 and 1 via its binary representation — should satisfy the law of large numbers: the frequency of 1 must be the same as the frequency of 0, i.e.,  $1/2$ . Going further we can also ask that the frequency of 00, 01, 10 and 11 be the same, namely  $1/4$ , and so on for all finite sequences of bits. Émile Borel calls such numbers *normal numbers*, and showed that a number is normal with probability 1. But is this sufficient to consider that a number can be the result of a random draw of its bits? The answer is no, and one can use to see that a result of Champernowne [31] which showed that the sequence obtained by concatenating in order all the integers written in binary, is a normal number: 0 1 10 11 100 101 110 111 . . . . It is clear, however, that this number is not random, because it is computable by a very simple process, while a random number should not be computable: can one imagine an algorithm predicting the lottery numbers, or the next

card drawn at a game of blackjack? It would be the end of casinos. Let us note the various electronic gambling machines resort to physical processes paired with algorithmic pseudo-random generators, precisely so that a player cannot “predict the future” with a simple computation.

The first attempt to formally define the randomness of an infinite sequence of bits is generally attributed to Richard von Mises, who proposed in 1919 the following definition: a sequence is random if each of its infinite subsequences, obtainable by a “reasonable selection process”, satisfies the law of large numbers. Computability Theory that developed soon after provides us with a natural framework to formally define what a reasonable selection process should be: simply a computable one. In 1939 Jean Ville demonstrated in his thesis “Étude critique de la notion de collectif” [227] that von Mises’ definition was insufficient: for any countable set of selection rules, some sequences are random in von Mises sense, but have each of their prefix containing more 0 than 1. It is indeed very improbable to flip a coin a 1000 times, while always having more heads than tails. When the number of draws extends to infinity, this probability drops to 0. One could consider extending von Mises’ idea and asking the selected sub-sequences not only to satisfy the law of large numbers, but also other laws which are satisfied with probability 1. But which laws to choose? In fact if we fix a finite number  $L_1, \dots, L_n$  of them, there is no reason to believe that we will not be able to find a sequence of which all the computable sub-sequences satisfy each of these laws, while escaping an  $n + 1$ -th.

It was in 1966 that Martin-Löf proposed the definition which is today still considered “satisfactory” by most of the specialists: he considers all the  $\Pi_2^0$  laws which are satisfied with probability 1, noting that with the addition of a certain condition on these  $\Pi_2^0$ , they can combine into a single universal law. A little later Chaitin [28], Gács [68] and Levin [141] all three independently and almost simultaneously introduced a definition of Algorithmic Randomness using a completely different paradigm: that of compressibility, using a special version of Kolmogorov complexity. Levin [140] and Schnorr [192] demonstrated that Martin-Löf’s definition of randomness coincided with that of Chaitin/Gács/Levin, thus revealing the robustness of this notion, for which we subsequently found many other characterizations.

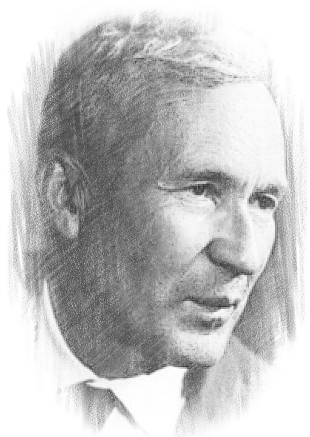
Kolmogorov’s notions of compressibility and complexity allow a smooth transition from Computability Theory to the study of randomness, and so this is where we’ll start.



# Chapter 16

## Kolmogorov complexity and random numbers

Andrei Nikolayevich Kolmogorov (1903 - 1987) is certainly one of the most famous and prolific mathematicians of *the Moscow school*, founded by Dmitri Egorov and Nikolai Luzin at the beginning of the XX<sup>e</sup> century, and on which we will have the opportunity to discuss again in Section 29-1. By the end of his thesis in 1929, carried out under the supervision of Luzin, Kolmogorov had already published numerous articles and gained international fame. In 1931 he became a professor at the University of Moscow and there would lead a brilliant career during which he participated in the founding of whole sections of modern mathematics.



Andreï Nikolaïevitch Kolmogorov, 1903–1987

His best-known works undoubtedly concern the axiomatization, in 1933, of probability theory [10], which we will discuss again in Chapter 17. Thirty years later, Kolmogorov has made important contributions in topology, in dynamical systems, and participated in the resolution of Hilbert's thirteenth problem. His career is not over, however. He is then on the verge of starting another field of mathematical

study which will once again have a considerable impact: the theory of algorithmic randomness, complementary to probability theory, this time with the help of a new tool, namely computer science. He develops in particular his eponymous notion of complexity, the richness of which we will endeavor to show throughout this chapter.

## 1. Kolmogorov complexity

Informally, the Kolmogorov complexity of a finite object is a measure of the amount of information needed to compute it. If Kolmogorov [120] was the first to publish on the subject, this idea actually dates back to the work of Solomonoff [208].

**Definition 1.1.** We call *Machine* a partial computable function from  $2^{<\mathbb{N}}$  to  $2^{<\mathbb{N}}$ . The *Kolmogorov complexity of  $\sigma$  relative to  $M$* , denoted  $C_M(\sigma)$ , is the length of the smallest string  $\tau$  such that  $M(\tau) = \sigma$ . Formally  $C_M(\sigma) = \min\{|\tau| : M(\tau) = \sigma\}$  (and if no such string exists then  $C_M(\sigma) = +\infty$  by convention).  $\diamond$

The Kolmogorov complexity of a string relative to a machine  $M$  can be seen as a measure of its maximum possible compression via  $M$ . It is therefore a relative notion, which depends on the machine that is used, and if the machine in question does nothing, the notion is not very interesting. The idea is of course to use machines which compress information as much as possible.

### 1.1. Universal machine

Solomonoff was the first to understand the possibility of defining an optimal machine, which is also called *universal*: a machine whose compression ratio will be at least as good as that of any other machine, up to constant.

**Definition 1.2.** A machine  $U$  is said to be *universal* if for any machine  $M$  there exists a constant  $c_M \in \mathbb{N}$  such that  $C_U(\sigma) \leq C_M(\sigma) + c_M$  for any string  $\sigma$ .  $\diamond$

A universal machine therefore compresses as well as any other machine  $M$ , but up to a certain additive constant  $c_M$ . Obviously, if the constant is large compared to the string one wants to compress, the notion loses its force, but the important point is that the constant depends only on the machine  $M$  and not on elements to compress. The weight of this constant therefore decreases as the length of the strings considered increases. By introducing his concept of universal machine, Solomonoff obviously showed

that such a machine exists, which was also independently demonstrated by Kolmogorov.

**Theorem 1.3 (Solomonoff [209], Kolmogorov [121])**

*There is a universal machine.*

PROOF. Let  $(M_e)_{e \in \mathbb{N}}$  be an enumeration of all the machines, i.e.  $M_e$  is the machine of code  $e$ . It suffices to define the computable function  $U : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  which on the string  $0^e 1 \sigma$  returns the value of  $M_e(\sigma)$ . Given a machine  $M_e$  it is clear that for any string  $\sigma$  we have  $C_U(\sigma) \leq C_{M_e}(\sigma) + (e + 1)$ . ■

### Notation

We now fix a universal machine  $U$ , and we denote by  $C(\sigma)$  the value  $C_U(\sigma)$ . Therefore  $C(\sigma)$  will be the *Kolmogorov complexity* of  $\sigma$ .

Note that the universal machine we have fixed does not matter: up to an additive constant, all the universal machines compress the strings in an optimal way and all the theorems which will follow are independent of the one we choose.

## 1.2. Random strings

The idea of using Kolmogorov complexity as a measure of randomness is simple: the less compressible a string, the more random it is. It is easily shown that incompressible strings of every length exist.

**Proposition 1.4.** For all  $n \in \mathbb{N}$  there exists a string  $\sigma$  of length  $n$  such that  $C(\sigma) \geq n$ . ★

PROOF. This is a simple counting argument:  $U$  is a function and therefore maps a string to at most one other string. Also for all  $n$  the number of strings of length strictly less than  $n$  is  $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ . There is therefore at least one string of length  $n$  which is not computed via  $U$  by any string of length strictly less than  $n$ . ■

In what follows, we will be interested in strings which are incompressible up to a constant: if a string of length 10,000 is compressible by a program of length 9,990, but not better, it can be morally considered as “strongly” random. The importance of this constant will vanish completely when we continue in Section 2 our randomness study on prefixes of infinite objects.

### 1.3. The Turing degree of Kolmogorov complexity

With the use of a universal machine, the Kolmogorov complexity of a string is morally the length of the smallest computer program capable of computing that string. In a way, this is its best possible compression, if we do not take into account the time required for its decompression, which can turn out to be particularly long . . . As for the time required for its compression, the situation is even worse: it is even no longer a computable process!

#### 1.3.1. The Kolmogorov complexity is not computable

We first prove the non-computability of Kolmogorov complexity, via a mathematical formalization of the Berry paradox: “let  $n$  be the smallest integer that we cannot define in less of 50 words”. The paradox — which should appear clearly to the reader — arises from the fact that the word “define” is itself poorly defined. It suffices to replace it with “computable”.

**Proposition 1.5.** The function  $\sigma \mapsto C(\sigma)$  is not computable. ★

PROOF. Suppose that  $\sigma \mapsto C(\sigma)$  is computable. Then, we can create the computable function  $f : \mathbb{N} \rightarrow 2^{<\mathbb{N}}$  which on  $n$  returns the first string  $\sigma$  — let’s say lexicographically — such that  $C(\sigma) > n$ . By using the usual binary encoding of integers, we can therefore define the machine  $M : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  which on the string  $\sigma_n$ , the binary encoding of  $n$ , returns  $f(n)$ .

As the length necessary to encode  $n$  in base 2 is  $\log_2(n)$ , we therefore have  $C_M(\sigma) \leq \log_2(n)$  for any string  $\sigma = f(n)$ . There is therefore a constant  $c_M$  such that  $C(\sigma) < \log_2(n) + c_M$  for any string  $\sigma = f(n)$ . At the same time each of these strings is chosen such that  $C(\sigma) > n$ , which gives  $n < C(\sigma) < \log_2(n) + c_M$ . For  $n$  sufficiently large such that  $n > \log_2(n) + c_M$  we have a contradiction. ■

The function  $\sigma \mapsto C(\sigma)$  is therefore not computable. On the other hand it is *approachable from above*:

**Definition 1.6.** A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is *approachable from above* if it is the limit of uniformly computable functions  $(f_n)_{n \in \mathbb{N}}$  with  $f_{n+1} \leq f_n$  for all  $n$ . ◇

The approximation  $(f_n)_{n \in \mathbb{N}}$  of Kolmogorov complexity can be defined by assigning  $f_0(\sigma)$  the length of the first  $\tau$  found such that  $U(\tau) \downarrow = \sigma$ , and assigning  $f_{n+1}(\sigma)$  the length of the smallest  $\tau$  such that  $U(\tau)[n+1] \downarrow = \sigma$  and  $|\tau| < f_n(\sigma)$  when such a string  $\tau$  exists, and assigning it to  $f_n(\sigma)$  otherwise.

One easily shows that approachable from above functions can be computed by the halting problem.

**Exercise 1.7.** Show that any approachable from above function is  $\emptyset'$ -computable.  $\diamond$

### 1.3.2. Kolmogorov complexity is Turing-complete

It is possible to strengthen Proposition 1.5 to show that Kolmogorov complexity knowledge actually allows to compute the halting problem. This is a good exercise, for which we prepare the reader with a simpler proposition, obvious if we stick to the construction that was made above of a universal machine, but which requires a bit of work if we consider universal machines in an abstract way:

**Proposition 1.8.** Let  $X$  be a representation of a universal machine  $U : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  (for example with  $X(\langle\sigma, \tau\rangle) = 1$  iff  $U(\sigma) \downarrow = \tau$  and  $X(\langle\sigma, \epsilon\rangle) = 1$  iff  $U(\sigma) \uparrow$ ). Then,  $X \geq_T \emptyset'$ .  $\star$

PROOF. We define the machine  $M$  such that  $M(0^e) \downarrow = 0^s$  if  $s$  is the smallest integer such that  $\Phi_e(e)[s] \downarrow$ . If such an integer does not exist the computation  $M(0^e)$  does not halt. Let  $d$  be a constant such that  $C_U(\sigma) < C_M(\sigma) + d$  for any string  $\sigma$ . Given the knowledge of  $U$ , to know if  $\Phi_e(e) \downarrow$  it suffices to look at  $U(\sigma)$  for any string  $\sigma$  of length less than  $e + d + 1$ , to retrieve the largest value  $s$  such that  $U(\sigma) \downarrow = 0^s$  for one of these strings  $\sigma$ , and to compute  $\Phi_e(e)[s]$ . We then have  $\Phi_e(e) \downarrow \leftrightarrow \Phi_e(e)[s] \downarrow$ .  $\blacksquare$

One of the following exercises consists in showing that the Kolmogorov complexity knowledge makes it possible to compute the universal machine  $U$  associated with it and therefore the halting problem. The following consideration will be useful, and has also its own interest: Proposition 1.5 not only shows that Kolmogorov complexity is not computable, but also that for any compression function  $I : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that  $C_I(\sigma) \leq C(\sigma)$ , the function  $\sigma \mapsto C_I(\sigma)$  is not computable. On the other hand, as shown in Exercise 1.11, such a function does not necessarily compute the halting problem, and therefore neither does its associated Kolmogorov complexity. It is necessary for this to use Kolmogorov complexity *exact* knowledge.

**Exercise 1.9. ( $\star$ )** (*Tibor Radó [181]*). Tibor Radó introduced in a 1962 article [181] the famous *busy beaver* function  $\Sigma : \mathbb{N} \rightarrow \mathbb{N}$ , which can be defined as follows with the notations of this book:  $\Sigma(n)$  is the greatest computation time  $t$  such that  $U(\sigma)[t] \downarrow$  for a string  $\sigma$  of length less than or equal to  $n$ . The strings  $\sigma$  realizing for each  $n$  these greatest computation times are baptized “busy beavers” by Tibor Radó. Show that any function  $f \geq \Sigma$  can compute  $\emptyset'$ .  $\diamond$

**Exercise 1.10. (★★)** (*credited to P. Gács. See 2.7.7. in [143]*). Let  $X$  be a representation of  $C : 2^{<\mathbb{N}} \rightarrow \mathbb{N}$  (e.g. with  $X(\langle \sigma, n \rangle) = 1$  iff  $C(\sigma) = n$ ). Show that  $X \geq_T \emptyset'$ .

Hint: show that for  $n$  large enough, there exists a string of length  $2n$ , of complexity less than  $2n$  but such that no program of length less than  $2n$  can compute it in less than  $\Sigma(n)$  time steps.  $\diamond$

**Exercise 1.11. (★)** Show that there exists a compression function  $I : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  such that  $|I(\sigma)| \leq C(\sigma)$  for all  $\sigma \in 2^{<\mathbb{N}}$  and such that  $I' \leq_T \emptyset'$ .  $\diamond$

## 1.4. Numerical properties of Kolmogorov complexity

Here we see some elementary properties of Kolmogorov complexity. We start with two important notations that we will use again throughout this chapter.

### Notation

Given an integer  $n$  we will write by abuse of notation  $C(n)$  to mean  $C(\sigma)$  where  $\sigma$  is the standard binary representation of  $n$ . Note that the length of  $n$ 's binary representation is in the order of  $\log_2(n)$ .

### Notation

We will write  $\leq^+$  or  $=^+$  to mean inequality or equality up to an additive constant. For example the inequality  $C(\sigma) \leq^+ |\sigma|$  of the forthcoming proposition means there exists a constant  $d$  such that  $C(\sigma) \leq |\sigma| + d$  for any string  $\sigma$ .

### 1.4.1. Bounds of Kolmogorov complexity

The Kolmogorov complexity of a string is bounded from below by the Kolmogorov complexity of its length, and from above by its length itself:

**Proposition 1.12.** For any string  $\sigma$ , we have  $C(|\sigma|) \leq^+ C(\sigma) \leq^+ |\sigma|$ .  $\star$

**PROOF.** A string  $\sigma$  always provides at least its own length as information, via the machine  $M$  which on  $\sigma$  returns  $|\sigma|$ . Using  $M$  it is clear that  $C(|\sigma|) \leq^+ C(\sigma)$ : indeed if  $U(\tau) = \sigma$  then  $M(U(\tau)) = |\sigma|$  and therefore a short description of  $\sigma$  also provides the same short description of  $|\sigma|$ .

Moreover, by using the identity machine which on  $\sigma$  returns  $\sigma$ , we obtain  $C(\sigma) \leq^+ |\sigma|$ .  $\blacksquare$

Both upper and lower bounds are reached for arbitrarily long strings. We have seen it with Proposition 1.4 for the upper bound, and we will see it with Theorem 1.17 for the lower bound.

### 1.4.2. The complexity of a pair of elements

Here we present a technical tool that we will need. Given the strings  $\sigma$  and  $\tau$ , what is the length of the smallest program that produces both  $\sigma$  and  $\tau$ ? To formalize this we will use the notation  $\langle \sigma, \tau \rangle$  to indicate a standard encoding of pairs of string. Intuitively the complexity of  $\langle \sigma, \tau \rangle$  can be at least as great as the complexity of  $\sigma$  added to that of  $\tau$ , especially when the strings  $\sigma$  and  $\tau$  do not “share common information”. It is in fact possible to show that the complexity can be even greater than that. If  $\sigma^*$  and  $\tau^*$  are the smallest descriptions for  $\sigma$  and  $\tau$  respectively, the string  $\sigma^*\tau^*$  will not necessarily be a description of both  $\sigma$  and  $\tau$ , because we do not know where to split the string  $\sigma^*\tau^*$  to find the description of  $\sigma$  and that of  $\tau$ . We will see with Exercize 1.15 that there is no solution to get around this problem. It is however still possible not to “lose too much”, as the following proposition shows.

**Proposition 1.13.** For all strings  $\sigma, \tau$  we have  $C(\langle \sigma, \tau \rangle) \leq^+ C(\sigma) + C(\tau) + 2\log_2(C(\tau))$ . ★

PROOF. Let  $\sigma^*$  and  $\tau^*$  be the smallest descriptions for  $\sigma$  and  $\tau$  respectively. A first idea is to encode  $\langle \sigma, \tau \rangle$  with the string  $1^{|\tau^*|}0\tau^*\sigma^*$ . It is then enough to design the machine which first reads all the bits 1 of its input until arriving at a 0, then thanks to the information given by the number  $n$  of 1, knows where to cut the rest of the string to produce  $\tau$  and  $\sigma$ . We then obtain  $C(\langle \sigma, \tau \rangle) \leq^+ C(\sigma) + 2C(\tau)$ .

We can further improve this in the following way: given a string  $\rho$  which codes for  $|\tau^*|$  and therefore such that  $|\rho| = \log_2(C(\tau))$ , it suffices to consider the string  $\bar{\rho}$  which doubles all the bits of  $\rho$ , i.e., replacing 0 by 00 and 1 by 11, then consider the string  $\bar{\rho}01\tau^*\sigma^*$ . This time the machine reads the bits of its input until it reaches 01, then undoubles all the bits to find an encoding of the length of  $\tau^*$ , which then allows to know where to cut the rest of his input. ■

The following exercises show that we necessarily lose when encoding pairs of strings.

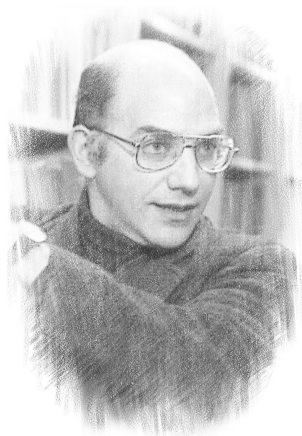
**Exercize 1.14. (★★)** (*Martin-Löf (see [119])*). Show that for any  $k$ , any string  $\sigma$  sufficiently long — of length  $c + k + 2^{c+k}$  for some constant  $c$  — has a prefix  $\tau \preceq \sigma$  such that  $C(\tau) \leq |\tau| - k$ .

Hint: Use the fact that any string  $\rho$  also contains  $|\rho|$  as information. ◇

**Exercise 1.15. (★★)** Deduce from the previous exercise that for all  $k$  there are strings  $\tau, \rho$  such that  $C(\tau\rho) \geq C(\tau) + C(\rho) + k$ . Deduce that for all  $k$  there exist strings  $\tau, \rho$  such that  $C(\langle \tau, \rho \rangle) \geq C(\tau) + C(\rho) + k$ .  $\diamond$

### 1.5. Chaitin's theorem

Gregory Chaitin (1947 -) is a mathematician who, along with Kolmogorov and Solomonov, contributed to the development of algorithmic randomness. He is the source of several deep and remarkable theorems. His most famous work is undoubtedly the discovery of Chaitin's  $\Omega$  number (see Definition 2.9) and of its first properties. If the result we present here is less known, it is nonetheless rich in lessons both mathematically and philosophically.



Gregory Chaitin, 1947 -

Suppose  $X \subseteq \mathbb{N}$  is computable. So for every  $n$ , it should be pretty clear that the first  $n$  bits of  $X$  contain as much information as their length, i.e.  $n$ . Formally  $C(X \upharpoonright_n) =^+ C(n)$ : according to Proposition 1.12 we have  $C(n) \leq^+ C(X \upharpoonright_n)$ , then as  $X$  is computable we can easily create a machine which from  $n$  produces  $X \upharpoonright_n$ , which gives  $C(X \upharpoonright_n) =^+ C(n)$ .

Chaitin succeeded in proving the converse of this result: if each prefix of a set  $X$  admits a maximal compression, then  $X$  is a computable number.

**Theorem 1.17 (Chaitin [29])**

*Let  $X \in 2^{\mathbb{N}}$ . Then,  $X$  is computable iff  $C(X \upharpoonright_n) =^+ C(n)$  for all  $n$ .*

The proof is based on two surprising lemmas. The first tells us that for any machine  $M$  and any string  $\sigma$ , the number of “maximal”  $M$ -compressions of  $\sigma$  is bounded by a constant which does not depend on  $\sigma$ . The second follows from the first and tells us that the number of strings of length  $n$  which admit a “maximal” compression is bounded by a constant which does not depend on  $n$ .

**Lemma 1.18.** For any machine  $M$  there exists a constant  $c$  such that for

all  $\sigma \in 2^{<\mathbb{N}}$  and  $d \in \mathbb{N}$  we have:

$$|\{\tau : |\tau| < C(\sigma) + d \wedge M(\tau) = \sigma\}| < 2^{c+d} \quad \star$$

PROOF. The idea is the following: given  $m, k$ , one can computably enumerate the list  $L_{m,k}$  of the strings which have at least  $2^m$  descriptions of length strictly less than  $k$  via the machine  $M$ . Note that there are at most  $2^{k-m}$  elements in  $L_{m,k}$ . Now to describe an element of  $L_{m,k}$  it suffices to have as information  $m, k$  and a string  $\rho$  of length  $k - m$  which represents its position in the list  $L_{m,k}$ . In fact the information given by  $m$  and  $\rho$  is sufficient since  $k$  can be found from  $m$  and  $\rho$ . According to Proposition 1.13 we have  $C(\langle \rho, m \rangle) \leq^+ C(\rho) + C(m) + 2 \log_2(C(m)) \leq^+ k - m + 2 \log_2(m)$ . We can therefore build a machine  $N$  — which depends on  $M$  — such that each element  $\sigma$  of  $L_{m,k}$  admits a description of length less than or equal to  $k - m + 2 \log_2(m) + c_0$  via  $N$ , for a certain constant  $c_0$  independent of  $m, k$ . We therefore have  $C_N(\sigma) \leq k - m + 2 \log_2(m) + c_0$  for all  $m, k \in \mathbb{N}$  and all  $\sigma \in L_{m,k}$ .

Let  $c_1 \in \mathbb{N}$  be such that  $C \leq C_N + c_1$ . By setting  $c = c_0 + c_1$ ,  $m = 2^{c+d}$  and  $k = C(\sigma) + d$  for some string  $\sigma$ , it suffices to show that  $\sigma \notin L_{m,k}$ . Suppose for contradiction  $\sigma \in L_{m,k}$ . So we have

$$C_N(\sigma) \leq C(\sigma) + d - 2^{c+d} + 2 \log_2(2^{c+d}) + c_0 \leq C_N(\sigma) + 3(c + d) - 2^{c+d}$$

which gives  $2^{c+d} \leq 3(c + d)$ . We can of course assume  $c \geq 4$ , which leads to a contradiction. So  $\sigma \notin L_{m,k}$  and the lemma holds. ■

**Lemma 1.19.** There exists a constant  $c$  such that for all  $n, d \in \mathbb{N}$  we have:

$$|\{\sigma : |\sigma| = n \text{ and } C(\sigma) < C(n) + d\}| < 2^{d+c} \quad \star$$

PROOF. Now it suffices to apply the previous lemma noting that a short description of  $\sigma$  also gives a short description of  $|\sigma|$ . Suppose there are more than  $2^{d+c}$  strings  $\sigma$  of length  $n$  such that  $C(\sigma) < C(n) + d$ . Then there is also a machine  $M$  such that there are more than  $2^{d+c}$  strings  $\tau$  of length less than  $C(n) + d$  for which  $M(\tau) = n$ , which contradicts the previous lemma. ■

PROOF OF THEOREM 1.17. The idea is as follows: given  $d$  fixed the set  $T = \{\sigma : \forall \tau \preceq \sigma \ C(\tau) \leq C(|\tau|) + d\}$  is a tree. According to Lemma 1.19 there exists  $c$  such that for any  $n$  this tree contains at most  $2^{c+d}$  strings of length  $n$ . In particular  $[T]$  is finite, and according to Proposition 8-3.6 all the paths of  $T$  are therefore computable relative to the knowledge of  $T$ .

The problem is that  $T$  itself is not computable, because  $C$  is not. We are therefore going to transform it as follows: we first notice that according to

Proposition 1.4 there exists for all  $k$  an integer  $n$  between  $2^k$  and  $2^{k+1} - 1$  such that  $\log_2(n) \leq C(n)$ . Moreover according to Proposition 1.12 there exists a constant  $e$  for which we always have  $C(n) \leq \log_2(n) + e$ .

We use this information to make a first transformation of  $T$ :

$$T_1 = \{\sigma : \forall \tau \preceq \sigma \ C(\tau) \leq \log_2(|\tau|) + e + d\}.$$

Since  $C(n) \leq \log_2(n) + e$  we have  $T \subseteq T_1$ . Given that for all  $k$  there exists an integer  $n$  between  $2^k$  and  $2^{k+1} - 1$  such that  $\log_2(n) \leq C(n)$  we also have according to Lemma 1.19 for all  $k$  an integer  $n$  between  $2^k$  and  $2^{k+1} - 1$  such that  $|\{\sigma \in T_1 : |\sigma| = n\}| < 2^{e+d+c}$ .

In particular,  $|[T_1]| < 2^{e+d+c}$ . The tree  $T_1$  is however still not computable, but just c.e. Let  $a_k$  be defined for all  $k$  as the minimum for  $n$  between  $2^k$  and  $2^{k+1} - 1$ , of the number of nodes of length  $n$  in  $T_1$ . Note that we necessarily have  $a_k < 2^{e+d+c}$ . Let then  $a = \limsup_{k \in \mathbb{N}} a_k$ , and let  $m$  be the smallest integer such that  $a_k \leq a$  for all  $k \geq m$ .

We can now effectively search for a computation time  $s_0$  and an integer  $k_0 > m$  such that for all  $n$  between  $2^{k_0}$  and  $2^{k_0+1} - 1$ , at least  $a$  nodes of length  $n$  are enumerated in  $T_1[s_0]$ .

Let  $X \in [T_1]$ . We claim that  $X \upharpoonright_{2^{k_0}} \in T_1[s_0]$ . Indeed, otherwise, when  $X \upharpoonright_{2^{k_0+1}}$  and its prefixes will be enumerated in  $T_1$ , it would add a node of length  $n$  which is not in  $T_1[s_0]$ , for each  $n$  between  $2^{k_0}$  and  $2^{k_0+1} - 1$ . The number of nodes of length  $n$  would then go over  $a$ , contradicting  $a_k \leq a$  for all  $k \geq m$ . Thus  $T_1[s_0]$  contains all prefixes of length  $2^{k_0}$  of elements of  $[T_1]$ .

Once  $s_i$  and  $k_i$  defined, we compute the same way inductively  $s_{i+1} > s_i$  and  $k_{i+1} > k_i$  such that for all  $n$  between  $2^{k_{i+1}}$  and  $2^{k_{i+1}+1} - 1$ , at least  $a$  nodes of length  $n$  are enumerated in  $T_1[s_{i+1}]$ .

This allows us to define a computable tree  $T_2$  such that  $[T_2] = [T_1]$ . In particular,  $|[T_2]| < 2^{e+d+c}$  and  $T_2$  contains all the sets  $X$  such that  $C(X \upharpoonright_n) \leq C(n) + d$ , for all  $n$ . All these sets are therefore computable. ■

## 2. Random numbers in the sense of Chaitin/Levin

Kolmogorov's original definition of complexity suffers from certain problems. According to Exercize 1.10, we do not necessarily have  $C(\langle \sigma, \tau \rangle) \leq^+ C(\sigma) + C(\tau)$  for all strings  $\sigma, \tau$ . According to Exercize 1.14, for any  $k$  and

any string  $\sigma$  sufficiently long we have  $C(\tau) \leq |\tau| - k$  for a certain prefix  $\tau \preceq \sigma$ . This prevents us from using  $C$  to define randomness on infinite sequences. It is necessary for that to use a variation of  $C$ : the so-called “prefix-free” complexity.

### 2.1. Prefix-free complexity

Prefix-free complexity was introduced independently by Levin [138] and Chaitin [28]. It is defined like Kolmogorov complexity, while restricting it to the “prefix-free” machines, that we now introduce.

**Definition 2.1.** A set  $W \subseteq 2^{<\mathbb{N}}$  is *prefix-free* if  $\sigma \in W$  implies  $\tau \notin W$  for any string  $\tau \succ \sigma$ . A machine  $M$  is *prefix-free* if its domain of definition is prefix-free, that is to say, for any string  $\sigma$  we have  $M(\sigma) \downarrow$  implies  $M(\tau) \uparrow$  for any string  $\tau \succ \sigma$ .  $\diamond$

A prefix-free machine  $M$  can be seen as a machine on which programs are executed (the strings that  $M$  takes as parameters) *delimited by an end instruction*: if  $\sigma$  is a valid program, then no extension of  $\sigma$  is valid.

Note that in reality, computer programs tend to form prefix-free sets. In any programming language, functions have a start and end symbol. Even if it would not be the case, the filesystems used to record programs in binary are prefix-free (how else would one know where a file ends?). Let us now see a theorem analogous to the universal machine theorem for arbitrary machines.

#### Theorem 2.2

*There is a prefix-free machine which is universal for prefix-free machines.*

PROOF. Let  $(M_e)_{e \in \mathbb{N}}$  be an enumeration of the machines. Let's start by showing that there exists a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that:

- For all  $e \in \mathbb{N}$  the machine  $M_{f(e)}$  is prefix-free.
- If  $M_e$  is prefix-free then  $M_{f(e)} = M_e$

Given a code  $e$ , the function  $f$  computes the code of the machine which on an input  $\sigma$  searches for the smallest  $t \geq |\sigma|$  such that  $M_e(\sigma)[t] \downarrow$ . If such a  $t$  is found, then if there exists  $s < t$  such that  $M_e(\tau)[s] \downarrow$  for  $\tau \neq \sigma$  comparable with  $\sigma$  and such that  $|\tau| \leq s$ , we decide that the machine does not halt and therefore  $M_{f(e)}(\sigma) \uparrow$ . Otherwise if  $M_e(\tau)[t] \downarrow$  for  $\tau \prec \sigma$ , again we decide  $M_{f(e)}(\sigma) \uparrow$ . Otherwise we decide  $M_{f(e)}(\sigma) \downarrow = M_e(\sigma)[t] \downarrow$ .

It is clear that if  $M_e$  is prefix-free then  $M_{f(e)} = M_e$ . It is also clear that  $M_{f(e)}$  is always prefix-free: suppose by contradiction  $M_{f(e)}(\sigma) \downarrow$

and  $M_{f(e)}(\tau) \downarrow$  for  $\sigma \prec \tau$ . Let  $t_\sigma \geq |\sigma|, t_\tau \geq |\tau|$  be the smallest computation times such that  $M_e(\sigma)[t_\sigma] \downarrow$  and  $M_e(\tau)[t_\tau] \downarrow$ . If  $t_\sigma = t_\tau$  we have by definition  $M_{f(e)}(\tau) \uparrow$ . If  $t_\tau < t_\sigma$  we have by definition  $M_{f(e)}(\sigma) \uparrow$  and symmetrically if  $t_\sigma < t_\tau$ .

We now define the following machine  $U$ :  $U(1^e 0 \sigma) = M_{f(e)}(\sigma)$ . Note that as each machine  $M_{f(e)}$  is prefix-free, so is the machine  $U$ . ■

### Notation

In general, if  $M$  is a prefix-free machine, we will write  $K_M(\sigma)$  instead of  $C_M(\sigma)$ . We fix a universal prefix-free machine  $U$ . We write  $K(\sigma)$  for the value  $K_U(\sigma)$ . We will say that  $K(\sigma)$  is the *prefix-free complexity* of  $\sigma$ , and  $C(\sigma)$  its *plain complexity*.

Some theorems on plain complexity still holds for prefix-free complexity. We adapt for example without difficulty the proof of Proposition 1.8 to show that the knowledge of a universal machine prefix-free allows to compute  $\emptyset'$ . On the other hand, the different limits vary a little.

**Exercize 2.3. (★)** Show that no computable function — even partial —  $f : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  can increase the complexity of its parameters: show for such a function that there exists a constant  $c$  such that  $K(f(\sigma)) \leq K(\sigma) + c$  for all  $\sigma \in 2^{<\mathbb{N}}$  on which  $f$  is defined. ◇

## 2.2. Numerical properties of prefix-free complexity

A first practical specificity of prefix-free machines, is their ability to encode pairs of strings in an optimal way.

**Proposition 2.4.** For all strings  $\sigma, \tau$  we have  $K(\langle \sigma, \tau \rangle) \leq^+ K(\sigma) + K(\tau)$ . ★

PROOF. Let  $U$  be the universal prefix-free machine. We define the prefix-free machine  $M$  which on a string  $\sigma$  searches for strings  $\tau, \rho$  such that  $\tau\rho = \sigma$ , such that  $U(\tau) \downarrow$  and such that  $U(\rho) \downarrow$ . the machine then returns  $\langle U(\tau), U(\rho) \rangle$ . Note that if a prefix  $\tau \preceq \sigma$  such that  $U(\tau) \downarrow$  exists, it is unique because  $U$  is prefix-free.

Now if  $\sigma^*$  and  $\tau^*$  are the smallest programs such that  $U(\sigma^*) = \sigma$  and  $U(\tau^*) = \tau$  for arbitrary strings  $\sigma, \tau$ , then  $M(\sigma^* \tau^*) = \langle \sigma, \tau \rangle$  and thus  $K(\langle \sigma, \tau \rangle) \leq^+ K_M(\langle \sigma, \tau \rangle) \leq K(\sigma) + K(\tau)$ . ■

Just as we had  $C(|\sigma|) \leq^+ C(\sigma)$ , we still have  $K(|\sigma|) \leq^+ K(\sigma)$ . On the other hand, we can no longer necessarily guarantee  $K(\sigma) \leq^+ |\sigma|$ , the identity machine not being prefix-free. Instead, we have  $K(\sigma) \leq^+ |\sigma| + K(|\sigma|)$ .

**Proposition 2.5.** For any string  $\sigma$  we have  $K(|\sigma|) \leq^+ K(\sigma) \leq^+ |\sigma| + 2\log_2(|\sigma|)$ . ★

PROOF. Let  $U$  be the universal prefix-free machine. Let us show the first inequality. The machine  $M$  which to  $\tau$  associates  $|U(\tau)|$  is also prefix-free. Suppose  $U(\tau) = \sigma$ . Then,  $M(\tau) = |\sigma|$ . We therefore have  $K(|\sigma|) \leq^+ K(\sigma)$ .

Let us now show  $K(\sigma) \leq^+ |\sigma| + 2\log_2(|\sigma|)$ . We use the same technique as for the proof of Proposition 1.13: given a string  $\rho$ , let  $\bar{\rho}$  be the string which doubles all the bits of  $\rho$ , that is to say which replaces 0 by 00 and 1 by 11. We define the machine  $M$  which halts on strings of the form  $\bar{\rho}01\sigma$  such that  $\rho$  codes for the length of  $\sigma$ , and then returns  $\sigma$ . Note that the machine  $M$  is prefix-free: if  $M(\bar{\rho}01\sigma) \downarrow$  this means that  $\rho$  is an encoding of the length of  $\sigma$ , and if now  $M(\bar{\rho}01\sigma') \downarrow$  for  $\sigma' \prec \sigma$  or  $\sigma \prec \sigma'$  this would also mean that  $\rho$  encodes  $|\sigma'| \neq |\sigma|$  which is impossible. We finally obtain the inequality  $K(\sigma) \leq^+ |\sigma| + 2\log_2(|\sigma|)$  via the machine  $M$  by using the fact that the binary representation of an integer has length  $\log_2$  of this integer. ■

**Exercise 2.6. (★)** (Chaitin [28]). Show that we also have  $K(\sigma) \leq^+ |\sigma| + K(|\sigma|)$  for any string  $\sigma$ . How is this an improvement of Proposition 2.5? ◇

**Exercise 2.7. (★)** Show that for all  $c$  there exists a string  $\sigma$  such that  $K(\sigma) > |\sigma| + c$ . ◇

## 2.3. Random numbers in the sense of Chaitin/Levin

Levin the first, and independently Chaitin both used prefix-free complexity to define randomness for infinite objects.

**Definition 2.8 ([141], [30]).** A set  $X \in 2^{\mathbb{N}}$  is *random in the sense of Chaitin/Levin* if  $K(X \upharpoonright_n) \geq^+ n$  for all  $n$ . ◇

In other words, an infinite sequence of 0 and 1 is random if none of its prefix is compressible, up to an additive constant. Note that for such a definition, prefix-free complexity is necessary: as we have already mentioned, Exercise 1.14 shows that no set  $X \in 2^{\mathbb{N}}$  satisfies  $C(X \upharpoonright_n) \geq^+ n$  for all  $n$ . On the other hand we can show that a set is random in the sense of Chaitin/Levin with probability 1. We will come back to this point in Chapter 17 after having formally defined the notions of Borel classes and measure.

We will see that Chaitin and Levin's definition is a “good” definition of randomness, according to several conceivable criteria. In particular a random number for this definition satisfies the law of large numbers. Moreover if a

number presents a repetition, an incongruity, a “pattern”, it will be possible to create a machine using that to compress even a little its prefixes. We will in fact see, however formal this assertion may be, that a random number for this definition satisfies all the “natural” probability laws which are satisfied with probability 1.

Before looking at this in more detail, we now present the most famous of Chaitin’s results, which can be summarized as follows: the probability that a computer program halts, is a random number.

## 2.4. Chaitin’s $\Omega$ number

What exactly do we mean by “the probability that a computer program halts”? Consider a universal prefix-free machine  $U$ . Suppose we flip a coin without ever stopping to determine a program  $\sigma$  on which  $U$  will run. The successive draws produce strings  $\sigma_1 \prec \sigma_2 \prec \sigma_3 \prec \dots$  with  $|\sigma_i| = i$ . The probability one then obtains  $\sigma_i$  such that  $U(\sigma_i) \downarrow$ , is a random number: written in base 10 for example, the frequency of occurrence of the digits 0, 1, 2,  $\dots$ , 9 in the prefixes of this number’s decimal expansion will converge to 1/10. The frequency of occurrence of each two-digit number will converge to 1/100, etc. Let us now formally define this number:

**Definition 2.9.** Chaitin’s  $\Omega$  number is defined by:

$$\Omega = \sum_{\{\sigma: U(\sigma) \downarrow\}} 2^{-|\sigma|}$$

◇

Since  $U$  is a prefix-free machine, the number  $\Omega$  is less than 1. To see this we introduce informally the concept of measure for intervals  $(a, b) \subseteq [0, 1]$  as being simply what one would measure with a graduated ruler: the value  $b - a$ . A string  $\sigma$  can be seen as the interval of real numbers between  $0.\sigma 0^\infty$  and  $0.\sigma 1^\infty$ . We can easily check that the measure of the interval  $(0.\sigma 0^\infty, 0.\sigma 1^\infty)$  is  $2^{-|\sigma|}$ . The fact that the domain of definition of  $U$  is prefix-free corresponds to the fact that the intervals of the form  $(0.\sigma 0^\infty, 0.\sigma 1^\infty)$  such that  $U(\sigma) \downarrow$  are pairwise disjoint. The sum of their measure therefore corresponds to the measure of their union, which is necessarily less than the measure of  $[0, 1]$ , namely 1.

In the following, we consider the number  $\Omega$  as the binary representation of its decimal expansion. Note first that  $\Omega$  is left-c.e.

**Definition 2.10.** A set  $X \in 2^\mathbb{N}$  is *left-c.e.* if there exists a computable sequence  $(X_s)_{s \in \mathbb{N}}$  such that  $X_s$  is lexicographically smaller than  $X_{s+1}$  for all  $s \in \mathbb{N}$ , and for which  $X = \lim_s X_s$ .

◇

In an equivalent way  $X \in 2^{\mathbb{N}}$  can be left-c.e. if there exists a computable increasing sequence of reals whose binary representation converges to  $X$ . Concretely, in a left-c.e. approximation of  $X$ , a bit can go from 0 to 1 — which corresponds to an increase in the real's value — as for a c.e. set; and a bit can also change from 1 to 0, but only if a more significant bit changes at the same time from 0 to 1, which also corresponds to an increase of the real's value). For example the number 0,010000 is greater than the number 0,000101.

**Exercise 2.11.** Show that any non-empty  $\Pi_1^0$  class  $\mathcal{P}$  contains a left-c.e. element.  $\diamond$

It is clear that the left-c.e. sets are all  $\emptyset'$ -computable: this is a special case of Shoenfield's limit lemma 4-7.2.

**Proposition 2.12.** The number  $\Omega$  is left-c.e. In particular  $\Omega$  can be computed by the halting problem.  $\star$

PROOF. Let us define a computable sequence of computable reals  $(\Omega_s)_{s \in \mathbb{N}}$  such that  $\Omega_s \leq \Omega_{s+1}$  and  $\Omega = \lim_s \Omega_{s \in \mathbb{N}}$ . Each real  $\Omega_s$  is simply

$$\sum_{\{\sigma: |\sigma| \leq s \wedge U(\sigma)[s] \downarrow\}} 2^{-|\sigma|}.$$

It is clear that  $\Omega_s \leq \Omega_{s+1}$  for all  $s$  and that  $\Omega = \lim_s \Omega_{s \in \mathbb{N}}$ .  $\blacksquare$

**Theorem 2.13 (Chaitin [28])**

*The first  $n$  bits of  $\Omega$  are sufficient to determine uniformly whether  $U(\sigma) \downarrow$  or not among the strings  $\sigma$  of length less than or equal to  $n$ . In particular  $\Omega \equiv_T \emptyset'$ .*

PROOF. Let  $(\Omega_s)_{s \in \mathbb{N}}$  be the left-c.e. approximation of  $\Omega$  described in the proof of Proposition 2.12. Suppose we know the  $n$  first bits of  $\Omega$ . Then, it suffices to look for the smallest  $s$  such that  $\Omega_s \upharpoonright_{n+1} = \Omega \upharpoonright_{n+1}$ . Note then that a program  $\sigma$  of length  $m \leq n$  such that  $U(\sigma)[s] \uparrow$  must necessarily be such that  $U(\sigma) \uparrow$ . Indeed in the opposite case one should add the quantity  $2^{-m}$  to  $\Omega_s$ , which would necessarily change a bit of position less than or equal to  $m$  from 0 to 1, which in turn would contradict  $\Omega_s \upharpoonright_{n+1} = \Omega \upharpoonright_{n+1}$ . To know if  $U(\sigma) \downarrow$  on a string  $\sigma$  of length less than or equal to  $n$ , it suffices then to check if  $U(\sigma)[s] \downarrow$ .

In particular  $\Omega$  can compute  $U$  and therefore according to an adaptation of Proposition 1.8 to the prefix-free complexity,  $\Omega$  can compute  $\emptyset'$ .  $\blacksquare$

**Theorem 2.14 (Chaitin [28])**

*The number  $\Omega$  is random in the sense of Chaitin/Levin.*

PROOF. Suppose by contradiction that for all  $c$  there exists  $n$  such that  $K(\Omega \upharpoonright_n) < n - c$ . Let's build the machine  $M$  which on the string  $\tau$  computes  $\sigma = U(\tau)$ , and then seeks to determine the set  $A_\tau$  of strings of length less than or equal to  $|\sigma|$  on which the machine  $U$  halts, by applying the algorithm described in the proof of Theorem 2.13, as if  $\sigma \prec \Omega$ . If  $M$  ever thinks that it has determined  $A_\tau$  after some computation time, then  $M$  returns the first string which is different from  $U(\rho)$  for any string  $\rho \in A_\tau$ .

Note first that if  $M$  is applied to a string  $\tau$  such that  $U(\tau) = \sigma \prec \Omega$ , then  $M$  will halt and output a string  $\rho$  whose Kolmogorov complexity is greater than  $|\sigma|$ . At the same time we have  $K_M(\rho) \leq |\tau|$ . Let  $c$  be such that  $K \leq K_M + c$ . By hypothesis there exists  $n$  such that  $K(\Omega \upharpoonright_n) < n - c$ . So there exists  $\tau$  such that  $|\tau| < n - c$  and such that  $U(\tau) = \Omega \upharpoonright_n$ . We then have by construction  $M(\tau) = \rho$  for a string  $\rho$  such that  $n = |\Omega \upharpoonright_n| \leq K(\rho)$ . This gives us  $n \leq K(\rho) \leq K_M(\rho) + c < n - c + c < n$  which is a contradiction.

So there exists  $c$  such that for all  $n$  we have  $K(\Omega \upharpoonright_n) \geq n - c$ , which means that  $\Omega$  is random in the sense of Chaitin/Levin. ■

Chaitin's  $\Omega$  number has generated a lot of interest within the community. Solovay in a series of unpublished notes on the work of Chaitin [211] introduced a key notion which should allow an in-depth study: for two left-c.e. reals  $X, Y$ ,  $X$  is *Solovay reducible* to  $Y$  if there exists a computable function  $f : \mathbb{Q} \rightarrow \mathbb{Q}$  and a constant  $c$  such that for a rational  $q < Y$ , we have  $0 \leq X - f(q) < c(Y - q)$ . In other words, a rational approximation close to  $Y$ , allows to compute a rational approximation as close to  $X$ , up to a multiplicative constant. Solovay reducibility in particular implies Turing reducibility. Solovay then showed that  $\Omega$  was Solovay-complete i.e., any left-c.e. real is Solovay reducible to  $\Omega$ . Subsequently Calude, Hertling, Khoussainov and Wang [26] showed that a Solovay-complete left-c.e. real is necessarily of the form  $\Omega$  for a certain universal prefix-free machine. Then Kučera and Slaman [126] finally showed that a left-c.e. real which is also random in the sense of Chaitin/Levin, is necessarily Solovay-complete. Putting these results end to end gives us the following theorem.

**Theorem 2.15**

*A set  $X \in 2^{\mathbb{N}}$ , random in the sense of Chaitin/Levin, is left-c.e. iff it is the probability that a certain universal prefix-free machine halts on its input.*

### 3. Characterization of prefix-free complexity

The goal of this section is to show the following theorem, which constitutes an elegant characterization of the prefix-free complexity:

**Theorem 3.1 (Chaitin [30])**

*The function  $\sigma \mapsto K(\sigma)$  is, up to the additive constant, the smallest function  $f : 2^{\mathbb{N}} \rightarrow \mathbb{N}$ , approachable from above and such that  $\sum_{\sigma} 2^{-f(\sigma)} \leq 1$ .*

The heart of this characterization lies in the KC theorem, known as Kraft-Chaitin, allowing to build prefix-free machines from an abstract set of requests, called “bounded requests set”:

**Definition 3.2.** We call *bounded requests set* any set  $A \subseteq 2^{<\mathbb{N}} \times \mathbb{N}$  such that

$$\sum_{(\sigma, l) \in A} 2^{-l} \leq 1$$

and we denote by  $\text{weight}(A)$  this quantity. ◇

We will intensively reuse in what follows the concept of bounded requests set and the KC theorem, proved independently by Levin [138], Schnorr [192] and Chaitin [28].

**Theorem 3.3 (KC theorem)**

*Let  $A$  be a c.e. bounded requests set. Then, there exists a prefix-free machine  $M$  such that  $(\sigma, l) \in A$  implies  $M(\tau) = \sigma$  for a string  $\tau$  such that  $|\tau| = l$ . In particular  $K_M(\sigma) \leq l$ .*

**PROOF.** The reader can use Figure 3.4 to understand the following construction. Suppose without loss of generality that  $A$  is an infinite set. Let  $(A_s)_{s \in \mathbb{N}}$  be an approximation of  $A$ . Note that we therefore necessarily have  $\text{weight}(A_s) < 1$  for all  $s$ .

We will describe a machine  $M$  as a c.e. set of pairs  $(\sigma, \tau)$ , meaning then  $M(\sigma) = \tau$ . We will write  $M_s$  for the enumeration of  $M$  at step  $s$ . At each computation step  $s$ , for each length  $l \geq 1$  we define a string  $\sigma_s^l$ , either of length  $l$ , or equal to the empty word  $\epsilon$ , and a set  $r_s \in 2^{\mathbb{N}}$ . The strings  $\sigma_s^l$  other than the empty word will correspond to the strings available for a mapping at step  $s + 1$ . The role of the sequence  $(r_s)_{s \in \mathbb{N}}$  is twofold. First, the real represented by  $r_s$  in base 2, will be equal to the weight of  $A_s$ . Then if the  $(n - 1)$ -th bit of  $r_s$  is 0, this also means that the string  $\sigma_s^n$  is different from  $\epsilon$  and is therefore available for a future mapping. We must ensure at each step  $s$  that:

- (1) The set of mapped strings in  $M_s$  together with the set of strings  $\sigma_s^l$  different from the empty word, form a prefix-free set.
- (2)  $r_s$  seen as the binary representation of a real equals  $\text{weight}(A_s)$
- (3) If  $r_s(n-1) = 0$ , then the string  $\sigma_s^n$  is a string of length  $n$ . Otherwise it is the empty word.

At step 0, we define  $\sigma_0^l = 1^{l-1}0$  and  $r_0$  as the infinite sequence of 0. Thus (1), (2) and (3) are verified at step 0.

At step  $s+1$ , suppose that  $(\tau, l)$  is enumerated in  $A_{s+1}$ . If  $r_s(l-1) = 0$  we enumerate  $(\sigma_s^l, \tau)$  in  $M$  at step  $s+1$ , we assign  $\sigma_{s+1}^l$  to the empty word and we change  $r_{s+1}(l-1)$  to 1. For  $i \neq l$  we keep  $r_{s+1}(i-1) = r_s(i-1)$  and  $\sigma_{s+1}^i = \sigma_s^i$ . We easily check by induction that (1), (2) and (3) are true at step  $s+1$ .

Otherwise if  $r_s(l-1) = 1$ , let  $n$  be the largest integer strictly greater than 0 and less than  $l$  such that  $r_s(n-1) = 0$ . Note that such an integer necessarily exists because otherwise  $r_s + 2^{-l} \geq 1$  and therefore  $\text{weight}(A_s) + 2^{-l} \geq 1$  which is impossible by hypothesis. We assign  $\sigma_{s+1}^n$  to the empty word and we change  $r_s(n-1) = 1$ . Then for each  $n < i \leq l$ , we assign  $\sigma_{s+1}^i$  to  $\sigma_s^n 1^{i-n-1}0$  and  $r_{s+1}(i-1) = 0$ . Then we enumerate  $(\sigma_s^n 1^{l-n-1}1, \tau)$  in  $M$  at step  $s+1$ . For  $1 \leq i < n$  and  $i > l$  we define  $r_{s+1}(i-1) = r_s(i-1)$  and  $\sigma_{s+1}^i = \sigma_s^i$ . We easily verify by induction that (1), (2) and (3) are true at step  $s+1$ .

This concludes the construction. Since (1) is true at each step  $s$  it is clear that  $M$  is a prefix-free machine. Also by construction we have  $(\sigma, l) \in A$  implies  $M(\tau) = \sigma$  for a string  $\tau$  of length  $l$ . ■

**PROOF OF THEOREM 3.1.** Let  $f : 2^{<\mathbb{N}} \rightarrow \mathbb{N}$  be an approachable from above function such that  $\sum_{\sigma} 2^{-f(\sigma)} \leq 1$ . Suppose first that  $f$  is computable. We then only need to enumerate each pair  $(\sigma, f(\sigma))$  in a bounded requests set. According to the KC theorem there exists a machine  $M$  such that  $K(\sigma) \leq^+ K_M(\sigma) \leq f(\sigma)$ .

Now if  $f$  is approachable from above via an approximation  $(f_s)_{s \in \mathbb{N}}$ , we enumerate in a bounded requests set  $L$  the pairs  $(\sigma, f_0(\sigma) + 1)$ , then the pairs  $(\sigma, f_{s+1}(\sigma) + 1)$  for each  $\sigma$  and each  $s$  such that  $f_{s+1}(\sigma) < f_s(\sigma)$ . Let  $A_0$  be the set of pairs  $\langle s, \sigma \rangle$  such that  $s$  is the last computation time for which  $f_s(\sigma) > f_{s+1}(\sigma) = f(\sigma)$ , and let then  $A_{n+1}$  be recursively defined as the set of pairs  $\langle s, \sigma \rangle$  such that  $s$  is the last computation time for



such that  $K(\sigma) \leq^+ K_M(\sigma) \leq f(\sigma) + 1$ . ■

We will sometimes use Theorem 3.1 in the following form, known as the “coding theorem”: for any prefix-free machine  $M$ , the probability of obtaining  $\sigma$  via  $M$  is always bounded, up to a multiplicative constant, by  $2^{-K(\sigma)}$ .

**Theorem 3.5 (Chaitin [28], Levin [138, 141])**

*Let  $M$  be a prefix-free machine. We denote by  $P_M(\sigma)$  the probability that  $M$  writes  $\sigma$ , that is to say*

$$P_M(\sigma) = \sum_{\{\tau: M(\tau) \downarrow = \sigma\}} 2^{-|\tau|}.$$

*Then, there exists a constant  $c_M$  such that for all  $\sigma$  we have  $P_M(\sigma) \leq 2^{-K(\sigma)} 2^{c_M}$ .*

PROOF. It suffices to notice that the function  $\sigma \mapsto -\log_2(P_M(\sigma))$  is approachable from above and such that  $\sum_{\sigma} 2^{\log_2(P_M(\sigma))} \leq 1$ . We therefore have a constant  $c_M$  such that  $K(\sigma) \leq -\log_2(P_M(\sigma)) + c_M$  for all  $\sigma$ . ■

## 4. K-trivial sets

Recall now Chaitin’s remarkable characterization of the computable sets as those with minimal Kolmogorov complexity on each of their prefixes. We introduce the same notion here, but for the prefix-free complexity:

**Definition 4.1.** Let  $A \subseteq \mathbb{N}$ . The set  $A$  is  $K$ -trivial if  $K(A \upharpoonright_n) \leq^+ K(n)$  for all  $n$ . ◆

Chaitin tried to adapt his proof to show that  $K$ -trivial sets are all computable, but without success. He could however show there are only countably many of them, and in particular that they are all  $\Delta_2^0$ . Solovay, in his series of unpublished notes devoted to Algorithmic Randomness, demonstrated the existence of non-computable  $K$ -trivial sets. To show this, we will use a convenient technique allowing easy constructions of non-computable c.e. degrees: the so called *simple sets*.

### 4.1. Simple sets

Simple sets were introduced by Emil Post in his search for incomplete c.e. degrees (see Chapter 12). In particular, Post showed that all simple sets are incomplete for the many-one reduction.

**Definition 4.2.** A set  $A$  is *simple* if it is c.e., with an infinite complement, and if  $A \cap W_e \neq \emptyset$  for any infinite c.e. set  $W_e$ .  $\diamond$

In other words, a set  $A$  is simple if it is c.e. and its complement is infinite and immune.

**Proposition 4.3.** A simple set is not computable.  $\star$

PROOF. Let  $A$  be a simple set. Since the complement of  $A$  is infinite, and since  $A$  intersects any infinite c.e. set, the complement of  $A$  cannot be c.e. So according to Proposition 3-7.4,  $A$  cannot be computable.  $\blacksquare$

The following proposition gives a technique to build a simple set, which will be used several times in the study of the different randomness notions, and in particular to build a non-computable K-trivial set.

**Proposition 4.4.** There exists a simple set.  $\star$

PROOF. Let  $(W_e)_{e \in \mathbb{N}}$  be an enumeration of the c.e. sets. We enumerate a simple set  $A$ . At the computation step  $t$ , for any  $e \leq t$  such that  $A$  does not yet intersect  $W_e$  at step  $t$ , we check if there is an element  $x \in W_e[t]$  such that  $x > 2e$ . If this is the case we enumerate  $x$  into  $A$ .

It is clear that if  $W_e$  is infinite then  $A \cap W_e \neq \emptyset$ . Let us now show that  $\mathbb{N} \setminus A$  is infinite. For any  $W_e$  we enumerate at most one element in  $A$ . Moreover for any integer  $n = 2e$ , only the sets  $W_a$  for  $a < e$  can enumerate an element smaller than  $n$  into  $A$ . There are therefore at most  $n/2$  elements smaller than  $n$  into  $A$ , and this, for all  $n$ . So the complement of  $A$  is infinite.  $\blacksquare$

## 4.2. Solovay's theorem

We are now ready to show that K-trivial sets do not coincide with C-trivial sets, in other words there exists non-computable K-trivial sets.

### Theorem 4.5 (Solovay [211])

*There exist non-computable c.e. K-trivial sets.*

PROOF. The reader can use Figure 4.6 to follow the proof. Let  $(W_e)_{e \in \mathbb{N}}$  be an enumeration of the c.e. sets. We are going to build a simple and K-trivial set  $A$ , with the same technique as that of Proposition 4.4's proof. In order to guarantee that  $A$  is K-trivial we will create ourselves our machine  $M$  such that  $K_M(A \upharpoonright_n) \leq^+ K(n)$  for all  $n$ . For this, we will use a bounded requests set  $L$ .

We will denote by  $A_t$  and  $L_t$  the results of the enumerations of  $A$  and  $L$  at step  $t$ . At the computation step 0 the sets  $A_0$  and  $L_0$  are empty. At the stage of computation  $t + 1$  one proceeds as follows:

- (1) For any  $\langle n, \tau \rangle \leq t + 1$ , if  $t + 1$  is the smallest such that  $U(\tau)[t + 1] = n$  then we enumerate  $(A_t \upharpoonright_{n+1}, |\tau| + 1)$  into  $L$  at step  $t + 1$ .
- (2) Then for any  $e \leq t + 1$  such that  $A_t$  does not intersect  $W_e[t + 1]$ , we check if there exists  $n \geq 2e$  such that  $n \in W_e[t + 1]$  and such that

$$\sum_{\substack{U(\tau)[t+1] \downarrow = m \\ \text{with } m \geq n}} 2^{-|\tau|} < 2^{-e-2}.$$

If we find such an integer  $n$  then we enumerate it into  $A$  at step  $t + 1$ . Then for all  $\tau$  such that  $U(\tau)[t + 1] = m$  for  $m \geq n$ , we enumerate  $(A_{t+1} \upharpoonright_{m+1}, |\tau|)$  into  $L$  at step  $t + 1$ .

This completes the construction.

Let us first make sure that  $L$  is indeed a bounded requests set. The weight of  $L$  which comes from (1) is necessarily less than  $1/2$ , because in  $L$  we increase by 1 the length of each description coming from  $U$ . Finally for each  $e$ , the weight of  $L$  which comes from (2), that is to say, from what we add after having enumerated in  $A$  an element of  $W_e$ , is necessarily less than  $2^{-e-2}$ , by construction. The total weight for all integers  $e$  is therefore less than  $\sum_{e \in \mathbb{N}} 2^{-e-2} = 1/2$ . So the total weight of  $L$  is bounded by 1, which implies that  $L$  is a bounded requests set.

According to the KC theorem 3.3 we can now define a machine  $M$  such that  $(\sigma, l) \in L$  implies  $K_M(\sigma) \leq l$ . It is clear by construction that  $K(n) \leq l$  implies  $K_M(A \upharpoonright_n) \leq l + 1$ . Indeed each time a description of length  $l$  is discovered for  $n$  at step  $t$  we make sure that we will have a description of length  $l + 1$  for  $A_t \upharpoonright_n$ . In addition, each time  $A_{t+1} \neq A_t$  because of the enumeration of an element  $n$ , one updates the set  $L$  making sure that each string  $A_t \upharpoonright_m$  for  $m \geq n$  will have descriptions of the same length as all descriptions for  $m$  in the current version of  $U$ .

It remains to show that  $A$  is simple. By the usual argument (see Proposition 4.4)  $A$  is co-infinite because we enumerate at most one element in  $A$  for each  $W_e$  and this element is greater than  $2e$ . Now let  $e$  be such that  $W_e$  is infinite. We have  $\sum_{U(\tau) \downarrow} 2^{-|\tau|} < 1$  which implies that there exists  $n$  sufficiently large such that  $\sum_{U(\tau) \downarrow \geq n} 2^{-|\tau|} < 2^{-e-2}$ . So if  $W_e$  is infinite, we will end up finding such an integer  $n$  and enumerating it in  $A$ . ■

It is clear that if  $A$  is K-trivial, so is  $\overline{A}$ . There are therefore  $\Delta_2^0$  K-trivials which are not c.e.

**Exercise 4.7. (★)** Show that the class of K-trivial sets is closed under Turing join: if  $A_0$  and  $A_1$  are K-trivial then  $A_0 \oplus A_1$  is also K-trivial. ◇

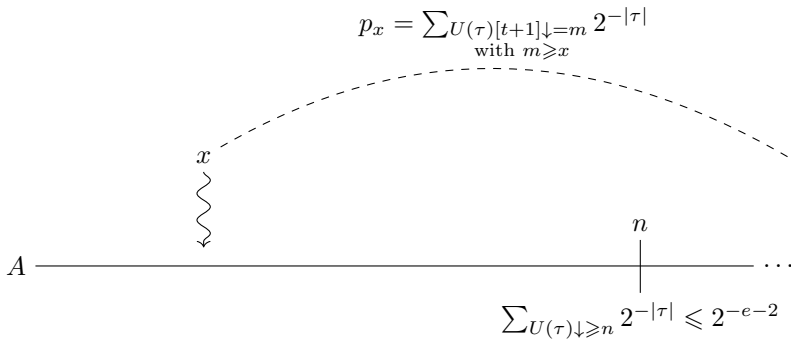


Figure 4.6: Illustration of theorem 4.5's proof. The addition of  $x$  into  $A$  at step  $t + 1$  implies a potential loss of  $p_x$ , coming from the weights that have been put on prefixes of  $A$  of length  $m$  greater than  $x$  (to match  $K(m)$ ). There necessarily exists an integer  $n$  sufficiently large such that this loss is bounded by  $2^{-e-2}$  at any computation step  $t$ .

### 4.3. Chaitin's theorem

One might be tempted at first to see Solovay's result as a failure of prefix-free complexity. The plain complexity gives a clean and neat characterization of the  $C$ -trivial sets, which therefore seems to lose of its elegance with  $K$ . Does this new class of sets deserve our attention? We will see that it does. The class of K-trivial sets is undoubtedly the richest and most studied of Algorithmic Randomness. The various works of many authors gave no less than ten different characterizations of it. We will therefore devote Chapter 20 entirely to this concept.

For the moment, we simply show the result of Chaitin, central for Chapter 20 to come: the K-trivial sets are all  $\Delta_2^0$ . The idea is the same as to show the  $C$ -trivial sets are computable.

**Theorem 4.8 (Chaitin [28])**

*There exists a constant  $c$  such that for all  $\sigma \in 2^{<\mathbb{N}}$  and  $d \in \mathbb{N}$  we have*

$$|\{\sigma : |\sigma| = n \wedge K(\sigma) \leq K(n) + d\}| < 2^c 2^d.$$

PROOF. Let  $U$  be our universal prefix-free machine and let  $M$  be the prefix-free machine which on  $\tau$  returns  $|U(\tau)|$ . Using Theorem 3.5 let  $c$  be a constant such that  $P_M(\sigma) < 2^{-K(\sigma)} 2^c$  for any string  $\sigma$ .

Suppose now that for a fixed  $n$  we have more than  $2^c 2^d$  strings  $\sigma$  of length  $n$  such that  $K(\sigma) \leq K(n) + d$ . Then, via  $M$ , we also have more than  $2^c 2^d$  distinct descriptions of  $n$  all of which of length less than or equal to  $K(n) + d$ .

This gives us  $P_M(n) \geq 2^c 2^d 2^{-K(n)-d} = 2^c 2^{-K(n)}$  which contradicts the choice of  $c$ . ■

**Corollary 4.9 (Chaitin [28])**

*The  $K$ -trivial sets are all  $\Delta_2^0$ .*

PROOF. Given that  $K \leq_T \emptyset'$ , for all  $d$  the class

$$\{X : \forall n \ K(X \upharpoonright_n) \leq K(n) + d\}$$

is a  $\Pi_1^0(\emptyset')$  class. By Theorem 4.8, this class contains a finite number of elements. They are therefore all  $\emptyset'$ -computable by Corollary 8-3.8 relativized to  $\emptyset'$ . ■

# Chapter 17

## Borel classes, measure and computability

The Swedish logician, philosopher and mathematician Per Martin-Löf proposes a notion of randomness following to a very different paradigm from that of Chaitin/Levin: a random set should have no “atypical” property, i.e., no property which occurs with probability 0, given independent draws of an infinite sequence of bits, via a random process which produces at each step 0 or 1 with probability  $1/2$ .

The mathematical formalization of probability theory was born in the 20th century, with the work of Kolmogorov, who published in 1933 his manual *Foundations of probability theory*. Kolmogorov relies for this on the early 20th century work of Borel and Lebesgue on Measure Theory and Integration. Measure theory applied to what we call “effective Borel classes” is the central concept that we will use in our study of Algorithmic Randomness.

### 1. A little history

We are in August of the year 1900. Mathematicians from all over the world converge to Sorbonne University where their second international congress is being held. The “Belle Époque” is at its best in the heart of a peaceful Paris. The Eiffel Tower was erected there a year earlier, Sarah Bernhardt at the height of her glory performs at the Renaissance theater, the first metro line is in service, the Lumière brothers are showing their first films on the Champ de Mars, on which the big wheel of Paris has just been built, on

the occasion of the fifth World's fair which welcomed more than 50 million visitors. The cultural and technical upheavals marking the beginning of the 20th century echoes the mathematical ones.

### 1.1. The 1900 International Congress

Émile Borel, then a young lecturer at the ENS school, is attending the international congress. He listens to Henry Poincaré who discusses with talent of this emerging discipline that is mathematical logic and the philosophical questions arising from it. Unlike some of his contemporaries, Poincaré is not fundamentally opposed to logic, but he defends a cautious attitude towards it, somehow marked by a certain lucidity [176]: *“By becoming rigorous, mathematical science takes on an artificial character that will strike anyone; it forgets its historical origins; we see how questions can be resolved, we no longer see how and why they arise”*. Poincaré, also a physicist, considers that mathematics should not stray away from physics, through which they keep a necessary connection with reality. This point of view appears more clearly a few years later in his work “Science and Method”, in 1908 [177]: *“Logic sometimes breeds monsters. A whole host of bizarre functions arose which seemed to attempt at resembling as little as possible the honest functions which served some purpose. No more continuity, or continuity, but no derivatives [...] In the past, new functions were invented with a view to some practical goal; today, we invent them on the sole purpose of defeating our fathers' reasoning, and that is all we will ever get from them”*.

Poincaré's viewpoint is opposed to that of the other mathematical giant of the time: the mathematician David Hilbert, for whom mathematics are a self-sufficient collection of abstract concepts. If this second international congress of mathematics has remained in history, it is undoubtedly due to the conference given there by Hilbert, during which he sets out a list of 23 unsolved mathematical problems, which shall greatly influence the research of the coming century. Hilbert is well acquainted with Cantor's work on infinity, and is undoubtedly one of the most fervent supporters of these new mathematics, to which logic alone and not reality gives its existence. Also is it quite naturally that Hilbert places as problem number one that of the continuum hypothesis: “Does there exist an infinite strictly comprised between that of the integers and that of the real numbers”? It was in this context that Borel began his career.

### 1.2. The French trio

Son of a Protestant pastor, Émile Borel was born in Saint-Affrique in 1871. He was a brilliant and versatile student. At the age of 18, he was ranked first at the École Polytechnique entrance exams, as well as at those of École Normale Supérieure, in which he entered in 1889. He was then received first in the aggregation of mathematics in 1893, then became a lecturer in Lille, then in Paris a few years later. He quickly became interested in Cantor's work, and showed his talent and creativity, by successfully using it for the study of functions. He will also introduced his students to it, among which two of them showed exceptional aptitudes: Henri Lebesgue and René Baire.



Émil Borel, 1871–1956

Both from very poor families, they shall emerge from their social condition through their work and their mathematical talent. Borel and his two pupils, the French trio, will mark all three a major advance in mathematics, through an ingenious use of the notions of “enumerable” and of “transfinite numbers” developed by Cantor, and which will lead to the discovery of Baire categories, of which we have already spoken in Section 10-2.2, to that of the so-called Borel sets, and to measure theory and integration. It is these latter two discoveries that are discussed in this chapter.

### 1.3. Borel's point of view

The remarkable research of our French trio will know its limits. Over the course of his career, Borel shows less and less confidence in Set Theory, which at the time is still poorly formalized, and subject to paradoxes. Borel will write towards the end of his life [106, p.59] “*I was extremely seduced from the age of 20, by reading Cantor's works [...] Georg Cantor brought to the study of mathematics that romantic spirit which is one of the most alluring sides of the German soul*”. This attraction will decline over time, and Borel will gradually adopt a point of view close to that of Poincaré regarding Set Theory, while adding a constructivist touch to it, a natural idea for him since his first mathematical work, in essence: the objects we manipulate must be explicitly describable. Only these objects can be guaranteed a certain form of existence, the rest being only inaccessible abstractions. Borel uses in fact, since 1898 in his “Lesson on the theory of functions” the word *effective* in a sense close to what we call today

“computable”, concept still not formalized at the time, but in which Borel — precursor — already detected a fundamental importance. He was thus the first to define computably enumerable sets, in order to solve Berry’s paradox (which we have already presented in Section 16-1.3): “Let  $n$  be the smallest number which is undefinable in less than 50 words”. Borel proposes a new approach for the resolution of this paradox [22]: if we consider the list of sentences written in English which define a number in a unique way, there is however no algorithm able to find which number is defined by a sentence. Borel then makes the first distinction between computably enumerable sets, which an algorithm can put in bijection with  $\mathbb{N}$ , and simply countable sets, for which one cannot necessarily compute a bijection with  $\mathbb{N}$ . This example shows Borel’s strong taste for mathematical constructivism. Arnaud Denjoy also tells about Borel [106, p.56]: “*The number had for him an almost carnal reality. He denied the quality of existence to most immeasurable numbers, the intervention of which is nevertheless essential to found our general theories. He demanded from mathematics a clarity, a Cartesian evidence, that is to say that they should be as close to the sensual as to the rational*”.

#### 1.4. The end of the trio

Borel will gradually lose his interest in Set Theory, which he undoubtedly considers too abstract. It must be said that mathematics is far from being his only field of activity. He is curious about everything, and has no trouble replacing his youthful passion. He created in 1906 with his wife the “Revue du Mois”, a scientific and literary journal, he became director in 1910 of the École normale supérieure, volunteered in 1914 during the First World War and quickly took many political responsibilities, first deputy, then chief of staff at the ministry of war and responsible for the technical services of the armament. He will finally be Minister of the Navy for a short time after the war. In 1936, he participated in the creation of the State Research Organization, which later became the CNRS. He will also be an active member of the resistance during the Second World War, and will publish throughout his life many works of mathematics and popularization. Borel was not bored, far from it, and probably not only stopped his mathematical research activity for lack of time, but also because he wanted to distance himself from what he called the “high mathematics”. He thus declared to his friend Paul Valéry in 1914 [106, p.83] that he was afraid of the consequences on his mind of research in Set Theory “because of the fatigue it causes and the serious disturbances whoever works on it shall fear”. The other two members of our trio had their own problems, and also stopped their work.

Baire is not in very good health and suffers from jealousy towards his colleague Henry Lebesgue, whose work — unlike his own — almost instantly had an international impact. Baire feels aggrieved at the fact that Lebesgue quickly occupies prestigious positions at the Sorbonne and at the College de France, while he must be content with the Dijon’s faculty of sciences<sup>1</sup>. Baire’s physical and psychological health problems prevent him from working for long periods of time, further leading to financial difficulties. Added to this is the frustration of what he sees as a lack of recognition of his scientific merits, which plunges him into serious depression. In 1914 he obtained sick leave which lasted until 1932, when he committed suicide in a hotel near Lake Geneva.

Lebesgue does not have Baire’s physical and psychological problems, but he too suffers from jealousy: towards his mentor Émile Borel. Lebesgue contests to Borel the paternity of certain ideas on measure theory and the relation between them deteriorates, as evidenced by this letter of 1917 addressed to Borel: *“I told you ... I no longer have the same confidence in you as I used to ... For the moment any relationship that comes out of the banality of comradeship would be hypocrisy on my part. I wouldn’t have lunch with you, but with old memories instead ... The kind of relationship we have had for a year has made you feel my state of mind enough so that this letter does not surprise you too much. I believe, however, that it will cause you some pain and I have kept too much hidden friendship for you not to be sorry myself”*. Thus, the creative impetus of the Set Theory French school gradually died out. It is a few thousand kilometers from Paris that the torch will be taken up, with the birth of the very famous Moscow school. This is another story, which we will discuss in Section 29-1.

## 2. First intuitions about Algorithmic Randomness

Let’s try to explain a little what is meant by *measure* in mathematics. One of the goals of Measure Theory is to formalize the notion of “size” of the parts of a set. Consider for example the line of reals. We have a good intuition of what the measure of an interval  $[a, b]$  is, because this corresponds to what we would actually measure in practice with a graduated rule: it is the real  $b - a$ . We will then write  $\lambda([a, b]) = b - a$ , the function  $\lambda$  denoting the so called “standard” measure, which we generally also call the *Lebesgue measure*. Given a union of two disjoint intervals  $[a, b] \cup [c, d]$  it seems just

---

<sup>1</sup>We transcribe here Baire’s state of mind, without prejudging in any way of Burundy’s University merits, of which we have no doubt.

as natural to define

$$\lambda([a, b] \cup [c, d]) = \lambda([a, b]) + \lambda([c, d]) = (b - a) + (d - c)$$

On the other hand, it is much less clear what is meant by the measure of an arbitrary set of reals  $I \subseteq \mathbb{R}$ . We will see that as soon as our notion of measure satisfies some reasonable properties, for example the measure of a disjoint union is the sum of the measures of the sets in the union, there necessarily exist non-measurable sets.

In our case, we are going to define a measure on subsets of the Cantor space. As we saw in Section 2-6, Cantor space can be seen as the interval  $[0, 1]$  by seeing any infinite binary sequence  $X \in 2^{\mathbb{N}}$  as the binary representation of the real  $0.X$ . In particular, we will define  $\lambda(2^{\mathbb{N}}) = \lambda([0, 1]) = 1$ . The rule that the measure of disjoint unions is equal to the sum of the measure of its elements also works for countable unions: for example

$$\begin{aligned} \lambda\left(\bigcup_{n \in \mathbb{N}} [2^{-2n-1}, 2^{-2n}]\right) &= \sum_{n \in \mathbb{N}} (2^{-2n} - 2^{-2n-1}) = \sum_{n \in \mathbb{N}} 2^{-2n} (1 - 2^{-1}) \\ &= 2^{-1} \sum_{n \in \mathbb{N}} 2^{-2n} = 1/2(1/(1 - 1/4)) = 2/3 \end{aligned}$$

Each open class of the Cantor space therefore has a measure: an open class  $\mathcal{U} \subseteq 2^{\omega}$  is of the form  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$ . Note that we can always choose  $W$  in such a way that it is prefix-free: if  $\sigma \preceq \tau$  and  $\sigma, \tau \in W$  then we can always remove  $\tau$  from  $W$  because  $[\tau] \subseteq [\sigma]$ . For such a prefix-free set  $W$ , we therefore have  $\lambda(\mathcal{U}) = \sum_{\sigma \in W} 2^{-|\sigma|}$ .

The rule of the measure of disjoint unions gives us at the same time a measure of closed classes: if  $\mathcal{C} \subseteq 2^{\mathbb{N}}$  is closed, then  $\lambda(\mathcal{C}) + \lambda(2^{\mathbb{N}} \setminus \mathcal{C}) = \lambda(2^{\mathbb{N}})$ . As  $\lambda(2^{\mathbb{N}}) = 1$ , we have  $\lambda(\mathcal{C}) = 1 - \lambda(2^{\mathbb{N}} \setminus \mathcal{C})$  where  $2^{\mathbb{N}} \setminus \mathcal{C}$  is an open class.

### 2.1. $\Pi_2^0$ classes

In order to be able to define Martin-Löf randomness, it is necessary to extend the definition of measure beyond closed and open classes. As an example, let us consider the law of large numbers: a random sequence should have as a property that the limit of the frequencies of 0 and 1 among its prefixes, exists and tends towards  $1/2$  when the length of the prefixes tends towards  $+\infty$ . The class  $\mathcal{A}$  of sequences which do not satisfy this property is the class of sequences  $X$  such that for a given  $\varepsilon > 0$ , for any  $n \in \mathbb{N}$  there exists  $m \geq n$  such that the frequency of 0 in the prefix of  $X$  of length  $m$  exceeds  $1/2 + \varepsilon$  or is below  $1/2 - \varepsilon$ . For  $\varepsilon$  fixed we can

describe these classes as follow:  $\mathcal{A}_\varepsilon = \bigcap_n \mathcal{U}_{\varepsilon,n}$  where

$$\mathcal{U}_{\varepsilon,n} = \bigcup_{m \geq n} [C_{\varepsilon,m}] \quad \text{and} \quad C_{\varepsilon,m} = \left\{ \sigma \in 2^m : \left| \frac{\#\{i \leq m : \sigma(i)=0\}}{m} - \frac{1}{2} \right| > \varepsilon \right\}$$

where  $[C_{\varepsilon,m}]$  denotes the open class described by the set  $C_{\varepsilon,m} \subseteq 2^{<\mathbb{N}}$ , where  $||$  denotes the absolute value and  $\#$  the size of a set. It is clear that each set  $\mathcal{U}_{\varepsilon,n}$  is a  $\Sigma_1^0$  class: an effectively open class. So each  $\mathcal{A}_\varepsilon$  is a countable intersection of effectively open classes.

We will call  $\Pi_2^0$  classes the effective intersections of effectively open classes. What is the measure of the  $\Pi_2^0$  class  $\mathcal{A}_\varepsilon = \bigcap_n \mathcal{U}_{\varepsilon,n}$ ? It is possible to show that the class  $\mathcal{A}_\varepsilon$  is neither open nor closed: it is of a higher complexity. On the other hand, according to Fact 10-2.2 each step of the countable intersection is an open one, that is to say for all  $n$  the class  $\mathcal{A}_{\varepsilon,n} = \bigcap_{m \leq n} \mathcal{U}_{\varepsilon,m}$  remains an open class and even a  $\Sigma_1^0$  class (see Lemma 3.5). It follows that each class  $\mathcal{A}_{\varepsilon,n}$ , as an open one, is measurable: the measure  $\lambda(\mathcal{A}_{\varepsilon,n})$  is well defined. Moreover  $\mathcal{A}_{\varepsilon,n+1} \subseteq \mathcal{A}_{\varepsilon,n}$ . At each step the measure can therefore only decrease and as it cannot fall below 0 it necessarily has a limit. We can therefore logically define

$$\lambda\left(\bigcap_n \mathcal{U}_{\varepsilon,n}\right) = \lim_{n \rightarrow +\infty} \lambda(\mathcal{A}_{\varepsilon,n}) = \lim_{n \rightarrow +\infty} \lambda\left(\bigcap_{m \leq n} \mathcal{U}_{\varepsilon,m}\right)$$

## 2.2. Link with probabilities

The notions of measure, of  $\Pi_2^0$  class and other classes of higher complexity, were formally defined and studied by Borel and Lebesgue at the beginning of the 20th century. If Lebesgue mainly uses Measure Theory for his theory of integration, Borel very early understood the links between Measure Theory and probability, which he used for example in 1909 to show that the law of large numbers is satisfied with probability 1 by sets of integers [23]. However, it was not until the work of Kolmogorov in 1933 that Measure Theory was used in all generality for a formal axiomatization of probability theory [10].

The analogy between measure and probability is simple: suppose we randomly flip bits, using a process that gives 0 with probability 1/2 and 1 with probability 1/2. This process gradually generates a sequence  $Z \in 2^{\mathbb{N}}$ . The measure of a class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is simply the probability that  $Z$  belongs to  $\mathcal{B}$ . Martin-Löf's idea will therefore be to say that a “random” sequence should not belong to any measure 0 class, that is to say to no “atypical” class. Obviously each set  $Z$  belongs to the class  $\{Z\}$  which is of measure 0 — indeed we can easily verify that the complement of  $\{Z\}$  is an open set of measure 1 — which is problematic for this intuition: no set is then random.

Martin-Löf then proposes to select only some of these classes: those which are  $\Pi_2^0$  and whose measure effectively converges to 0.

### 3. Borel classes

We start by presenting the classes on which the measure is naturally defined, as described in the previous section. This is the Borel hierarchy, named after Émile Borel. This hierarchy is written in boldface to distinguish it from its effective version (lightface) that will be defined later. A “ $\sim$ ” is sometimes added below the Borel hierarchy notation to avoid confusion with its lightface counterpart — this is what is done in this text.

**Definition 3.1 (Borel Hierarchy).** A class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is  $\Sigma_1^0$  if it is open and  $\Pi_1^0$  if it is closed. We then define inductively on  $n > 1$ :

1. A class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is  $\Sigma_{n+1}^0$  if it is a countable union of  $\Pi_n^0$  classes.
2. A class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is  $\Pi_{n+1}^0$  if it is a countable intersection of  $\Sigma_n^0$  classes.

A class is  $\Delta_n^0$  if it is both  $\Sigma_n^0$  and  $\Pi_n^0$ . ◇

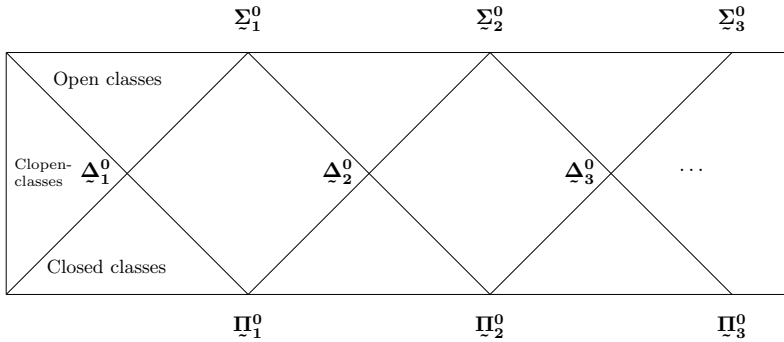


Figure 3.2: Borel hierarchy

Note that the  $\Sigma_n^0$  classes are the complements in  $2^{\mathbb{N}}$  of the  $\Pi_n^0$  classes (this comes from the equality  $2^{\mathbb{N}} \setminus \bigcup_n \mathcal{A}_n = \bigcap_n (2^{\mathbb{N}} \setminus \mathcal{A}_n)$ ). The Borel hierarchy is in a way a hierarchy on sets “shape-complexity”. It is clear the  $\Sigma_n^0$  classes (resp.  $\Pi_n^0$ ) are also  $\Pi_{n+1}^0$  (resp.  $\Sigma_{n+1}^0$ ) because we can always write  $\mathcal{A} = \mathcal{A} \cup \mathcal{A} \cup \mathcal{A} \dots$  or  $\mathcal{A} = \mathcal{A} \cap \mathcal{A} \cap \mathcal{A} \dots$ . We will see with Lemma 3.12 that the  $\Sigma_n^0$  classes are also  $\Sigma_{n+1}^0$  and the  $\Pi_n^0$  classes are also  $\Pi_{n+1}^0$ . We will also see that the Borel hierarchy is strict: in the same

way that an open class is not necessarily closed, a countable intersection of open classes cannot necessarily be described as a countable union of closed classes, and so on.

Let us remember that the open and closed classes have an effective analogue, which we denote by  $\Sigma_1^0$  and  $\Pi_1^0$  respectively. We will now see how to extend this correspondence by defining the  $\Sigma_n^0$  and  $\Pi_n^0$  classes, corresponding to the effective  $\Sigma_n^0$  and  $\Pi_n^0$  classes: we simply ask for the unions/intersections to be uniform.

**Definition 3.3 (Effective Borel Hierarchy).** A  $\Sigma_1^0$  code for a  $\Sigma_1^0$  class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$  is an integer of the form  $\langle 1, 0, e \rangle$ , where  $W_e \subseteq 2^{<\mathbb{N}}$  describes  $\mathcal{U}$ , that is,  $\mathcal{U} = \bigcup_{\sigma \in W_e} [\sigma]$ . A  $\Pi_1^0$  code for its complement is given by  $\langle 1, 1, e \rangle$ . We define by induction on  $n > 0$ :

- (1) A class  $\mathcal{B}$  is  $\Sigma_{n+1}^0$  if there exists a c.e. set  $W_e \subseteq \mathbb{N}$  enumerating  $\Pi_n^0$  codes of classes such that  $\mathcal{B} = \bigcup_n \mathcal{B}_n$  where  $\mathcal{B}_n$  is the class corresponding to  $n$ -th code enumerated in  $W_e$ . A  $\Sigma_{n+1}^0$  code for  $\mathcal{B}$  is given by  $\langle n+1, 0, e \rangle$ .
- (2) A class  $\mathcal{B}$  is  $\Pi_{n+1}^0$  if there exists a c.e. set  $W_e \subseteq \mathbb{N}$  enumerating  $\Sigma_n^0$  codes of classes such that  $\mathcal{B} = \bigcap_n \mathcal{B}_n$  where  $\mathcal{B}_n$  is the class corresponding to  $n$ -th code enumerated in  $W_e$ . A  $\Pi_{n+1}^0$  code for  $\mathcal{B}$  is given by  $\langle n+1, 1, e \rangle$ .

A  $\mathcal{B}$  class is  $\Delta_n^0$  if it is both  $\Sigma_n^0$  and  $\Pi_n^0$ . ◇

Note the similarity between the  $\Sigma_{n+1}^0$  classes of the Cantor space and the  $\Sigma_{n+1}^0$  sets of integers. The latter can also be defined as countable unions of  $\Pi_n^0$  classes, via the usual correspondence union/existential quantification and intersection/universal quantification.

**Exercise 3.4.** Show that from a  $\Sigma_n^0$  code (resp.  $\Pi_n^0$  code) of a class  $\mathcal{A}$ , we can computably find a  $\Pi_n^0$  code (resp.  $\Sigma_n^0$  code) of  $2^{\mathbb{N}} \setminus \mathcal{A}$ . ◇

Let us see a small lemma similar to Propositions 5-1.7 and 5-1.6 on the arithmetic hierarchy for integers, which we will use implicitly in the what follows.

**Lemma 3.5.** The  $\Sigma_n^0$  classes are closed under finite intersections and effective countable unions. The  $\Pi_n^0$  classes are closed under finite unions and effective countable intersections. ★

PROOF. If a countable union of  $\Sigma_n^0$  classes is effective, we can easily compute a  $\Sigma_n^0$  code for this union. By complement (see exercise 3.4) the countable intersections of  $\Pi_n^0$  classes is a  $\Pi_n^0$  class.

Let us show that the  $\Sigma_1^0$  classes are closed under finite intersections. Let  $\mathcal{U}_0 = \bigcup_{\sigma \in W_0} [\sigma]$  and  $\mathcal{U}_1 = \bigcup_{\sigma \in W_1} [\sigma]$  be two  $\Sigma_1^0$  classes. Then,

$$\mathcal{U}_0 \cap \mathcal{U}_1 = \bigcup_{\sigma_0 \in W_0, \sigma_1 \in W_1} [\sigma_0] \cap [\sigma_1].$$

We can also describe the c.e. set of strings constituting the open class  $\mathcal{U}_0 \cap \mathcal{U}_1$  in the following way: one enumerates a string  $\sigma$  when it is found in  $W_i$  for  $i < 2$  and when a prefix of  $\sigma$  is enumerated in  $W_{i-1}$ . By complement the  $\Pi_1^0$  classes are closed under finite unions. Note that we can uniformly produce a code of  $\mathcal{U}_0 \cap \mathcal{U}_1$  from codes of  $\mathcal{U}_0$  and  $\mathcal{U}_1$ , and that we can do the same for unions of two closed classes.

Suppose now that the  $\Sigma_n^0$  classes are closed under finite intersections and that the  $\Pi_n^0$  classes are closed under finite unions. Let  $\mathcal{B}_0 = \bigcup_{n \in W_0} \mathcal{A}_{0,n}$  and  $\mathcal{B}_1 = \bigcup_{n \in W_1} \mathcal{A}_{1,n}$  be  $\Sigma_{n+1}^0$  classes where each  $\mathcal{A}_{i,n}$  is a  $\Pi_n^0$  class for  $i < 2$ . Then,  $\mathcal{B}_0 \cap \mathcal{B}_1 = \bigcup_{n_0 \in W_0, n_1 \in W_1} \mathcal{A}_{0,n_0} \cap \mathcal{A}_{1,n_1}$ . By induction hypothesis each class  $\mathcal{A}_{0,n_0} \cap \mathcal{A}_{1,n_1}$  is  $\Pi_n^0$  and one can uniformly obtain a  $\Pi_n^0$  code for it from  $\Pi_n^0$  codes of  $\mathcal{A}_{0,n_0}$  and  $\mathcal{A}_{1,n_1}$ . We can therefore easily find a code for the class  $\mathcal{B}_0 \cap \mathcal{B}_1$ . By complement (see exercise 3.4) the  $\Pi_{n+1}^0$  classes are closed under finite unions. ■

**Exercise 3.6.** Make sure the proof of Lemma 3.5 also comes with uniformity: given the codes of two  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) classes  $\mathcal{B}_1$  and  $\mathcal{B}_2$ , one can compute a  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) code of the class  $\mathcal{B}_1 \cap \mathcal{B}_2$  (resp.  $\mathcal{B}_1 \cup \mathcal{B}_2$ ). ◇

The previous lemma allows to generalize to Borel classes something which was seen in Section 8-2.1 for open and closed classes: the  $\Pi_{n+1}^0$  classes can always be considered decreasing: given  $\bigcap_n \mathcal{B}_n$  we can equivalently consider the class  $\bigcap_n (\bigcap_{m \leq n} \mathcal{B}_m)$ . Note that the usual classes — in general defined using formulas — tend to be effective. Let's see some examples:

**Example 3.7.** (1) The class of infinite sets is  $\Pi_2^0$ :

$$\{X : \forall n \exists m > n \ m \in X\} = \bigcap_n \{X : X(m) = 1 \text{ for } m > n\}.$$

By using Baire's categories we easily show this class is not  $\Sigma_2^0$ : indeed a closed class containing only infinite sets is necessarily of empty interior, and in the complement of a union of closed classes of empty interior, one easily finds an infinite set using Lemma 10-2.14.

(2) The class of sets which are other sets' Turing jumps is  $\Pi_2^0$ . We easily show (see Exercise 4-6.4) the existence of a functional  $\Phi_e$  such that for all  $X$ ,  $\Phi_e(X)$  is total and with  $\Phi_e(X') = X$ . Therefore we define

the set of jumps as:

$$\left\{ X : \forall n \left( X(n) = 1 \wedge \Phi_n(\Phi_e(X), n) \downarrow \vee (X(n) = 0 \wedge \Phi_n(\Phi_e(X), n) \uparrow) \right) \right\}$$

$$=$$

$$\bigcap_n \left( \{X : X(n) = 1 \wedge \Phi_n(\Phi_e(X), n) \downarrow\} \cup \{X : X(n) = 0 \wedge \Phi_n(\Phi_e(X), n) \uparrow\} \right)$$

which is a countable intersection of unions of an open and a closed class. According to forthcoming Lemma 3.12, the intersection of an open and a closed class is a  $\Pi_2^0$  class. By using the low basis theorem we can easily see that this class is not  $\Sigma_2^0$  since a low set is not the jump of another set. One can look at Exercize 3.10's correction for a proof this class is not  $\Sigma_2^0$ .

- (3) The class of sets of *positive upper density* — whose ratio of elements among their prefixes is infinitely often greater than a certain  $\varepsilon$  — is  $\Sigma_3^0$ :

$$\left\{ X : \exists \varepsilon > 0 \forall n \exists m > n \frac{\#\{i \leq m : X(i) = 1\}}{m} > \varepsilon \right\}.$$

There again one can consult Exercize 3.11's correction for a proof this class is not  $\Pi_3^0$ .

- (4) Any countable class  $\mathcal{B} = \{X_0, X_1, \dots\}$  is  $\Sigma_2^0$ . Indeed  $\mathcal{B} = \bigcup_n \{X_n\}$  where each  $\{X_n\}$  is a closed class.

The reader shall realize with the exercises 3.10 and 3.11 how difficult it is, in general, to separate Borel complexities. The used techniques are often sophisticated application of Baire's categories, via specific forcings.

The previous examples illustrate the fact that Borel class complexity really have to do with *shape* and not computational content: an effective intersection of effectively open classes may not be *of the shape* of a union of closed classes, regardless of the computational power available to define this union. This also leads us to the following consideration.

#### Oracle relativization

The effective Borel hierarchy can be relativized to an oracle  $X \in 2^{\mathbb{N}}$ , replacing c.e. by  $X$ -c.e. in Definition 3.3. Moreover, it is easy to show that a  $\Sigma_n^0$  class (resp.  $\Pi_n^0$  class) is  $\Sigma_n^0(X)$  (resp.  $\Pi_n^0(X)$ ) for some oracle  $X$ ,

which for example will encode the different unions/intersections of the different open/closed constituting the Borel class.

Let us now see other examples illustrating the relativization to an oracle:

**Example 3.8.** (1) The class of low sets is  $\Sigma_2^0(\emptyset'')$ : each low set is the unique point of a  $\Pi_1^0(\emptyset')$ . On the other hand  $\emptyset''$  is necessary to get uniformity in this union. The reader can consult Exercize 3.9 to be sure of it.

(2) The class of high sets is  $\Sigma_4^0(\emptyset'')$ . Let  $T_{e,n} \subseteq 2^{<\mathbb{N}}$  be the  $\emptyset''$ -computable set of strings  $\sigma$  such that  $\Phi_e(\sigma, n) \downarrow = \emptyset''(n)$ . The class can be described as:

$$\bigcup_e \bigcap_n \bigcup_{\sigma \in T_{e,n}} \mathcal{A}_{e,\sigma}$$

where  $\mathcal{A}_{e,\sigma}$  is the set

$$\{X : \forall i < |\sigma| [(\sigma(i) = 1 \wedge \Phi_i(X, i) \downarrow) \vee (\sigma(i) = 0 \wedge \Phi_i(X, i) \uparrow)]\}.$$

To see that this is indeed a description of the high sets we must use the fact that  $\sigma \preceq \tau$  implies  $\mathcal{A}_{e,\tau} \subseteq \mathcal{A}_{e,\sigma}$  and having  $\sigma, \tau$  incomparable implies  $\mathcal{A}_{e,\tau} \cap \mathcal{A}_{e,\sigma} = \emptyset$ . According to forthcoming Lemma 3.12, each class  $\mathcal{A}_{e,\sigma}$  is  $\Sigma_2^0$  (uniformly in  $e, n$  and  $\sigma$ ) as the union of an open and a closed class, which makes  $\Sigma_4^0(\emptyset'')$  the class of high sets.

**Exercize 3.9. (\*\*) Show the class of low sets is not  $\Sigma_2^0(\emptyset')$ .** ◇

**Exercize 3.10. (\*\*\*) Show the class of sets which are Turing jumps is not  $\Sigma_2^0$ .** ◇

**Exercize 3.11. (\*\*\*) Show the class of sets having *positive upper density* — whose ratio of elements among their prefixes is infinitely often greater than a certain  $\varepsilon$  — is not  $\Pi_3^0$ .** ◇

We end with a last lemma whose counterpart for the arithmetic hierarchy is obvious, but which requires a little work for the Borel hierarchy.

**Lemma 3.12.** The  $\Sigma_n^0$  or  $\Pi_n^0$  classes are also  $\Delta_{n+1}^0$ . ★

PROOF. It is clear by definition that the  $\Sigma_n^0$  classes are  $\Pi_{n+1}^0$  and that the  $\Pi_n^0$  classes are  $\Sigma_{n+1}^0$ . The  $\Sigma_1^0$  classes are  $\Sigma_2^0$  as an effective union of cylinders, each cylinder being a degenerate  $\Pi_1^0$  class. Note that from the code of a  $\Sigma_1^0$  class we can uniformly compute a  $\Sigma_2^0$  code of the same class. By

complement (see exercise 3.4) the  $\Pi_1^0$  classes can be uniformly transformed into  $\Pi_2^0$  classes. Suppose now that the  $\Sigma_n^0$  classes are uniformly  $\Sigma_{n+1}^0$  and that the  $\Pi_n^0$  classes are uniformly  $\Pi_{n+1}^0$ . Let  $\bigcup_n \mathcal{A}_n$  be a  $\Sigma_{n+1}^0$  class where each  $\mathcal{A}_n$  is  $\Pi_n^0$ . By induction hypothesis each  $\mathcal{A}_n$  can be uniformly transformed into a  $\Pi_{n+1}^0$  class and therefore  $\bigcup_n \mathcal{A}_n$  is  $\Sigma_{n+2}^0$ . By complement (see exercise 3.4) the  $\Pi_{n+1}^0$  classes are  $\Pi_{n+2}^0$ . ■

## 4. Lebesgue measure

We now see how to extend the Lebesgue measure — hitherto defined for open, closed and  $\Pi_2^0$  classes — to all Borel classes of the Cantor space, while keeping good additivity properties.

### Notation

We denote by  $\lambda$  the Lebesgue measure on the Borel classes of the Cantor space, i.e., such that  $\lambda([\sigma]) = 2^{-|\sigma|}$  for any cylinder  $\sigma$ .

### 4.1. Definition of the measure

The mathematician Caratheodory showed that the Lebesgue measure, defined simply on the cylinders  $[\sigma]$  as being  $2^{-|\sigma|}$ , uniquely extends on the Borel classes using the intuition given at the beginning of this chapter to measure the  $\Pi_2^0$  classes.

#### Theorem 4.1 (Caratheodory)

The measure  $\lambda$  such that  $\lambda([\sigma]) = 2^{-|\sigma|}$  for any cylinder  $[\sigma]$ , extends in a unique way on the Borel classes in a function which keeps countable additivity property : if  $(\mathcal{B}_n)_{n \in \mathbb{N}}$  is a sequence of pairwise disjoint Borel classes then:

$$\lambda\left(\bigcup_n \mathcal{B}_n\right) = \sum_n \lambda(\mathcal{B}_n).$$

The extension is defined by induction as follows: for an increasing (resp. decreasing) sequence of Borel classes  $\mathcal{B}_n \subseteq \mathcal{B}_{n+1}$  (resp.  $\mathcal{B}_{n+1} \subseteq \mathcal{B}_n$ ) we have  $\lambda(\bigcup_n \mathcal{B}_n) = \sup_n \lambda(\mathcal{B}_n)$  (resp.  $\lambda(\bigcap_n \mathcal{B}_n) = \inf_n \lambda(\mathcal{B}_n)$ ).

The concept of measure defined by Lebesgue in 1901 is initially different from that presented in Theorem 4.1 [135]:

“Consider a set of points of  $[a, b]$ ; one can in infinitely many of ways enclose these points in a countable infinity of intervals; the lower limit of the sum

of the lengths of these intervals is the measure of the set. A set  $E$  is said to be measurable if its measure increased by that of the set of points not forming part of  $E$  gives the measure of  $[a, b]$ . Here are two properties of these sets: an infinity of measurable sets  $E_i$  given, the set of points which are part of at least one of them is measurable; if the  $E_i$  pairwise have no common point, the measure of the set obtained is the sum of the measures of the  $E_i$ . The set of points common to all  $E_i$  is measurable.”

Note that Lebesgue does not speak of Borel classes, but instead of *measurable* classes (or measurable sets in the above terminology): he claims that if each class  $(E_i)_{i \in \mathbb{N}}$  is measurable then  $\bigcup_i E_i$  is measurable, as is  $\bigcap_i E_i$ . It also gives the property of countable additivity. But to define the measure of any class  $\mathcal{B}$ , he uses the fact that the measure of open — disjoint unions of intervals — is unambiguously defined for the human mind, to then define the measure of any class  $\mathcal{B}$  as being

$$\inf_{\sum_1^0 \text{ class } \mathcal{U} \text{ with } \mathcal{B} \subseteq \mathcal{U}} \lambda(\mathcal{U}).$$

Note that Lebesgue suggests the possibility that *not all classes are necessarily measurable*. he poses for the measurability of a class the condition that the computation of its measure is consistent with the intuition that the measure of  $\mathcal{B} \subseteq [a, b]$  plus the measure of  $[a, b] \setminus \mathcal{B}$  must correspond to the measure of  $[a, b]$ , that is  $b - a$ .

Caratheodory formalized Lebesgue’s intuition under the name of *outer measure*. It is indeed possible to construct — using the axiom of choice — classes which are *non-measurable* in the sense imagined by Lebesgue, that is to say such that their outer measure plus that of their complement in  $[0, 1]$  is not equal to 1. We are about to see however that Lebesgue’s definition match the one we gave for Borel classes, which are all measurable (see Theorem 4.4).

## 4.2. Measure properties

Let us summarize here the Lebesgue measure properties, which will be used regularly throughout this part.

### Measure properties

- Measure of base classes:  $\lambda([\sigma]) = 2^{-|\sigma|}$
- Measure of Borel classes (defined by induction):

$$\lambda\left(\bigcup_n \mathcal{B}_n\right) = \sup_n \lambda\left(\bigcup_{m \leq n} \mathcal{B}_m\right) \text{ and } \lambda\left(\bigcap_n \mathcal{B}_n\right) = \inf_n \lambda\left(\bigcap_{m \leq n} \mathcal{B}_m\right).$$

- Measure of complements:

$$\lambda(\mathcal{A} \setminus \mathcal{B}) = \lambda(\mathcal{A}) - \lambda(\mathcal{A} \cap \mathcal{B}).$$

- Additivity: For a sequence of pairwise disjoint measurable classes  $(\mathcal{A}_n)_{n \in \mathbb{N}}$  we have

$$\lambda\left(\bigcup_n \mathcal{A}_n\right) = \sum_n \lambda(\mathcal{A}_n).$$

- Sub-additivity: For a sequence of measurable classes  $(\mathcal{A}_n)_{n \in \mathbb{N}}$  we have

$$\lambda\left(\bigcup_n \mathcal{A}_n\right) \leq \sum_n \lambda(\mathcal{A}_n).$$

Note that sub-additivity is deduced from additivity: if  $\mathcal{A}_0$  and  $\mathcal{A}_1$  are not disjoint, we have on the other hand  $\mathcal{A}_0 \cup \mathcal{A}_1 = \mathcal{A}_0 \cup (\mathcal{A}_1 \setminus \mathcal{A}_0)$  with  $\mathcal{A}_0$  disjoint from  $\mathcal{A}_1 \setminus \mathcal{A}_0$ . By measure additivity we therefore have  $\lambda(\mathcal{A}_0 \cup \mathcal{A}_1) = \lambda(\mathcal{A}_0) + \lambda(\mathcal{A}_1 \setminus \mathcal{A}_0) \leq \lambda(\mathcal{A}_0) + \lambda(\mathcal{A}_1)$ . We are now going to show that any Borel class  $\mathcal{B}$  is measurable in the sense given by Lebesgue: if we consider the infimum of the measure of open classes containing  $\mathcal{B}$  and 1 minus the infimum of the measure of open classes containing the complement of  $\mathcal{B}$ , we reach the same number: the measure of  $\mathcal{B}$ . We will also show an effective version of this, and we need to study for that the computational complexity of measuring a Borel class.

### 4.3. Computation of the measure

We will show that the measure of a  $\Sigma_n^0$  class is a real which is left-c.e. relative to  $\emptyset^{(n-1)}$ , and in particular that the measure of a  $\Sigma_1^0$  class is a left-c.e. real. For that, we need a lemma about  $\Sigma_1^0$  classes which will often be used later without explicit reference. Let us remember Definition 16-2.1 of prefix-free sets. We then have:

**Lemma 4.2.** Let  $\mathcal{U} = [W]$  be a  $\Sigma_1^0$  class defined from the c.e. set  $W \subseteq 2^{<\mathbb{N}}$ . We can compute uniformly in a code of  $W$  the code of a prefix-free c.e. set  $W'$  such that  $\mathcal{U} = [W']$ . ★

**PROOF.** At the computation step  $t$ , if a new string  $\sigma$  enters  $W$ , if a prefix of  $\sigma$  already belongs to  $W'$  we do not enumerate  $\sigma$  in  $W'$ . Otherwise, if extensions  $\tau \succ \sigma$  are already in  $W'$ , then we compute  $n$ , the smallest length such that the extensions of  $\sigma$  in  $W'$  at this step are all of length less than  $n$  (which is possible because at each step  $W'$  only contains a finite number

of elements). We then enumerate in  $W'$  all the extensions of  $\sigma$  of length  $n$  which do not have a prefix in  $W'$ .

It is clear that  $W'$  is prefix-free and that we have  $[W] = [W']$ . ■

### Minimal prefix-free sets

A prefix-free set  $W$  is *minimal* if for any string  $\sigma$  and any integer  $n$ ,  $\sigma\tau \notin W$  for at least one string  $\tau$  of length  $n$ . Note that the previous lemma does not guarantee for the enumerated prefix-free set to be minimal: it is for instance possible for the prefix-free enumeration to contain strings  $\sigma 0$  and  $\sigma 1$ , which could instead be replaced by  $\sigma$ . Any open class  $\mathcal{U}$  is uniquely generated by a minimal prefix-free set  $W$ , but  $W$  need not to be c.e.

We can now move on to the announced proposition.

**Proposition 4.3.** Let  $n > 0$  and  $\varepsilon \in \mathbb{Q}^+$ . Let  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  be a  $\Sigma_n^0$  class. The real  $\lambda(\mathcal{B})$  is left-c.e. relative to  $\emptyset^{(n-1)}$  and uniformly in a code of  $\mathcal{B}$ . In particular  $\lambda(\mathcal{B})$  is  $\emptyset^{(n)}$ -computable uniformly into a code of  $\mathcal{B}$ . ★

PROOF. Let  $\mathcal{B} = \bigcup_{\sigma \in W} [\sigma]$  be a  $\Sigma_1^0$  class described from the c.e. set  $W$ . By Lemma 4.2 we can assume that  $W$  is prefix-free. Then,  $\lambda(\mathcal{B}) = \sum_{\sigma \in W} 2^{-|\sigma|}$ . It is therefore clear that  $\lambda(\mathcal{B})$  is a real which is left-c.e. uniformly in a code of  $\mathcal{B}$ .

Suppose the measure of any  $\Sigma_n^0$  class is  $\emptyset^{(n)}$ -computable uniformly in its code. Let  $\mathcal{B} = \bigcup_m \mathcal{A}_m$  be a  $\Sigma_{n+1}^0$  class with each  $\mathcal{A}_m$  a  $\Pi_n^0$  class. Let us set  $\mathcal{C}_m = \bigcup_{k \leq m} \mathcal{A}_k$ . As  $\mathcal{C}_m \subseteq \mathcal{C}_{m+1}$  we have  $\lambda(\mathcal{C}_m) \leq \lambda(\mathcal{C}_{m+1})$ . Also by definition we have  $\lambda(\bigcup_m \mathcal{A}_m) = \sup_m \lambda(\mathcal{C}_m)$ .

By Lemma 3.5 and Exercize 3.6 each class  $\mathcal{C}_m$  is  $\Pi_n^0$  and one can find a  $\Pi_n^0$  code for it uniformly in  $m$  and in a code of  $\mathcal{B}$ . As the measure of the complement of each class  $\mathcal{C}_m$  is  $\emptyset^{(n)}$ -computable, so is the measure of  $\mathcal{C}_m$  — as 1 minus the measure of the complement of  $\mathcal{C}_m$  — and moreover it is uniform in a code of  $\mathcal{C}_m$ . The measure  $\mathcal{B}$  is therefore the supremum of the sequence  $r_1 \leq r_2 \leq r_3 \leq \dots$  where each  $r_m = \lambda(\mathcal{C}_m)$ . As each  $r_m$  is a  $\emptyset^{(n)}$ -computable real uniformly in  $m$ , then  $\lambda(\mathcal{B}) = r = \sup_m r_m$  is a left-c.e. real relative to  $\emptyset^{(n)}$  and uniformly in a code of  $\mathcal{B}$ . ■

We are now ready to show the desired theorem, in its effective version.

### Theorem 4.4

For any  $\Sigma_n^0$  class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  and for any rational  $\varepsilon > 0$  there exists:

- (1) A  $\Sigma_1^0(\emptyset^{(n-1)})$  class  $\mathcal{U}$  such that  $\mathcal{B} \subseteq \mathcal{U}$  and such that  $\lambda(\mathcal{U} \setminus \mathcal{B}) \leq \varepsilon$   
 (2) A  $\Pi_1^0(\emptyset^{(n-1)})$  class  $\mathcal{F}$  such that  $\mathcal{F} \subseteq \mathcal{B}$  and such that  $\lambda(\mathcal{B} \setminus \mathcal{F}) \leq \varepsilon$

In addition, a code of  $\mathcal{U}$  can be computed uniformly in  $\varepsilon$  and in a code of  $\mathcal{B}$ , and a code of  $\mathcal{F}$  can be computed using  $\emptyset^{(n)}$  uniformly in  $\varepsilon$  and in a code of  $\mathcal{B}$ .

PROOF. Let  $\mathcal{B}$  be a  $\Sigma_1^0$  class. Then (1) is obvious. For (2) it suffices to consider the clopen approximations of  $\mathcal{U}$  at each computation step  $t$ , given by  $\mathcal{U}[t]$ . Using the halting set, we can compute for any  $\varepsilon$  the smallest  $t$  such that  $\lambda(\mathcal{U} \setminus \mathcal{U}[t]) < \varepsilon$ . Suppose the theorem holds for  $\Sigma_n^0$  classes and let us show that it holds for  $\Sigma_{n+1}^0$  classes. Let  $\mathcal{B} = \bigcup_m \mathcal{A}_m$  be a  $\Sigma_{n+1}^0$  class with each  $\mathcal{A}_m$  a  $\Pi_n^0$  class.

Let us show (1). The reader can use Figure 4.5 to follow the proof. The complement  $\overline{\mathcal{A}_m}$  of each  $\mathcal{A}_m$  is  $\Sigma_n^0$ . By induction hypothesis, according to (2) one can uniformly find for all  $m$  using  $\emptyset^{(n)}$  a  $\Pi_1^0(\emptyset^{(n-1)})$  class  $\mathcal{F}_m \subseteq \overline{\mathcal{A}_m}$  such that  $\lambda(\overline{\mathcal{A}_m} \setminus \mathcal{F}_m) \leq \varepsilon 2^{-m-1}$  for all  $m$ . Let  $\mathcal{U}_m$  be the complement of the closed class  $\mathcal{F}_m$ . We have in particular  $\lambda(\mathcal{U}_m \setminus \mathcal{A}_m) \leq \varepsilon 2^{-m-1}$  for all  $m$ . This gives us:

$$\begin{aligned} \lambda(\bigcup_m \mathcal{U}_m \setminus \bigcup_m \mathcal{A}_m) &\leq \lambda(\bigcup_m (\mathcal{U}_m \setminus \mathcal{A}_m)) \\ &\leq \sum_m \lambda(\mathcal{U}_m \setminus \mathcal{A}_m) \\ &\leq \sum_m \varepsilon 2^{-m-1} \\ &\leq \varepsilon \end{aligned}$$

Moreover each  $\mathcal{U}_m$  is a  $\Sigma_1^0(\emptyset^{(n-1)})$  class whose code can be computed in  $\emptyset^{(n)}$  uniformly. So  $\mathcal{U} = \bigcup_m \mathcal{U}_m$  is a  $\Sigma_1^0(\emptyset^{(n)})$  class.

Let us show (2). The reader can use Figure 4.6 to follow the proof. By induction hypothesis, from (1) we can find a  $\Sigma_1^0(\emptyset^{(n-1)})$  class  $\mathcal{U}_m$  such that  $\overline{\mathcal{A}_m} \subseteq \mathcal{U}_m$  and such that  $\lambda(\mathcal{U}_m \setminus \overline{\mathcal{A}_m}) \leq \varepsilon 2^{-m-2}$  for all  $m$ . Let  $\mathcal{F}_m$  be the complement of the open class  $\mathcal{U}_m$ . We then have  $\lambda(\mathcal{A}_m \setminus \mathcal{F}_m) \leq \varepsilon 2^{-m-2}$ . According to Lemma 3.5 each class  $\bigcup_{m \leq k} \mathcal{A}_m$  for  $k \in \mathbb{N}$  is  $\Pi_n^0$  and therefore each class  $\bigcup_m \mathcal{A}_m \setminus \bigcup_{m \leq k} \mathcal{A}_m$  for  $k \in \mathbb{N}$  is  $\Sigma_{n+1}^0$ . Using  $\emptyset^{(n+1)}$  we search using Proposition 4.3 for the smallest  $k$  such that  $\lambda(\mathcal{C}) < \frac{1}{2}\varepsilon$  for  $\mathcal{C} = \bigcup_m \mathcal{A}_m \setminus \bigcup_{m \leq k} \mathcal{A}_m$ . We then have:

$$\begin{aligned} \lambda(\bigcup_m \mathcal{A}_m \setminus \bigcup_{m \leq k} \mathcal{F}_m) &\leq \lambda(\mathcal{C}) + \lambda(\bigcup_{m \leq k} \mathcal{A}_m \setminus \bigcup_{m \leq k} \mathcal{F}_m) \\ &\leq \lambda(\mathcal{C}) + \lambda(\bigcup_{m \leq k} (\mathcal{A}_m \setminus \mathcal{F}_m)) \\ &\leq \frac{1}{2}\varepsilon + \frac{1}{2} \sum_{m \leq k} \varepsilon 2^{-m-1} \\ &\leq \varepsilon \end{aligned}$$

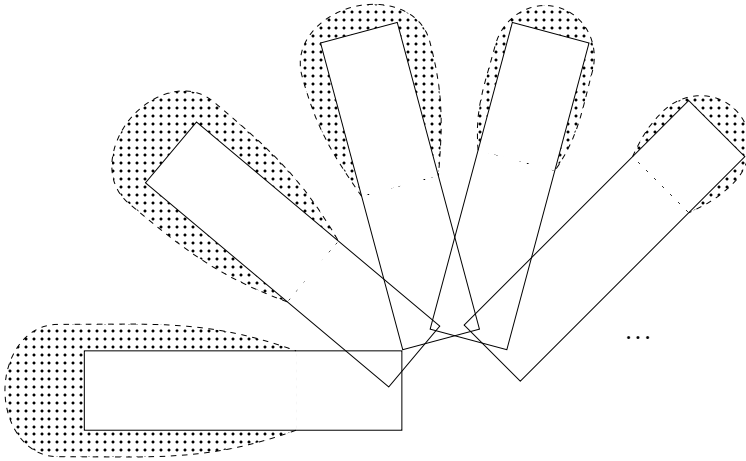


Figure 4.5: Illustration. The rectangles are the  $\Pi_n^0$  components of an increasing  $\Sigma_{n+1}^0$  class. The rounded dotted corners are the open classes containing each component. The gray part is the exceeding part, whose measure must decrease sufficiently fast.

As each  $\mathcal{F}_m$  is a  $\Pi_1^0(\emptyset^{n-1})$  class whose code can be found uniformly in  $m$ , then  $\mathcal{F} = \bigcup_{m \leq k} \mathcal{F}_m$  is also a  $\Pi_1^0(\emptyset^{n-1})$  class and in particular  $\Pi_1^0(\emptyset^n)$ , whose code is uniform in the bound  $k$  itself computable in  $\emptyset^{(n+1)}$ . ■

**Exercise 4.7. (★)** Show that the computational bound of Proposition 4.3 cannot be simplified: for all  $n$  there exists a  $\Sigma_n^0$  class  $\mathcal{B}$  such that the real  $\lambda(\mathcal{B})$  computes  $\emptyset^{(n)}$ . ◇

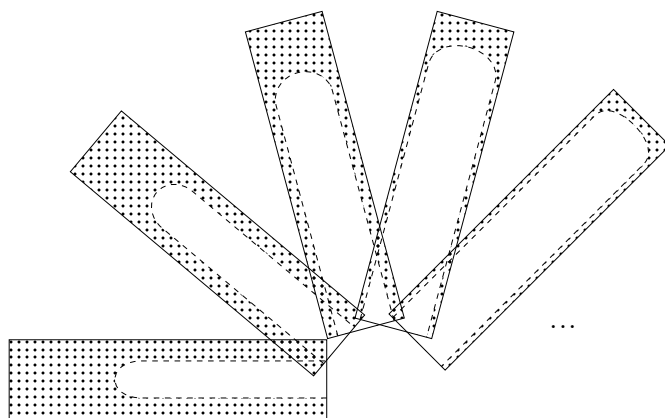


Figure 4.6: Illustration. The rectangles are the  $\Pi_n^0$  components of an increasing  $\Sigma_{n+1}^0$  class. The rounded dotted corners are the open classes containing each component. The gray part is the exceeding part, whose measure must decrease sufficiently fast. Moreover, it will not be necessary to take closed classes in each component: one can stop when the components of the  $\Sigma_{n+1}^0$  classes will add almost no measure. We thus have a finite union of closed classes which remains closed.



# Chapter 18

## Martin-Löf Randomness

We now have the necessary tools to define Martin-Löf randomness. The idea is to define a set  $X$  as being random if it has no “atypical” property, a property being atypical if the class of sets sharing it is of measure 0.

### 1. Intuitions and definitions

Let us consider our example on the law of large numbers from Section 17-2.1. We have defined for some  $\varepsilon > 0$  the class  $\mathcal{A}_\varepsilon = \bigcap_n \mathcal{U}_{\varepsilon,n}$  where

$$\mathcal{U}_{\varepsilon,n} = \bigcup_{m \geq n} [C_{\varepsilon,m}] \quad \text{and} \quad C_{\varepsilon,m} = \left\{ \sigma \in 2^m : \left| \frac{\#\{i \leq m : \sigma(i) = 0\}}{m} - \frac{1}{2} \right| > \varepsilon \right\}$$

where  $[C_{\varepsilon,m}]$  denotes the open class described by the set  $C_{\varepsilon,m} \subseteq 2^{<\mathbb{N}}$ . The class  $\mathcal{A}_\varepsilon$  contains all the sets  $X$  whose frequency of 1 among their prefixes is infinitely often above  $1/2 + \varepsilon$  or infinitely often below  $1/2 - \varepsilon$ . It is possible to show, using Hoeffding’s inequalities [92] that we have  $\lambda([C_{\varepsilon,m}]) \leq 2e^{-2m\varepsilon^2}$  for all  $m$ . Note that the classes  $([C_{\varepsilon,m}])_{m \in \mathbb{N}}$  are not necessarily disjoint, and let us recall here the *subadditivity* property: if  $(\mathcal{B}_n)_{n \in \mathbb{N}}$  is any sequence of measurable classes then  $\lambda(\bigcup_n \mathcal{B}_n) \leq \sum_n \lambda(\mathcal{B}_n)$ .

Let us set  $a_m = e^{-2m\varepsilon^2}$ . Then, for all  $n$ , using subadditivity, we have  $\lambda(\mathcal{U}_{\varepsilon,n}) \leq \sum_{m \geq n} 2a_m = 2a_n \times (1 + a_1^1 + a_1^2 + \dots)$ . The convergence formula for geometric series gives  $\lambda(\mathcal{U}_{\varepsilon,n}) \leq 2a_n/(1 - a_1)$ . When  $n$  goes to infinity the sequence  $2a_n/(1 - a_1)$  clearly converges to 0. It follows that  $\lambda(\mathcal{U}_{\varepsilon,n})$  gets closer and closer to 0 as  $n$  increases and therefore  $\lambda(\bigcap_n \mathcal{U}_{\varepsilon,n}) = 0$ .

For any  $\varepsilon$ , each class  $\bigcap_n \mathcal{U}_{\varepsilon,n}$  is therefore like a *test* that any  $X$  shall pass to be accepted as a random number: the test in question is a specific class of measure 0 in which no random worthy of the name can appear. Measure subadditivity implies that  $\bigcup_{\varepsilon \in \mathbb{Q}^+} \bigcap_n \mathcal{U}_{\varepsilon,n}$  is still a measure 0 class. A random number worthy of the name is therefore not in it. Martin-Löf's intuition is then the following: the important criteria for us mathematicians, who seek a randomness definition matching our intuition, can be captured by measure 0 classes whose complexity is that of each  $\bigcap_n \mathcal{U}_{\varepsilon,n}$ :  $\Pi_2^0$  classes. Finally, the key idea of Martin-Löf is the following: if we consider all the  $\Pi_2^0$  classes of measure 0, we are dealing with countably many tests that each random sequence must pass. On the other hand, with a simple restriction on the considered  $\Pi_2^0$  classes, it is possible to regroup them all in one unique test (we will see this with Corollary 2.2). It suffices to consider the decreasing  $\Pi_2^0$  classes of the form  $\bigcap_n \mathcal{U}_n$  such that  $\lambda(\mathcal{U}_n) \leq 2^{-n}$  for all  $n$ : we do not simply want  $\lambda(\bigcap_n \mathcal{U}_n)$  to equal 0, but we also want the convergence towards 0 to be achieved sufficiently quickly.

**Definition 1.1 (Martin-Löf [149]).** A *Martin-Löf test* is given by a decreasing  $\Pi_2^0$  class  $\bigcap_n \mathcal{U}_n$  such that  $\lambda(\mathcal{U}_n) \leq 2^{-n}$  for all  $n$ . A set  $Z \in 2^{\mathbb{N}}$  *passes* the test if  $Z \notin \bigcap_n \mathcal{U}_n$ . If  $Z$  does not pass the test it is *captured* by the test.  $\diamond$

Note the condition  $\lambda(\mathcal{U}_n) \leq 2^{-n}$  for a test  $\bigcap_n \mathcal{U}_n$  can be relaxed: it suffices that there exists a computable function  $f : \mathbb{N} \rightarrow \mathbb{Q}$  such that  $\lim_n f(n) = 0$  and such that  $\lambda(\mathcal{U}_n) \leq f(n)$  for all  $n$ . From this we can then define another  $\Pi_2^0$  class satisfying the above definition: given some  $n$  we can simply look for the  $n$ -th component  $\mathcal{U}_{m_n}$  such that  $\lambda(\mathcal{U}_{m_n}) \leq 2^{-n}$ . The class  $\bigcap_{m_n} \mathcal{U}_{m_n} = \bigcap_n \mathcal{U}_n$  is still  $\Pi_2^0$ .

**Definition 1.2 (Martin-Löf [149]).** A set  $Z$  is *Martin-Löf random* if it passes all the Martin-Löf tests.  $\diamond$

We have shown above that the class  $\bigcap_n \mathcal{U}_{\varepsilon,n}$ , capturing the sets whose frequency of 1 among their prefixes is infinitely often above  $1/2 + \varepsilon$ , or infinitely often below  $1/2 - \varepsilon$ , is such that  $\lambda(\mathcal{U}_{\varepsilon,n}) \leq 2a_n/(1-a_1)$  where  $a_n = e^{-2n\varepsilon^2}$ . Each  $\mathcal{U}_n$  is therefore bounded by a function which is computable in  $n$ , and which goes towards 0. So if  $Z$  is Martin-Löf random it cannot belong to any class  $\bigcap_n \mathcal{U}_{\varepsilon,n}$ ; and the frequency of 1 among its prefixes converges to  $1/2$ . We can generalize this result:

**Exercise 1.3. (★)** Generalize the example given at the start of the section to show that for any Martin-Löf random set  $X$ , for any string  $\sigma$  of length  $n$ , if  $\tau_0\tau_1\tau_2\cdots = X$  is the division of  $X$  in strings of length  $n$ , then

the frequency of  $i \leq m$  such that  $\tau_i = \sigma$  converges to  $2^{-|\sigma|}$  when  $m$  goes to  $+\infty$ .

Note: we can use Hoeffding's inequality for  $X$  written in base  $2^n$ . For any basis  $q$  and any  $a < q$ , Hoeffding's inequality gives  $\lambda([C_{\varepsilon, m}^{q, a}]) \leq 2e^{-2m\varepsilon^2}$  where

$$C_{\varepsilon, m}^{q, a} = \left\{ \sigma \in q^m : \left| \frac{\#\{i \leq m : \sigma(i) = a\}}{m} - \frac{1}{q} \right| > \varepsilon \right\} \quad \diamond$$

We immediately give another type of test, less restrictive in appearance but leading to the same randomness notion, and which will be useful from time to time.

**Definition 1.4.** A *Solovay test* is given by a computable sequence of  $\Sigma_1^0$  class  $(\mathcal{U}_n)_{n \in \mathbb{N}}$  such that  $\sum_n \lambda(\mathcal{U}_n)$  is finite. A set  $Z$  passes the Solovay test if  $Z$  only belongs to finitely many classes  $\mathcal{U}_n$ . Otherwise  $Z$  is captured by the test.  $\diamond$

### Theorem 1.5

A set is Martin-Löf random iff it passes all Solovay tests.

PROOF. One direction is trivial: any Martin-Löf test is a Solovay test. For the other direction suppose  $Z$  is captured by a Solovay test  $(\mathcal{U}_n)_{n \in \mathbb{N}}$ . There must exist  $m$  sufficiently large such that  $\sum_{n \geq m} \lambda(\mathcal{U}_n) < 1$ . Note that  $Z$  is still captured by  $(\mathcal{U}_n)_{n \geq m}$  and we can therefore consider  $\sum_n \lambda(\mathcal{U}_n) < 1$  without loss of generality.

We define  $\mathcal{V}_m$  as being the  $\Sigma_1^0$  class of sets  $X$  belonging at least to  $2^m$  distinct classes  $\mathcal{U}_{i_1}, \mathcal{U}_{i_2}, \dots, \mathcal{U}_{i_{2^m}}$ . Let  $W_m$  be the minimal prefix-free set describing  $\mathcal{V}_m$ . For each string  $\sigma \in W_m$ , we have  $[\sigma] \subseteq \mathcal{U}_{i_j}$  for at least  $2^m$  distinct classes  $\mathcal{U}_{i_1}, \mathcal{U}_{i_2}, \dots, \mathcal{U}_{i_{2^m}}$ . In particular  $2^m 2^{-|\sigma|} = \lambda(\mathcal{U}_{i_1} \cap [\sigma]) + \dots + \lambda(\mathcal{U}_{i_{2^m}} \cap [\sigma])$ . Since  $\sigma_1, \sigma_2 \in W_m$  implies  $[\sigma_1] \cap [\sigma_2] = \emptyset$ , then  $\sigma_1, \sigma_2 \in W_m$  implies  $(\mathcal{U}_{i_j} \cap [\sigma_1]) \cap (\mathcal{U}_{i_j} \cap [\sigma_2]) = \emptyset$  for all  $i$ . We therefore have  $2^m \sum_{\sigma \in W_m} 2^{-|\sigma|} \leq \sum_n \lambda(\mathcal{U}_n) < 1$ . Which gives  $2^m \lambda(\mathcal{V}_m) \leq 1$  and therefore  $\lambda(\mathcal{V}_m) < 2^{-m}$ . So  $\bigcap_m \mathcal{V}_m$  is a Martin-Löf test which by hypothesis on  $Z$  captures it.  $\blacksquare$

## 2. The Martin-Löf and Chaitin/Levin randoms coincide

Martin-Löf in his article [149] showed how to construct a *universal test*: a Martin-Löf test containing them all. A few years later Levin and Schnorr independently showed that this universal test could in fact be defined using Kolmogorov prefix-free complexity: random numbers in the sense of Chaitin/Levin are exactly those which are random in the sense of Martin-Löf. This result makes *robust* the notion of Martin-Löf randomness, in the sense that it has several characterizations which have a priori nothing to do with each other.

**Theorem 2.1 (Levin [139], Schnorr [192])**

*A set  $Z$  is Martin-Löf random iff it is Chaitin/Levin random.*

PROOF. Suppose  $Z$  is not Chaitin/Levin random. Then, for all  $c$  there exists  $n$  such that  $K(Z \upharpoonright_n) < n - c$ . So  $Z$  belongs to the set  $\bigcap_c \mathcal{U}_c$  where  $\mathcal{U}_c = \{X : \exists n K(X \upharpoonright_n) < n - c\}$ . Note that each  $\mathcal{U}_c$  is a  $\Sigma_1^0$  class uniformly in  $c$  and therefore  $\bigcap_c \mathcal{U}_c$  is a  $\Pi_2^0$  class. For each  $c \in \mathbb{N}$ , let  $W_c$  be a prefix-free set such that  $[W_c] = \mathcal{U}_c$  and such that for all  $\sigma \in W_c$ ,  $K(\sigma) < |\sigma| - c$ . Note that  $W_c$  is not necessarily c.e. We therefore have  $\sum_{\sigma \in W_c} 2^{-|\sigma|+c} \leq \sum_{\sigma \in W_c} 2^{-K(\sigma)}$ . Since  $K$  is the prefix-free complexity we have  $\sum_{\sigma \in W_c} 2^{-K(\sigma)} \leq 1$ . So  $\lambda(\mathcal{U}_c) = \sum_{\sigma \in W_c} 2^{-|\sigma|} \leq 2^{-c}$ . It follows that  $\bigcap_c \mathcal{U}_c$  is Martin-Löf test and therefore that  $Z$  is not Martin-Löf random.

Suppose now  $Z$  is not Martin-Löf random. Then, there exists a  $\Pi_2^0$  class  $\bigcap_n \mathcal{U}_n$  such that  $\mathcal{U}_{n+1} \subseteq \mathcal{U}_n$ , such that  $\lambda(\mathcal{U}_n) \leq 2^{-n}$  and for which  $Z \in \bigcap_n \mathcal{U}_n$ . Let  $W_n \subseteq 2^{<\mathbb{N}}$  be a prefix-free  $\Sigma_1^0$  set such that  $\mathcal{U}_n = \bigcup_{\sigma \in W_n} [\sigma]$ . We define for any  $c$  the bounded requests set  $L_c$  by enumerating  $(\sigma, |\sigma| - c + 1)$  into  $L_c$  for all  $\sigma \in W_{2^c}$ . The weight of  $L_c$  is then equal to

$$\sum_{\sigma \in W_{2^c}} 2^{-|\sigma|+c-1} = 2^{c-1} \lambda(\mathcal{U}_{2^c}) \leq 2^{c-1} 2^{-2^c} = 2^{-c-1}$$

In particular the union of all  $L_c$  is a requests set whose weight is bounded by  $\sum_{c \in \mathbb{N}} 2^{-c-1} = 1$ . According to the KC Theorem (Theorem 16-3.3) we therefore have a prefix-free machine  $M$  such that  $K_M(\sigma) \leq |\sigma| - c + 1$  for any string  $\sigma \in W_{2^c}$ . It follows that for every  $Z \in \bigcap_n \mathcal{U}_n$  we have for every  $c$  an integer  $n$  such that  $K_M(Z \upharpoonright_n) \leq n - c$ . Therefore no  $Z \in \bigcap_n \mathcal{U}_n$  is random in the sense of Chaitin/Levin. ■

**Corollary 2.2**

*There is a universal Martin-Löf test, i.e., a Martin-Löf test which contains them all.*

PROOF. According to the proof of the previous theorem, the set  $\bigcap_c \mathcal{U}_c$  where  $\mathcal{U}_c = \{X : \exists n \ K(X \upharpoonright_n) < n - c\}$  is a Martin-Löf test containing all the Martin-Löf tests. ■

**Notation**

We will denote by *MLR* the Martin-Löf random sets.

**Corollary 2.3**

*There is a set  $Z$  which is both low and *MLR*. There is a set  $Z$  which is both computably dominated and *MLR*.*

PROOF. As the class of *MLR* is  $\Sigma_2^0$ , that is to say an effective union of  $\Pi_1^0$  classes, each of these  $\Pi_1^0$  classes contains according to the low basis theorem (Theorem 8-4.3) a set of low degree, and according to the computably dominated basis theorem (Theorem 8-4.5), a computably dominated set. ■

### 3. Randomness and Turing degrees

Let us now see a theorem which echoes Proposition 10-3.37 which told us that for a non-computable set  $X$ , the class of sets computing  $X$  is meager. Similarly, Sacks showed that the class of sets which compute a non-computable set  $X \in 2^{\mathbb{N}}$  is of zero measure.

We use for that a remarkable lemma from Lebesgue which follows from Theorem 17-4.4 and which says that any Borel class  $\mathcal{B}$  of positive measure “concentrates all its measure within intervals”: it is not possible to construct  $\mathcal{B}$  of measure, let’s say  $1/4$ , so that for any  $\sigma$ , the class  $\mathcal{B} \cap [\sigma]$  only contains a quarter of  $[\sigma]$  in terms of measure. There will necessarily be an interval  $[\sigma]$  within which  $\mathcal{B}$  occupies “almost all the space”. The following notation is introduced for this.

**Notation**

Let  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  be a Borel class and  $[\sigma]$  a cylinder. We write  $\lambda(\mathcal{B} \mid [\sigma])$  for

the measure of  $\mathcal{B}$  relative to  $[\sigma]$ , that is to say:

$$\lambda(\mathcal{B} \mid [\sigma]) = \frac{\lambda(\mathcal{B} \cap [\sigma])}{\lambda([\sigma])}.$$

For example if  $\lambda(\mathcal{B} \mid [\sigma]) = 1/2$  it means that  $\mathcal{B}$  occupies “half the space” in the cylinder  $[\sigma]$ . Let us now see the Lebesgue density lemma.

**Lemma 3.1 (Lebesgue density lemma).** Let  $\mathcal{B}$  be a Borel class of positive measure. Then, for all  $\varepsilon > 0$  there exists a cylinder  $[\sigma]$  such that  $\lambda(\mathcal{B} \mid [\sigma]) > 1 - \varepsilon$ . ★

PROOF. The idea is simple. According to Theorem 17-4.4 we can approximate  $\mathcal{B}$  by an open class  $\mathcal{U}$  containing it and whose measure is as close to that of  $\mathcal{B}$  as we want. For an open class whose measure is sufficiently close to that of  $\mathcal{B}$ , it is not possible for the measure of  $\mathcal{B}$  inside each cylinder  $[\sigma]$  of the open class to be too small, otherwise the total measure of  $\mathcal{B}$  would be too small compared to that of  $\mathcal{U}$ . We now give the formal proof.

Let us fix  $\varepsilon > 0$ . According to Theorem 17-4.4 there exists an open class  $\mathcal{U} \supseteq \mathcal{B}$  such that  $\lambda(\mathcal{U} \setminus \mathcal{B}) < \varepsilon\lambda(\mathcal{B})$ . Let  $W \subseteq 2^{<\mathbb{N}}$  be a prefix-free set such that  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$ . Note that we have  $\lambda(\mathcal{B}) = \sum_{\sigma \in W} \lambda(\mathcal{B} \cap [\sigma])$ . Suppose by contradiction that for all  $\sigma \in W$  we have  $\lambda(\mathcal{B} \mid [\sigma]) \leq 1 - \varepsilon$ . Then,  $\sum_{\sigma \in W} \lambda(\mathcal{B} \cap [\sigma]) \leq \sum_{\sigma \in W} (1 - \varepsilon)\lambda([\sigma]) = (1 - \varepsilon) \sum_{\sigma \in W} \lambda([\sigma]) = (1 - \varepsilon)\lambda(\mathcal{U})$ . We therefore have  $\lambda(\mathcal{B}) \leq (1 - \varepsilon)\lambda(\mathcal{U})$ . But by hypothesis  $\lambda(\mathcal{U}) - \lambda(\mathcal{B}) = \lambda(\mathcal{U} \setminus \mathcal{B}) < \varepsilon\lambda(\mathcal{B}) \leq \varepsilon\lambda(\mathcal{U})$ . This gives  $\lambda(\mathcal{B}) > \lambda(\mathcal{U}) - \varepsilon\lambda(\mathcal{U}) = (1 - \varepsilon)\lambda(\mathcal{U})$ , a contradiction. ■

Note that Lebesgue showed something even stronger, which we will see with Theorem 19-4.6: the intervals  $\sigma$  of the previous lemma are in fact very numerous: for almost all  $X \in \mathcal{B}$ , the quantity  $\lambda(\mathcal{B} \mid [X \upharpoonright_n])$  goes towards 1 when  $n$  goes towards  $+\infty$ . Let us now see Sacks’ theorem.

**Theorem 3.2 (Sacks [187])**

Let  $Y \in 2^{\mathbb{N}}$  be non-computable. Then,  $\lambda(\{X \in 2^{\mathbb{N}} : X \geq_T Y\}) = 0$ .

PROOF. Let  $\Phi$  be a Turing functional. Suppose by contradiction that  $\lambda(\{X \in 2^{\mathbb{N}} : \Phi(X) = Y\}) > 0$ . According to Lebesgue density Lemma 3.1 there exists a string  $\sigma$  such that  $\lambda(\{X \in 2^{\mathbb{N}} : \Phi(X) = Y\} \mid [\sigma]) > 1/2$ .

We now describe an algorithm to compute  $Y$ , the principle of which comes down to a “majority vote”: for all  $n$  and for all  $i \in \{0, 1\}$  we enumerate the open class  $\mathcal{U}_{n,i}$  described by the strings  $\tau \succeq \sigma$  such that  $\Phi(\tau, n) \downarrow = i$ . According to our hypothesis on  $\sigma$ , for all  $n$  we must have  $\lambda(\mathcal{U}_{n,i} \mid [\sigma]) > 1/2$

for  $i = Y(n)$ . Note that for obvious “lack of space” reasons we cannot have  $\lambda(\mathcal{U}_{n,i} \mid [\sigma]) > 1/2$  for  $i \neq Y(n)$ . It is therefore enough to wait until we have  $\lambda(\mathcal{U}_{n,i} \mid [\sigma]) > 1/2$  for a given  $i$ . At this point we are sure that  $Y(n) = i$ .

The algorithm contradicts the non-computability of  $Y$ . It follows that  $\lambda(\{X \in 2^{\mathbb{N}} : \Phi(X) = Y\}) = 0$ . As this is the case for all functionals, and as a countable union of measure 0 classes remains a measure 0 class, we therefore have  $\lambda(\{Y \in 2^{\mathbb{N}} : Y \geq_T X\}) = 0$ . ■

The previous theorem implies that a sufficiently random set  $Z$  cannot compute a non-computable set  $Y$ . This shows in particular that probabilistic algorithms are powerless to provide more computational power: assuming that we have a means of producing random bits (via a physical process for example), the probability that the produced sequence allows for example to compute the halting problem is null.

We now see a dual theorem: given an arbitrary set, we can always find a Martin-Löf random computing it. The general idea is the following: given a computable tree  $T$  where each node has at least two incomparable extensions, it is trivial to compute a bijection between  $2^{\mathbb{N}}$  and  $[T]$ : if  $f(\sigma)$  is defined then  $f(\sigma 0)$  and  $f(\sigma 1)$  are each sent to the first incomparable extensions of  $f(\sigma)$  into  $T$ . As the bijection is computable, we deduce that  $[T]$  contains for any set an element Turing equivalent to it.

If now the tree  $T$  also contains leaves — as it is the case for trees which represent  $\Pi_1^0$  classes — the algorithm does not work at all. The problem is to know which node is “really” branching: it may be that a node is branching, but that one of its extensions ends up only in leaves. It is in fact possible to construct uncountable  $\Pi_1^0$  classes no member of which computes the halting set, or even no member of which computes a Martin-Löf random set [37, Lemma 5.1]. On the other hand, when the  $\Pi_1^0$  class in question has positive measure, the situation changes: the positive measure gives us some guarantee that “many” nodes are branching, allowing to develop a — non-computable — encoding of any element in such a way that a decoding algorithm is possible. We need for that a lemma establishing how fast one can find branching nodes in a tree of positive measure.

**Lemma 3.3 (Kučera [127]).** Let  $\mathcal{B}$  be a Borel class. Suppose  $\lambda(\mathcal{B} \mid [\sigma]) \geq 2^{-n}$ . Then, there are two distinct extensions  $\tau_0, \tau_1 \succeq \sigma$  of length  $|\sigma| + n + 1$  such that  $\lambda(\mathcal{B} \mid [\tau_i]) \geq 2^{-n-1}$ . ★

PROOF. This is a simple counting argument: suppose that for all strings  $\tau$  of length  $|\sigma| + n + 1$  extending  $\sigma$ , except one, we have  $\lambda(\mathcal{B} \mid [\tau]) < 2^{-n-1}$ .

Then, by considering  $\tau$  such that  $|\tau| = |\sigma| + n + 1$  we have:

$$\begin{aligned}
 \lambda(\mathcal{B} \cap [\sigma]) &\leq 2^{-|\sigma|-n-1} + (2^{n+1} - 1)2^{-|\tau|-n-1} \\
 &\leq 2^{-|\sigma|-n-1} + (2^{n+1} - 1)2^{-|\sigma|-2n-2} \\
 &\leq 2^{-|\sigma|-n-1} + 2^{n+1}2^{-|\sigma|-2n-2} - 2^{-|\sigma|-2n-2} \\
 &\leq 2^{-|\sigma|-n-1} + 2^{-|\sigma|-n-1} - 2^{-|\sigma|-2n-2} \\
 &< 2^{-|\sigma|-n}
 \end{aligned}$$

which contradicts  $\lambda(\mathcal{B} \mid [\sigma]) \geq 2^{-n}$ . ■

According to the previous lemma, if  $\mathcal{B}$  is a  $\Pi_1^0$  class of positive measure represented by a computable tree  $T$ , then  $\lambda([T] \mid [\sigma]) \geq 2^{-n}$  implies that the next branching extension of  $\sigma$  will arrive before  $n + 1$  bits.

**Theorem 3.4 (Kučera [127], Gács [69])**

*Let  $X \in 2^{\mathbb{N}}$ . Then, there exists an MLR set  $Z \in 2^{\mathbb{N}}$  such that  $Z \geq_T X$ .*

PROOF. We prove the following theorem: let  $\mathcal{P}$  be a  $\Pi_1^0$  class of positive measure. Then, for all  $X$  there exists  $Z \in \mathcal{P}$  such that  $Z \geq_T X$ . The reader can consult Figure 3.5 for an illustration of the proof. Let us fix  $X$ . Let's build our element  $Z \in \mathcal{P}$  such that  $Z \geq_T X$ . First let's fix  $c$  such that  $\lambda(\mathcal{P}) > 2^{-c}$ . We define, according to Lemma 3.3 the constants  $m_0 = 0$  and  $m_{n+1} = m_n + c + n + 1$ .

We define  $\sigma_0 = \epsilon$ . Suppose that  $\sigma_n$  of length  $m_n$  is defined such that  $\lambda(\mathcal{P} \mid [\sigma_n]) \geq 2^{-c-n}$ . Let's define an extension  $\sigma_{n+1}$  with the same property. According to Lemma 3.3 there are two distinct extensions  $\tau_0, \tau_1$  of length  $m_n + c + n + 1$  such that  $\lambda(\mathcal{P} \mid [\tau_i]) \geq 2^{-c-(n+1)}$ . If  $X(n) = 0$  then we define  $\sigma_{n+1}$  as the leftmost extension satisfying the inequality. If  $X(n) = 1$  then we define  $\sigma_{n+1}$  as the rightmost extension satisfying the inequality.  $Z$  is defined as the limit point of  $\sigma_0 \prec \sigma_1 \prec \dots \prec \sigma_n \prec \sigma_{n+1} \prec \dots$ .

It is clear that  $Z \in \mathcal{P}$ . We must now detail how to use  $Z$  to compute  $X$ . To find the bit  $X(n)$ , let  $\sigma = Z \upharpoonright_{m_n}$ . We co-enumerate the set  $A$  of strings  $\tau \succeq \sigma$  of length  $m_{n+1}$  such that  $\lambda(\mathcal{P} \mid [\tau]) \geq 2^{-c-(n+1)}$ . If  $Z \upharpoonright_{m_{n+1}}$  ever appears at some computation step  $t$  to be the leftmost string of  $A[t]$  then  $X(n) = 0$ . If  $Z \upharpoonright_{m_{n+1}}$  ever appears at some computation step  $t$  to be the rightmost string of  $A[t]$  then  $X(n) = 1$ . By construction one and only one of these two events must happen. ■

## 4. Randomness and DNC degree

Kučera/Gács's Theorem 3.4 tells us there exists a Martin-Löf random *above* of each Turing degree, but not necessarily *in* every Turing degree. We

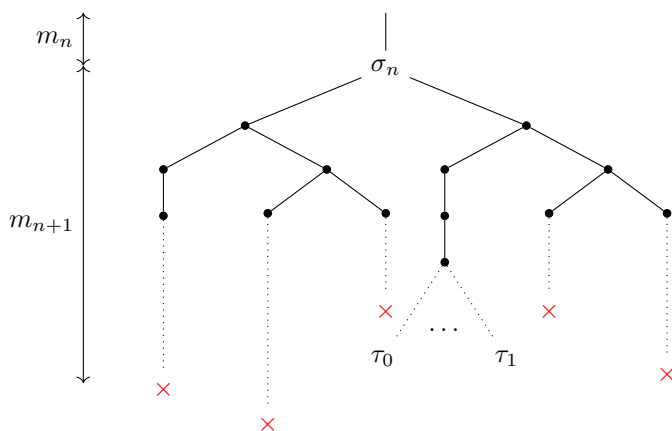


Figure 3.5: Illustration of the proof of Kučera/Gács's Theorem. We have defined our string  $\sigma_n$  of length  $m_n$ . We are guaranteed the existence of two distinct strings  $\tau_0, \tau_1$  of length  $m_{n+1}$  such that the construction can continue from any of them. It is enough to pick  $\tau_0$  as the leftmost of these strings and  $\tau_1$  as the rightmost. We pick  $\sigma_{n+1} = \tau_0$  to encode a 0 and  $\sigma_{n+1} = \tau_1$  to encode a 1. For the decoding, knowing  $\sigma_n$  and  $\sigma_{n+1}$ , one can wait for sufficiently many nodes of the tree extending  $\sigma_n$  to end as leaves — more specifically for their measure condition to go below a certain value — until  $\sigma_{n+1}$  becomes the leftmost or the rightmost node of what remains. We then know which bit has been encoded.

will in fact see that if for example any 1-generic set can be computed by a Martin-Löf random set, no Martin-Löf random set cannot be computed by a 1-generic set. The idea is simply that any Martin-Löf random is of DNC degree, and as we have seen it with Theorem 10-3.21 no 1-generic set bounds a DNC degree. Precisely we are going to show the following theorem, which gives two other characterizations of DNC degrees. The equivalence (1)  $\leftrightarrow$  (2) has been shown by Kjos-Hanssen, Merkle and Stephan [111], and the equivalence (1)  $\leftrightarrow$  (3) by Greenberg and Miller [75].

**Theorem 4.1**

Let  $X \in 2^{\mathbb{N}}$ . The following statements are equivalent:

- (1)  $X$  is of DNC degree
- (2)  $X$  computes a function  $f : \mathbb{N} \rightarrow 2^{<\mathbb{N}}$  such that  $K(f(n)) \geq n$
- (3)  $X$  computes an infinite subset of an MLR set.

In order to prove (1)  $\rightarrow$  (3), we will use another lemma from Kučera.

**Lemma 4.2 (Kučera [127]).** Let  $\mathcal{P}$  be a  $\Pi_1^0$  class of positive measure. There exists a  $\Pi_1^0$  subclass  $\mathcal{Q} \subseteq \mathcal{P}$  of positive measure and an integer  $c$  such that for the  $\Pi_1^0$  class  $\mathcal{P}_e$  of code  $e$  we have  $\mathcal{Q} \cap \mathcal{P}_e \neq \emptyset \rightarrow \lambda(\mathcal{Q} \cap \mathcal{P}_e) > 2^{-e-c}$ . ★

PROOF. Let  $c$  be such that  $\lambda(\mathcal{P}) > 2^{-c+1}$ . We define  $\mathcal{Q}$  as being the class which co-enumerates  $\mathcal{P}$  and at the same time, at each step  $s$ , for all  $e \leq s$ , if the measure of  $\mathcal{Q}[s] \cap \mathcal{P}_e[s]$  goes below  $2^{-e-c}$ , remove  $\mathcal{P}_e[s]$  from  $\mathcal{Q}[s]$ .

For each  $e$  we remove one piece of measure of at most  $2^{-e-c}$ . The total measure removed is therefore bounded by  $\sum_{e \in \mathbb{N}} 2^{-e-c} = 2^{-c+1}$ . As  $\lambda(\mathcal{P}) > 2^{-c+1}$  we have  $\lambda(\mathcal{Q}) > 0$ . ■

We can now show the announced theorem.

PROOF OF THEOREM 4.1. Let us show (1)  $\rightarrow$  (2). Suppose  $X$  of DNC degree. For any  $n$  we can compute the code  $a_n$  of the set  $W_{a_n}$  which enumerates all the strings  $\sigma$  such that  $K(\sigma) < n$ . Note that  $|W_{a_n}| < \sum_{m < n} 2^m = 2^n$ . According to Theorem 7-2.6,  $X$  therefore computes uniformly in  $n$  a string  $\sigma \notin W_{a_n}$  and therefore such that  $K(\sigma) \geq n$ .

Let us now show (2)  $\rightarrow$  (1). Suppose that  $X$  is not of DNC degree. Then, for any function  $f \leq_T X$  we have  $f(n) = \Phi_n(n) \downarrow$  for infinitely many  $n$ . However  $\Phi_n(n) \downarrow$  implies  $K(\Phi_n(n)) <^+ K(n)$  (see Exercize 16-2.3 if this step is not clear) and we have  $K(n) \leq^+ \log_2(n) + 2\log_2(\log_2(n))$  according to Proposition 16-2.5 which gives  $K(\Phi_n(n)) \leq^+ \log_2(n) + 2\log_2(\log_2(n))$ . Also  $\log_2(n) + 2\log_2(\log_2(n)) < n$  for  $n$  sufficiently large.

Let us now show (1)  $\rightarrow$  (3). Let  $X$  be of DNC degree. Let  $\mathcal{P}$  be a  $\Pi_1^0$  class of positive measure containing only Martin-Löf randoms. From Lemma 4.2 we can suppose that there exists  $c \in \mathbb{N}$  such that we have  $\mathcal{P} \cap \mathcal{P}_e \neq \emptyset$  implies  $\lambda(\mathcal{P} \cap \mathcal{P}_e) > 2^{-c-e}$  for the  $\Pi_1^0$  class  $\mathcal{P}_e$  of code  $e$ . For any set  $S \subseteq \mathbb{N}$  we define  $\mathcal{Q}_S = \{Y : S \subseteq Y\}$ . Note that  $\mathcal{Q}_S$  is  $\Pi_1^0$  for any finite set  $S \subseteq \mathbb{N}$ . Suppose we have defined  $S_n \subseteq \mathbb{N}$  of size  $n$  such that  $\mathcal{Q}_{S_n}$  is of code  $e_n$  and such that  $\mathcal{P} \cap \mathcal{Q}_{S_n} \neq \emptyset$ . Let  $W_n$  be the c.e. set of integers  $a$  such that  $\mathcal{P} \cap \mathcal{Q}_{S_n \cup \{a\}} = \emptyset$ . By hypothesis we have  $\lambda(\mathcal{P} \cap \mathcal{Q}_{S_n}) > 2^{-c-e_n}$ . If an integer  $a$  is in  $W_n$  this means that for any element  $Z$  of  $\mathcal{P} \cap \mathcal{Q}_{S_n}$  we have  $Z(a) = 0$ . For a fixed sequence of integers  $a_1, a_2, \dots, a_m$ , the set of  $Z$  such that  $\forall i < m \ Z(a_i) = 0$  is of measure  $2^{-m}$  since half of the elements  $Z$  are such that  $Z(a_1) = 0$ , then half of this half are such that  $Z(a_2) = 0$ , etc. As  $\lambda(\mathcal{P} \cap \mathcal{Q}_{S_n}) > 2^{-c-e_n}$  we therefore necessarily have at most  $c + e_n$  elements in  $W_n$ . We can therefore apply Theorem 7-2.6 to compute uniformly in  $X$  an integer  $a \notin W_n$ , and therefore such that  $\mathcal{P} \cap \mathcal{Q}_{S_n \cup \{a\}} \neq \emptyset$ . We define  $S_{n+1} = S_n \cup \{a\}$  and we compute a

code  $e_{n+1}$  for  $\mathcal{Q}_{S_{n+1}}$ . We compute this way using  $X$  the infinite set  $S = \bigcup_n S_n$  for which  $\mathcal{P} \cap \mathcal{Q}_S = \bigcap_n \mathcal{P} \cap \mathcal{Q}_{S_n}$ . As  $\lambda(\mathcal{P} \cap \mathcal{Q}_{S_n}) > 0$  for all  $n$  then  $\mathcal{P} \cap \mathcal{Q}_{S_n}$  is non-empty for all  $n$ . As a decreasing intersection of non-empty closed classes, the class  $\mathcal{P} \cap \mathcal{Q}_S$  is non-empty. In particular there is an element  $Z \in \mathcal{P}$  such that  $S \subseteq Z$ .

Let us now show (3)  $\rightarrow$  (1). Let  $X \subseteq Z$  be an infinite subset of some MLR. For any  $n$  let  $f(n)$  be the  $n$ -th element of  $X$ . Let us show  $K(X \upharpoonright_{f(n)}) >^+ n$ . Suppose the opposite, that is  $\forall c \exists n K(X \upharpoonright_{f(n)}) < n - c$ . Then, we can also compress  $Z \upharpoonright_{f(n)}$  by  $c$  bits, using the machine  $M$  which on  $\tau\sigma$  such that  $U(\tau) \downarrow = \rho$  returns the string  $\sigma$  in which we insert the 1's of  $\rho$  at the positions at which they are in  $\rho$  (if  $\sigma$  is sufficiently large). The string  $\sigma$  here represents the prefix of  $Z \upharpoonright_{f(n)}$  to which we have “removed” the  $n$  bits at 1 of  $X \upharpoonright_{f(n)}$ . It is therefore of length  $f(n) - n$ . The string  $\tau$  is for its part a compression of  $X \upharpoonright_{f(n)}$  of length less than  $n - c$ . The string  $\tau\sigma$  is therefore a compression of length less than  $f(n) - c$  of a prefix of  $Z$  of length  $f(n)$ , which results in a contradiction for  $c$  large enough. So we have  $K(X \upharpoonright_{f(n)}) >^+ n$  and according to (2)  $\rightarrow$  (1) we have (3)  $\rightarrow$  (1). ■

### Corollary 4.3

*The class of sets of DNC degree is of measure 1.*

PROOF. We can easily deduce from the previous theorem that all MLR sets are of DNC degree. ■

We have seen with Exercize 8-7.6 that for any set  $X$ , any function  $f : \mathbb{N} \rightarrow \{0, 1\}$  which is DNC relative to  $X$ , computes  $X$ ; in other words, any PA degree relative to  $X$  computes  $X$ . The situation is different when considering DNC functions with arbitrary values:

### Corollary 4.4

*For any set  $X$ , there exists a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  DNC relative to  $X$  which does not compute  $X$ .*

PROOF. By relativized Corollary 4.3, the measure of the class of sets computing a DNC function relative to  $X$  is 1, while by Theorem 3.2, the measure of the class of sets computing  $X$  is 0. It follows that there exists a set computing a DNC function relative to  $X$ , but not computing  $X$ . ■



# Chapter 19

## Other randomness notions

Is Martin-Löf’s randomness definition *the* correct one? The concordance between the “incompressibility” and “typicality” viewpoints goes in that direction. There is, however, little hope to be one day certain that this mathematical definition — or another — will ever perfectly embrace the contours of the epistemological aspects to which it relates.

We agree here with Christopher Porter’s viewpoint, who defends in a multidisciplinary mathematics/philosophy thesis [178], the “no thesis thesis” according to which there is no absolute definition of what a random number is. As an example, we can legitimately consider that the ability to compute  $\emptyset'$  is an atypical property. This is however the case of the MLR  $\Omega$ , which is therefore not that much random.

We therefore study here other randomness notions of various strength, all worthy of interest, in particular through the links that we can forge between them and Computability Theory. A very large number of randomness classes have been studied in the literature, some weaker and others stronger or even incomparable with Martin-Löf randomness. Our objective is not to list them exhaustively, but rather to give a photograph of some of the most emblematic ones.

### 1. Weak-2 randomness

Let’s start by examining what happens if we remove the MLR tests condition of being effectively of measure 0.

**Definition 1.1.** A set  $Z$  is *weakly 2-random* if it does not belong to any  $\Pi_2^0$  class of measures 0. ◇

The name “weak 2-randomness” relates to the forthcoming notion of “2-randomness” (see Section 3). We actually think the name “strong Martin-Löf randomness” would have better fit the concept, but as it is referred to as weak 2-randomness everywhere in the literature, we kept it that way.

The notion of weak 2-randomness is natural. After all, why bother with the condition of rapid convergence of our  $\Pi_2^0$ ’s measure to 0? We obtain a stronger notion, but perhaps less elegant, as evidenced by the following proposition and corollaries:

**Proposition 1.2.** No  $\Delta_2^0$  set is weakly 2-random. ★

PROOF. Let  $A$  be a  $\Delta_2^0$  set. Let  $(A_s)_{s \in \mathbb{N}}$  be a sequence of computable sets such that  $\lim_s A_s = A$ . We define  $\mathcal{U}_{\langle n, t \rangle} = \bigcup_{s > t} [A_s \upharpoonright_n]$ . It is clear that  $\bigcap_{\langle n, t \rangle \in \mathbb{N}} \mathcal{U}_{\langle n, t \rangle} = \{A\}$ . In particular  $\bigcap_{\langle n, t \rangle \in \mathbb{N}} \mathcal{U}_{\langle n, t \rangle}$  is a  $\Pi_2^0$  class of measure 0 which contains  $A$ . ■

### Corollary 1.3

*The class of weakly 2-randoms is strictly included in that of the MLRs.*

PROOF. As a left-c.e. set,  $\Omega$  is  $\Delta_2^0$  and therefore not weakly 2-random. ■

### Corollary 1.4

*There is no universal test for weak 2-randomness.*

PROOF. Let  $\bigcap_n \mathcal{U}_n$  be a  $\Pi_2^0$  class of measure 0. For some  $n$  the set  $2^{\mathbb{N}} \setminus \mathcal{U}_n$  is therefore a  $\Pi_1^0$  class of positive measure. Let  $T$  be the computable tree representing  $2^{\mathbb{N}} \setminus \mathcal{U}_n$ . The leftmost path of  $T$  is a left-c.e. set which is therefore  $\Delta_2^0$  and therefore not weakly 2-random.

We conclude there is no  $\Pi_2^0$  class of measure 0 which contains all the non weak 2-random sets. ■

It is possible to considerably improve Corollary 1.4. Yu Liang has indeed shown [235] that the class of elements which are not weakly 2-random is not  $\Pi_2^0$  (even relative to any oracle there is no test capturing exactly those which are not weakly 2-random) and even is not  $\Pi_3^0$ . The Borel complexity of the class of non weakly 2-randoms sets is therefore strictly  $\Sigma_3^0$ .

By removing the condition of rapid convergence towards 0 in our  $\Pi_2^0$ ’s measure, we thus remove certain MLR sets : at least those which are  $\Delta_2^0$ .

It seems likely that these are not the only ones to be excluded. There is in fact give it an elegant characterization discovered by Downey, Nies, Weber and Yu.

**Theorem 1.5 (Downey, Nies, Weber and Yu [48])**

*Let  $Z$  be an MLR set. The following are equivalent:*

- (1)  $Z$  is not weakly 2-random.
- (2)  $Z$  computes a non-computable  $\Delta_2^0$  set.
- (3)  $Z$  computes a non-computable c.e. set.

PROOF. Let us first show (2)  $\rightarrow$  (1). Let  $A$  be a non-computable  $\Delta_2^0$  set with  $A = \lim_s A_s$ . Let  $\Phi$  be such that  $\Phi(Z) = A$ . We define the  $\Sigma_1^0$  class  $\mathcal{U}_{\langle n, t \rangle}$  equal to  $\bigcup_{s > t} \{X : \Phi(X) \succeq A_s \upharpoonright_n\}$ . It is clear that  $\bigcap_{\langle n, t \rangle} \mathcal{U}_{\langle n, t \rangle}$  contains exactly the class of elements  $X$  such that  $\Phi(X) = A$ . By Sacks's Theorem 18-3.2 this class is of zero measure. So  $Z$  is not weakly 2-random.

Let us now show (1)  $\rightarrow$  (3). Suppose  $Z$  MLR such that  $Z \in \bigcap_n \mathcal{U}_n$  where  $\bigcap_n \mathcal{U}_n$  is a  $\Pi_2^0$  class of measure zero. We can assume  $\mathcal{U}_{n+1} \subseteq \mathcal{U}_n$ . We are going to construct a c.e. simple set  $A$  (see Definition 16-4.2) such that the “entry time” of  $Z$  into  $\mathcal{U}_n$  bounds the time necessary for the enumeration of  $n$  into  $A$ . While constructing  $A$  we also construct for all  $e$  some classes  $\mathcal{V}_e$  which are  $\Sigma_1^0$  and such that  $\lambda(\mathcal{V}_e) < 2^{-e}$ . The classes  $\mathcal{V}_e$  will be used to create a Solovay test that will help us conclude. We fix an enumeration  $(W_e)_{e \in \mathbb{N}}$  of the c.e. sets. Initially each  $\mathcal{V}_e$  is the empty set. At the stage of computation  $t$ , for all  $e \leq t$ , if  $W_e$  does not intersect  $A$  yet then we look for  $n \in W_e[t]$  with  $n > 2e$  and such that  $\lambda(\mathcal{U}_n[t]) < 2^{-e}$ . If such an integer  $n$  is found, it is enumerated into  $A$  at step  $t$  and we set  $\mathcal{V}_e = \mathcal{U}_n[t]$ . This concludes the construction.

Note that if  $W_e$  is infinite, we will inevitably end up in finding  $n \in W_e[t]$  with  $n > 2e$  such that  $\lambda(\mathcal{U}_n[t]) < 2^{-e}$  because for  $n$  sufficiently large, we have anyway  $\lambda(\mathcal{U}_n) < 2^{-e}$ . So by the usual argument (see Proposition 16-4.4)  $A$  is indeed a simple set and therefore not computable. It remains to show that  $Z$  allows to compute  $A$ . We use for that our sets  $\mathcal{V}_e$  which form a Solovay test, because  $\lambda(\mathcal{V}_e) \leq 2^{-e}$  and therefore  $\sum_e \lambda(\mathcal{V}_e)$  is finite. In particular if  $Z$  is MLR it can only belong to a finite number of  $\mathcal{V}_e$ . There is therefore  $m$  such that for all  $n \geq m$  if the integer  $n$  is enumerated into  $A$  at step  $t$  then  $Z \notin \mathcal{U}_n[t]$ . Then, to know if  $n \geq m$  belongs to  $A$ , it suffices to search for the smallest time  $t$  such that  $Z \in \mathcal{U}_n[t]$ , and to enumerate  $A$  up to step  $t$ . We then have  $n \in A$  iff  $n \in A[t]$ .

Finally (3)  $\rightarrow$  (2) is trivial. ■

Among the MLR sets which are captured by  $\Pi_2^0$  classes of zero measure, there are of course all the MLR computing  $\emptyset'$ , but they are not the only ones (it is in particular a consequence of Corollary 20-3.10 to come). On the other hand, we will see with Corollary 20-3.2 that if an MLR is incomplete — does not compute  $\emptyset'$  — and computes a non-computable c.e. set, then this set is necessarily K-trivial. It is in fact a characterization of the c.e. K-trivials as we will see with Corollary 20-3.10.

Let us now see an interesting corollary to Theorem 1.5. We have seen with Corollary 18-2.3 that there exist MLR and computably dominated sets. We will see with Theorem 3.4 that being computably dominated is however an atypical property: the class of computably dominated sets is of measure zero. The following corollary indicates that the notion of being weakly-2 random is not sufficient to account for this.

**Corollary 1.6**

*Let  $Z$  be an MLR but not weakly 2-random set. Then,  $Z$  is not computably dominated.*

PROOF. If  $Z$  is an MLR set but not weakly 2-random, then it computes a non-computable c.e. set which is therefore by Proposition 7-4.7 not computably dominated. ■

We deduce from the previous corollary that all MLRs which are computably dominated are also weak 2-randoms. We will therefore need a stronger notion to capture these elements. Weak 2-randomness is on the other hand sufficient to show that it is atypical to be of PA degree, via a Frank Stephan's proof, of which we hope the reader will appreciate the finesse. Stephan himself sums up his result in a very Eastwoodian way: “*There are two types of Martin-Löf randoms sets: those which are computationally powerful enough to solve the halting problem, and those which are computationally weak in the sense that they are not PA-complete.*”

**Theorem 1.7 (Stephan [217])**

*Let  $Z$  be Martin-Löf random. Then,  $Z$  is of PA degree iff  $Z \geq_T \emptyset'$ .*

PROOF. Suppose that  $\Phi(Z)$  is a DNC<sub>2</sub> set, in order to prove  $Z \geq_T \emptyset'$ . During the construction we will effectively define a sequence of partial computable function codes  $a_0 < a_1 < a_2 < \dots$ . Also from the fixed point theorem we can assume that each  $a_k$  codes for a function having access to this sequence, including the code  $a_k$  itself. We partition the sequence  $(a_k)_{k \in \mathbb{N}}$  into a sequence of consecutive intervals  $I_n$  in such a way that  $I_n$  contains  $2^n$  elements of the sequence. We will henceforth name  $a_n^k$

the  $k$ -th element of  $I_n$ . Initially each  $a_n^k$  is the code of a nowhere defined function, which waits for some events to occur, before perhaps halting on some inputs.

For all  $n \in \mathbb{N}$  we define the  $\Sigma_1^0$  class  $\mathcal{U}_{n,i}^0 = \{X : \Phi(X, a_n^0) \downarrow = i\}$ . We then look for  $i \in \{0, 1\}$  and for a computation time  $t$  such that  $\lambda(\mathcal{U}_{n,i}^0[t]) > 2^{-n}$ . If for a given  $n$  we find such an element  $i \in \{0, 1\}$  and such a computation time  $t$  then we define  $\mathcal{U}_n^0 = \mathcal{U}_{n,i}^0[t]$ , we decide that  $a_n^0$  is the code of the function which on  $a_n^0$  returns  $i$ , and we define  $\mathcal{U}_{n,i}^1 = \{X \notin \mathcal{U}_n^0 : \Phi(X, a_n^1) \downarrow = i\}$ . We then continue inductively: if  $\mathcal{U}_{n,i}^{k-1}$  is defined as well as  $\mathcal{U}_{n,i}^k$  for  $i \in \{0, 1\}$ , then we look for  $i \in \{0, 1\}$  and a computation time  $t$  such that  $\lambda(\mathcal{U}_{n,i}^k[t]) > 2^{-n}$ . If we find such an element  $i \in \{0, 1\}$  and such a computation time  $t$  we define  $\mathcal{U}_n^k = \mathcal{U}_{n,i}^{k-1} \cup \mathcal{U}_{n,i}^k[t]$ , we decide that  $a_n^k$  is the code of the function which on  $a_n^k$  returns  $i$ , and we define  $\mathcal{U}_{n,i}^{k+1} = \{X \notin \mathcal{U}_n^k : \Phi(X, a_n^{k+1}) \downarrow = i\}$ .

In the following, we call *versions* of  $\mathcal{U}_{n,i}$  the different  $\mathcal{U}_{n,i}^0, \mathcal{U}_{n,i}^1, \mathcal{U}_{n,i}^2, \dots$ . The classes  $\mathcal{U}_n^0, \mathcal{U}_n^1, \mathcal{U}_n^2, \dots$  will be *truncated versions*. During this process, let us notice two things: first for all  $n$  we will arrive at final and not truncated versions  $\mathcal{U}_{n,0}^k$  and  $\mathcal{U}_{n,1}^k$  of  $\mathcal{U}_{n,0}$  and  $\mathcal{U}_{n,1}$ , such that  $\lambda(\mathcal{U}_{n,0}^k \cup \mathcal{U}_{n,1}^k) < 2^{-n+1}$ . Indeed by construction each new version is disjoint from its previous truncated version, which has a measure strictly greater than  $2^{-n}$ . Since there are  $2^n$  possible versions, each of the last two must necessarily have a measure less than  $2^{-n}$ .

Then, no set belonging to a truncated version  $\mathcal{U}_n^k$  can compute a  $\text{DNC}_2$  set via  $\Phi$ : we make sure of this by defining the code  $a_n^k$  as being such that  $\Phi_{a_n^k}(a_n^k) \downarrow = i$  for  $i$  such that any element of  $\mathcal{U}_n^k$  also returns  $i$  on  $a_n^k$ . Since  $\Phi(Z)$  is a  $\text{DNC}_2$  set, then  $Z$  is necessarily in every last non-truncated version. On the other hand, if for the last versions  $\mathcal{U}_{n,0}^k$  and  $\mathcal{U}_{n,1}^k$  we have indeed that  $\mathcal{U}_{n,0}^k \cup \mathcal{U}_{n,1}^k$  is a  $\Sigma_1^0$  class of measure less than  $2^{-n+1}$ , these classes are not obtained uniformly in  $n$ , because one potentially never knows if one has arrived or not at the last version. This is where  $\emptyset'$  comes in.

We will now define a Solovay test via the following sets  $\mathcal{V}_n$ : at the start each  $\mathcal{V}_n$  is the empty set. Then if  $n$  is enumerated in  $\emptyset'$  at time  $t$ , we declare that  $\mathcal{V}_n$  is the union  $\mathcal{U}_{n,0}^k \cup \mathcal{U}_{n,1}^k$  of the versions available at time  $t$ , possibly truncated so that the measure does not exceed  $2^{-n+1}$ . It is then clear that the sequence  $(\mathcal{V}_n)_{n \in \mathbb{N}}$  forms a Solovay test. Suppose now the entry time of  $Z$  into  $\mathcal{U}_n = \{X : \bigwedge_{a \in I_n} \Phi(X, a) \downarrow\}$  — the smallest  $t$  such that  $Z \in \mathcal{U}_n[t]$  — is infinitely often less than the enumeration time of  $n$  into  $\emptyset'$ . Let us remember that  $Z$  is necessarily in each last version and that these last versions are not truncated. We deduce that for each of these integers  $n$  the

set  $Z$  belongs to  $\mathcal{V}_n$ , which implies that  $Z$  is captured by the Solovay test, contradicting the fact that  $Z$  is random in the sense of Martin-Löf. We deduce that the entry time of  $Z$  into  $\mathcal{U}_n$  is for almost all  $n$  greater than the enumeration time of  $n$  into  $\emptyset'$ . So  $Z$  can tell if a sufficiently large integer  $n$  belongs to  $\emptyset'$  by simply checking whether  $n \in \emptyset'[t]$  where  $t$  is the smallest such that  $Z \in \mathcal{U}_n[t]$ . ■

### Corollary 1.8

*If  $Z$  is weakly-2 random then  $Z$  is not of PA degree. In particular, the class of sets of PA degree is of measure zero.*

## 2. Relativization of randomness

As for most notions in Computability Theory, it is possible to relativize Martin-Löf randomness to an oracle.

**Definition 2.1.** Let  $X \in 2^{\mathbb{N}}$ . A *X-Martin-Löf test* is given by a  $\Pi_2^0(X)$  class  $\bigcap_n \mathcal{U}_n$  such that  $\lambda(\mathcal{U}_n) \leq 2^{-n}$ . A set  $Z$  is *X-Martin-Löf random* or *MLR(X)* if it passes all X-Martin-Löf tests. ♦

According to Definition 17-3.3 and what precedes Example 17-3.8, the code of a  $\Pi_2^0(X)$  class is an integer  $\langle 2, 1, e \rangle$  for which  $W_e^X$  enumerates integers  $\langle 1, 0, a_n \rangle$  such that  $[W_{a_n}^X]$  is the  $n$ -th component of our  $\Pi_2^0(X)$ . Our code  $e$  can however be used with any other oracle  $Y$ : the set  $W_e^Y$  also enumerates integers. It is easy to see how to uniformly transform the code  $e$  in order to keep in the enumerations  $W_e^Y$  (for any oracle  $Y$ ) only the integers of the form  $\langle 1, 0, a \rangle$ . The goal being that for any oracle  $Y$ , the same integer  $e$  codes for a  $\Pi_2^0(Y)$  class.

### Oracle class and machine

We will speak of *oracle class* to insist on the fact that the same code  $e$  uniformly defines a  $\Sigma_n^0(X)$  class for any oracle  $X \in 2^{\mathbb{N}}$ . We will speak in the same way of *oracle machine* to illustrate the fact that a machine  $M : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  using an oracle  $X$ , is also a machine —that is to say a partial function from  $2^{<\mathbb{N}}$  to  $2^{<\mathbb{N}}$ — with any oracle  $Y$ .

The different theorems that we have seen so far can be relativized, in particular Levin/Schnorr's Theorem 18-2.1, for which we also relativize the notion of prefix-free complexity:

**Definition 2.2.** Given an oracle machine  $M$  such that  $M$  is prefix-free on its oracle  $X$ , we denote by  $K_M^X(\sigma)$  the length of the smallest string  $\tau$  such that  $M(X, \tau) \downarrow = \sigma$ .  $\diamond$

Note that given an oracle machine  $M$ , it is possible for  $M$  to be prefix-free on some of its oracles, but not on others. Likewise, given an  $\Pi_2^0$  oracle class  $\bigcap_n \mathcal{U}_n$ , it is possible for this class to be a Martin-Löf test on some of its oracles, but not all. To study this phenomenon we introduce the following notation.

### Notation

For an oracle machine  $M$  we will write  $M^X : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  for the machine used with the oracle  $X$ . Given an oracle  $\Pi_2^0$  class  $\bigcap_n \mathcal{U}_n$  we will write  $\bigcap_n \mathcal{U}_n^X$  for the corresponding  $\Pi_2^0(X)$  class.

Given an oracle  $X$ , the existence of an oracle machine which is prefix-free and universal on the oracle  $X$  does not present any difficulty. It is however possible to go further, and to show the existence of an oracle machine which is both prefix-free and universal for all oracles.

### Theorem 2.3

*There exists an oracle machine  $U$  such that for any oracle  $X \in 2^{\mathbb{N}}$ :*

- (1) *The machine  $U^X$  is prefix-free*
- (2) *For any oracle machine  $M$  such that  $M^X$  is prefix-free we have  $K_U^X(\sigma) \leq^+ K_M^X(\sigma)$  for any string  $\sigma$ .*

PROOF. It suffices to note that a relativization of Theorem 16-2.2's proof gives a uniform procedure, and therefore defines a Turing functional.  $\blacksquare$

The following corollary follows from the relativization of Levin/Schnorr's Theorem 18-2.1, which states that a string is MLR iff each of its prefixes is incompressible.

### Corollary 2.4

*There is a  $\Pi_2^0$  oracle class which is an  $X$ -Martin-Löf test uniformly universal in each oracle  $X$ .*

PROOF. It suffices to consider the following  $\Pi_2^0(X)$  class:

$$\{Y : \forall c \exists n K^X(Y \upharpoonright_n) \leq n - c\}.$$

According to Theorem 18-2.1 this class is an  $X$ -Martin-Löf test for any oracle  $X$ . According to the previous theorem it is uniformly  $\Pi_2^0(X)$  for any

oracle  $X$ . ■

This uniform universality in each oracle will be useful to us from time to time, we see a first use of it with the elegant theorem of van Lambalgen:

**Theorem 2.5 (van Lambalgen [226])**

*Let  $X, Y$  be two sets. Then  $X \oplus Y$  is MLR iff  $X$  is MLR and  $Y$  is MLR( $X$ ).*

PROOF. Note that we have  $\lambda([\sigma_1 \oplus \sigma_2]) = \lambda([\sigma_1]) \times \lambda([\sigma_2])$ . Suppose  $X$  not MLR or  $Y$  not MLR( $X$ ) in order to show  $X \oplus Y$  not MLR. By symmetry we can assume  $Y$  not MLR( $X$ ). According to Corollary 2.4, let  $\bigcap_n \mathcal{U}_n$  be a universal Martin-Löf test for all oracles. Given a string  $\sigma$  we will denote by  $\mathcal{U}_n^\sigma$  the open piece which is enumerated with the oracle  $\sigma$ .

Since  $Y$  is not MLR( $X$ ) then  $Y \in \bigcap_n \mathcal{U}_n^X$ . For all  $n, m$  we then define the following  $\Sigma_1^0$  class:

$$\mathcal{U}_{n,m} = \bigcup_{|\tau|=m} \{[\tau] \oplus [\sigma] : [\sigma] \subseteq \mathcal{U}_n^\tau \text{ and } |\sigma| = |\tau|\}.$$

For all  $m, n$  we have

$$\lambda(\mathcal{U}_{n,m}) \leq \sum_{|\tau|=m} \lambda([\tau]) \times \lambda(\mathcal{U}_n^\tau) \leq \lambda(\mathcal{U}_n) \sum_{|\tau|=m} \lambda([\tau]) \leq \lambda(\mathcal{U}_n) \leq 2^{-n}.$$

Moreover for all  $m, n$  we have  $\mathcal{U}_{n,m} \subseteq \mathcal{U}_{n,m+1}$ , which implies  $\lambda(\bigcup_m \mathcal{U}_{n,m}) \leq 2^{-n}$  for all  $n$ . It is also clear that  $X \oplus Y \in \bigcup_m \mathcal{U}_{n,m}$ . So  $\bigcap_n \bigcup_m \mathcal{U}_{n,m}$  is a Martin-Löf test which captures  $X \oplus Y$ .

Suppose now  $X \oplus Y$  not MLR and captured by a Martin-Löf test  $\bigcap_n \mathcal{U}_n$ . Let  $\mathcal{U}_n^\sigma$  be the  $\Sigma_1^0$  class given by  $\{Y : [\sigma \oplus Y \upharpoonright_{|\sigma|}] \subseteq \mathcal{U}_n\}$ . Let  $\mathcal{V}_n^X$  be the  $\Sigma_1^0(X)$  class given by  $\bigcup_{\sigma \prec_X} \mathcal{U}_{2n}^\sigma$ . Suppose first  $\lambda(\mathcal{V}_n^X) \leq 2^{-n}$  for any  $n$  sufficiently large. Then,  $\bigcap_n \mathcal{V}_n^X$  is an  $X$ -Martin-Löf test capturing  $Y$ . Otherwise there are infinitely many  $n$  such that  $\lambda(\mathcal{V}_n^X) > 2^{-n}$ . For any  $n$  we then define  $\mathcal{S}_n$  as being the  $\Sigma_1^0$  class equal to  $\{Z : \lambda(\mathcal{V}_n^Z) > 2^{-n}\}$ . Let us show that for all  $n$  we have  $\lambda(\mathcal{S}_n) \leq 2^{-n}$ . Let  $W_n$  be the minimal prefix-free set describing  $\mathcal{S}_n$ . We have by definition:

$$\sum_{\tau \in W_n} \lambda(\tau) \times \lambda(\mathcal{U}_{2n}^\tau) \leq \lambda(\mathcal{U}_{2n}) \sum_{\tau \in W_n} \lambda(\tau) \leq \lambda(\mathcal{U}_{2n}) \leq 2^{-2n}$$

however for all  $\tau \in W_n$  we have  $\lambda(\mathcal{U}_{2n}^\tau) > 2^{-n}$ , which gives us:

$$\sum_{\tau \in W_n} \lambda(\tau) \times 2^{-n} \leq \lambda(\mathcal{U}_{2n}) \leq 2^{-2n}.$$

We deduce  $\sum_{\tau \in W_n} \lambda(\tau) \leq 2^{-n}$  and therefore  $\lambda(\mathcal{S}_n) \leq 2^{-n}$  for all  $n$ . It follows that  $(\mathcal{S}_n)_{n \in \mathbb{N}}$  is a Solovay test, capturing all the sets which are in infinitely many  $\mathcal{S}_n$ . So  $X$  is not MLR. ■

### Corollary 2.6

*Let  $X, Y$  be two MLR sets. Then  $X$  is MLR( $Y$ ) iff  $Y$  is MLR( $X$ ).*

The preceding corollary implies in particular that if  $X$  is random, then it is atypical to make  $X$  non-random.

### Corollary 2.7

*Let  $X$  be an MLR set. Then,  $\{Y \in 2^{\mathbb{N}} : X \text{ is not MLR}(Y)\}$  is a class of measure 0.*

PROOF. If  $X$  is non MLR( $Y$ ) then either  $Y$  is non MLR, or  $Y$  is MLR and in this case  $Y$  is non MLR( $X$ ). In particular the class  $\{Y \in 2^{\mathbb{N}} : X \text{ is non MLR}(Y)\}$  is included in the union of two classes of measure 0. ■

The relativization of Martin-Löf randomness obviously allows us to obtain stronger randomness notions. Any oracle  $X$  with enough computational power will be able “de-randomize” something. But how much power exactly does it takes to make non-random something that was? Is there actually a non-computable set  $X$  which is too weak to capture some MLR sets within an  $X$ -test? This question brings us to the following notion.

**Definition 2.8.** A set  $X$  is *low for Martin-Löf randomness* if any MLR set is MLR( $X$ ). ◇

We will see that the class of sets which are low for the Martin-Löf randomness is a little larger than that of the computable ones. This is one of the most beautiful and difficult results of Algorithmic Randomness: this class coincides with that of the K-trivial sets.

## 3. 2-randomness

Martin-Löf’s tests are  $\Pi_2^0$  classes. Why basically this restriction? One could quite consider classes of measure 0 of arbitrary complexity in order to capture more sets. We see here the first level of this hierarchy.

**Definition 3.1.** A set  $Z$  is *2-random* if it does not belong to any  $\Pi_3^0$  class of the form  $\bigcap_n \mathcal{B}_n$  such that  $\lambda(\mathcal{B}_n) \leq 2^{-n}$  for all  $n$ . ◇

There is an equivalent characterization, which is often the one used.

**Theorem 3.2**

*The 2-random sets are exactly the  $\text{MLR}(\emptyset')$  sets.*

PROOF. Consider first a  $\Pi_2^0(\emptyset')$  class  $\bigcap_n \mathcal{U}_n$ . Each  $\mathcal{U}_n$  is therefore a  $\Sigma_1^0(\emptyset')$  class described by a  $\Sigma_1^0(\emptyset')$  set  $W_n \subseteq 2^{<\mathbb{N}}$ . According to Corollary 5-5.4 and Proposition 5-3.3 the set  $W_n$  is  $\Sigma_2^0$ , that is  $W_n = \{\sigma \in 2^{<\mathbb{N}} : \exists x_1 \forall x_2 R(\sigma, x_1, x_2)\}$  for a computable predicate  $R$ . We define  $\mathcal{F}_{x_1, \sigma}$  as being the  $\Pi_1^0$  class equal to  $[\sigma]$  if  $\forall x_2 R(\sigma, x_1, x_2)$  and equal to the empty set otherwise. We have  $\mathcal{U}_n = \bigcup_{x_1, \sigma} \mathcal{F}_{x_1, \sigma}$  which is therefore a  $\Sigma_2^0$  set. So  $\bigcap_n \mathcal{U}_n$  is a  $\Pi_3^0$  class.

Consider now a  $\Pi_3^0$  class of the form  $\bigcap_n \mathcal{B}_n$  where each  $\mathcal{B}_n$  is a  $\Sigma_2^0$  class for which  $\lambda(\mathcal{B}_n) < 2^{-n}$ . According to Theorem 17-4.4 for each class  $\mathcal{B}_n$  we can uniformly find a  $\Pi_2^0(\emptyset')$  class of the form  $\bigcap_m \mathcal{U}_m^n$  such that  $\mathcal{B}_n \subseteq \bigcap_m \mathcal{U}_m^n$  and for which  $\lambda(\mathcal{U}_m^n \setminus \mathcal{B}_n) \leq 2^{-m}$  for all  $m$ . We therefore have  $\lambda(\mathcal{U}_{n+1}^{n+1}) \leq 2^{-(n+1)} + 2^{-(n+1)} = 2^{-n}$ . It follows that  $\bigcap_n \mathcal{U}_{n+1}^{n+1}$  is a  $\emptyset'$ -Martin-Löf test containing  $\bigcap_n \mathcal{B}_n$ . ■

**Exercize 3.3.** (★) A set is  $n$ -random if it does not belong to any  $\Pi_{n+1}^0$  class of the form  $\bigcap_n \mathcal{B}_n$  with  $\lambda(\mathcal{B}_n) \leq 2^{-n}$ . Show that we can iterate Theorem 3.2: a set is  $\text{MLR}(\emptyset^n)$  iff it is  $n$ -random. ◇

This stronger notion of randomness allows us to show that the class of computably dominated sets is atypical.

**Theorem 3.4**

*No 2-random set is computably dominated. In particular, the computably dominated sets form a class of measure 0.*

We will use the following lemma to prove Theorem 3.4.

**Lemma 3.5 (Monin [157]).** Any  $\Sigma_2^0$  class which intersects any non-empty  $\Pi_1^0$  class contains all the computably dominated sets. ★

PROOF. Let  $\bigcup_n \mathcal{F}_n$  be a  $\Sigma_2^0$  class which intersects all the non-empty  $\Pi_1^0$  classes. Suppose by contradiction that there exists a computably dominated set  $X \notin \bigcup_n \mathcal{F}_n$ . Let  $\bigcap_n \mathcal{U}_n$  be the complement of  $\bigcup_n \mathcal{F}_n$ . We define the  $X$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  which on  $n$  returns the smallest  $t$  such that  $X \in \mathcal{U}_n[t]$ . Let  $g > f$  be a computable function. Then,  $X$  belongs to the  $\Pi_1^0$  class given by  $\bigcap_n \mathcal{U}_n[g(t)]$  and which is disjoint from  $\bigcup_n \mathcal{F}_n$  which is a contradiction. ■

Let us move on to the proof of Theorem 3.4.

**PROOF OF THEOREM 3.4.** We are going to construct for all  $n$  a  $\Sigma_2^0$  class which intersects any non-empty  $\Pi_1^0$  class, and of measure less than  $2^{-n}$ . We will describe a computably enumerable set  $W$  of codes for the  $\Pi_1^0$  classes which constitute our  $\Sigma_2^0$ . Let  $\mathcal{F}_0, \mathcal{F}_1, \dots$  be an enumeration of all the  $\Pi_1^0$  classes.

For any  $e$  let  $\sigma_e$  be the lexicographically smallest string of length  $n + e + 1$ . At step  $t$ , for all  $e \leq t$ , if  $\mathcal{F}_e[t] \cap [\sigma_e] = \emptyset$  we update  $\sigma_e$  which becomes the lexicographically next string of length  $n + e + 1$  (unless it is the last possible such string, in which case nothing is done). Then we enumerate a code for  $\mathcal{F}_e \cap [\sigma_e]$  in  $W$ .

It is clear that if  $\mathcal{F}_e$  is non-empty, then we will have a code of  $\mathcal{F}_e \cap [\sigma_e]$  enumerated in  $W$  for the first string  $\sigma_e$  of length  $n + e + 1$  such that  $\mathcal{F}_e \cap [\sigma_e] \neq \emptyset$ . So  $W$  intersects all the non-empty  $\Pi_1^0$  classes. Moreover, for each  $e$  we add to the  $\Sigma_2^0$  class something of measure bounded by  $2^{-n-e-1}$ . The total measure of the  $\Sigma_2^0$  class is therefore bounded by  $\sum_e 2^{-n-e-1} = 2^{-n}$ .

We have therefore constructed a  $\Pi_3^0$  test of the form  $\bigcap_n \bigcup_e \mathcal{F}_{e,n}$  such that  $\lambda(\bigcup_e \mathcal{F}_{e,n}) < 2^{-n}$ . Then this test does not contain any 2-random. On the other hand, it contains by Lemma 3.5 all the computably dominated sets. ■

According to a relativization of Theorem 18-4.1, any 2-random set computes a  $\text{DNC}(\emptyset')$  function. It is then possible to improve the previous theorem as follow:

**Exercise 3.6. (★★)** Show that no set of  $\text{DNC}(\emptyset')$  degree is computably dominated (we can use a technique similar to that presented in the above proof). ◇

Jockusch and Stephan [102] showed something stronger than in the previous exercise: no set  $X$  such that  $X'$  is of  $\text{DNC}(\emptyset')$  degree is computably dominated. Let us now see an alternative proof of the existence of a high set not computing  $\emptyset'$ . It suffices to consider  $\Omega^{\emptyset'}$ , Chaitin's  $\Omega$  number relativized to  $\emptyset'$ .

**Proposition 3.7.** The set  $\Omega^{\emptyset'}$  is high, but does not compute  $\emptyset'$ . ★

**PROOF.** By using the same algorithm as that of Theorem 16-2.13's proof, with the  $n$  first bits of  $\Omega^{\emptyset'}$ , with the help of  $\emptyset'$ , we can find out which strings  $\sigma$  of length smaller than  $n$  are such that  $U(\emptyset', \sigma) \downarrow$ . We therefore have  $\Omega^{\emptyset'} \oplus \emptyset' \geq_T \emptyset''$ . On the other hand according to Theorem 1.5 the real  $\Omega^{\emptyset'}$  being 2-random, it does not compute  $\emptyset'$ . ■

The class of 2-randoms also makes it possible to demonstrate a well-known property in analysis: let  $f$  be a limit function of continuous functions, then there exists closed sets of arbitrarily large measure on which the restriction of  $f$  is continuous. This result echoes its categorical analogue: any limit of continuous functions is continuous over a co-meager set. We have shown it with Theorem 10-3.20 which says that any 1-generic set is generalized low. We now show an equivalent theorem, but for randomness:

**Theorem 3.8 (Kautz [107])**

*The 2-random sets are generalized low.*

PROOF. Given a code  $e$  we want to know if  $Z$  belongs to the class

$$\mathcal{U}_e = \{X : \Phi_e(X, e) \downarrow\}.$$

Note that  $\mathcal{U}_e$  is a  $\Sigma_1^0$  class. Using  $\emptyset'$  we compute the following  $\emptyset'$ -Martin-Löf test: for all  $e$  we look for the smallest  $t_e$  such that  $\lambda(\mathcal{U}_e \setminus \mathcal{U}_e[t_e]) < 2^{-e}$ . Let  $\mathcal{V}_e = \mathcal{U}_e \setminus \mathcal{U}_e[t_e]$ . The class  $\bigcap_d \bigcup_{e>d} \mathcal{V}_e$  is a  $\emptyset'$ -Martin-Löf test. There is therefore  $d$  such that for all  $e > d$  the set  $Z$  does not belong to  $\mathcal{V}_e$ .

For all  $e > d$ , to know if  $Z \in \mathcal{U}_e$ , it suffices to check whether  $Z \in \mathcal{U}_e[t_e]$ . If this is the case then  $\Phi_e(X, e) \downarrow$  and if it is not the case then as also  $Z \notin \mathcal{U}_e \setminus \mathcal{U}_e[t_e]$  we have  $Z \notin \mathcal{U}_e$  and therefore  $\Phi_e(Z, e) \uparrow$ . ■

Note that in the case of category theory, the same functional is used to compute  $G'$  from  $\emptyset' \oplus G$  for any 1-generic set  $G$ . For the above theorem there is not a unique functional which gives the right result for any 2-random set  $Z$ : it depends on the smallest component  $\mathcal{W}_e$  of a  $\emptyset'$ -Martin-Löf test to which  $Z$  does not belong. One must also hard code the  $e$  first bits of  $Z'$  in the algorithm. This step can however be uniformized thanks to the the halting problem's redundancy of information: to know  $Z'(n)$  for  $n < e$  it suffices to use the padding lemma 3-5.1 to find a code  $m > e$  equivalent to  $n$ . However, the procedure still depends on  $e$ : the larger it is, the closer to 1 will be the subclass of the 2-randoms on which the functional will give the right result. These ideas were precisely formalized by Hoyrup and Rojas [93] via the notions of *randomness deficit* and of *layer-computable functions*.

**Corollary 3.9**

*The class of high sets is of measure 0.*

PROOF. According to Theorem 3.8 if  $Z$  is 2-random then  $Z' \geq_T \emptyset''$  iff  $Z \oplus \emptyset' \geq_T \emptyset''$ . Also by a relativization of Sacks' theorem 18-3.2's proof we have  $\lambda(\{X : X \oplus \emptyset' \geq_T \emptyset''\}) = 0$ . So the class of high sets is of measure 0. ■

## 4. Incomplete random

Incomplete random numbers — those which do not compute  $\emptyset'$  — form an interesting randomness notion, just a little stronger than MLR and much weaker than weak 2-randomness. This study was really initiated by Franklin and Ng who discovered the notion of *difference test*, allowing to capture them.

**Definition 4.1 (Franklin and Ng [60]).** A *difference test* is given by a  $\Pi_2^0$  class  $\bigcap_n \mathcal{U}_n$  and a  $\Pi_1^0$  class  $\mathcal{F}$  such that  $\lambda(\mathcal{U}_n \cap \mathcal{F}) \leq 2^{-n}$  for all  $n$ . A set  $Z$  is *captured* by the test if  $Z \in \bigcap_n \mathcal{U}_n \cap \mathcal{F}$ . Otherwise  $Z$  *passes* the test. A set  $Z$  is a *difference random* if  $Z$  passes all difference tests.  $\diamond$

The notion of difference test corresponds in a way to the following : after having enumerated a string  $\sigma$  in an open component, one keeps the right to change his mind, and may finally remove this string from the enumeration. However, one cannot change his mind a second time and finally decide to put the string  $\sigma$  back.

**Theorem 4.2 (Franklin and Ng [60])**

Let  $Z$  be an MLR set. The following statements are equivalent:

- (1)  $Z$  computes  $\emptyset'$ .
- (2)  $Z$  is not difference random.

In order to show Theorem 4.2 we will use a lemma which has its own interest: if  $X$  is MLR, then “few” sets can compute prefixes of  $X$ .

**Lemma 4.3.** Let  $Z$  be an MLR set. Let  $\Phi$  be a functional. Then, there exists a constant  $c \in \mathbb{N}$  such that  $\lambda(\{X : \Phi(X) \succeq Z \upharpoonright_n\}) \leq 2^{-n}2^c$  for all  $n \in \mathbb{N}$ .  $\star$

PROOF. Suppose that for any constant  $c \in \mathbb{N}$  there exists  $n \in \mathbb{N}$  such that

$$\lambda(\{X : \Phi(X) \succeq Z \upharpoonright_n\}) > 2^{-n}2^c.$$

Let us show that  $Z$  is not MLR. Let  $\mathcal{U}_c$  be the  $\Sigma_1^0$  class generated by the strings  $\sigma$  such that  $\lambda(\{X : \Phi(X) \succeq \sigma\}) > 2^{-|\sigma|}2^c$ . Let  $W$  be a minimal prefix-free set of strings such that  $\mathcal{U}_c = [W]$ . In particular, all  $\sigma \in W$  satisfies the above inequality. Then,

$$\lambda(\mathcal{U}_c) = \sum_{\sigma \in W} 2^{-|\sigma|} \leq 2^{-c} \sum_{\sigma \in W} \lambda(\{X : \Phi(X) \succeq \sigma\})$$

As the class  $W$  is prefix-free then for two distinct strings  $\sigma_1, \sigma_2 \in W$  we have  $\{X : \Phi(X) \succeq \sigma_1\} \cap \{X : \Phi(X) \succeq \sigma_2\} = \emptyset$ . By additivity of the

measure we therefore have  $\sum_{\sigma \in W} \lambda(\{X : \Phi(X) \succeq \sigma\}) \leq 1$  and therefore  $\lambda(\mathcal{U}_c) \leq 2^{-c}$ . We can thus capture  $Z$  by the Martin-Löf test given by  $\bigcap_c \mathcal{U}_c$ . ■

We can now show the characterization of Franklin and Ng.

**PROOF OF THEOREM 4.2.** Suppose  $Z$  computes  $\emptyset'$ . Then,  $Z$  also computes Chaitin's  $\Omega$  via a functional  $\Phi$ . Let  $c$  be the constant of lemma 4.3 such that  $\lambda(\{X : \Phi(X) \succeq \Omega \upharpoonright_n\}) < 2^{-n}2^c$  for all  $n$ . Let  $\mathcal{C}_{n,s}$  be the  $\Sigma_1^0$  class given by  $\{X : \Phi(X) \succeq \Omega_s \upharpoonright_n\}$  where  $\Omega_s$  is the approximation of  $\Omega$  at step  $s$ . Now let  $\mathcal{C}'_{n,s}$  be the class  $\mathcal{C}_{n,s}$  for which the enumeration is blocked if necessary so that the measure does not exceed  $2^{-n}2^c$ .

Let  $\mathcal{U}$  be the union of the classes  $\mathcal{C}'_{n,s}$  for all  $n$  and for all  $s$  such that  $\Omega_s \upharpoonright_n \neq \Omega_{s+1} \upharpoonright_n$ . Note that no element of  $\mathcal{U}$  computes  $\Omega$  via  $\Phi$  because if  $\Omega_s \upharpoonright_n \neq \Omega_{s+1} \upharpoonright_n$  then  $\Omega_s \upharpoonright_n \neq \Omega \upharpoonright_n$ . In particular  $Z$  cannot be in  $\mathcal{U}$ . Finally for all  $n$  let  $\mathcal{U}_n$  be the union of the classes  $\mathcal{C}'_{n,s}$  for all  $s$  and let  $\mathcal{F}$  be the complement of  $\mathcal{U}$ .

Let  $s_n$  be such that  $\Omega_{s_n} \upharpoonright_n = \Omega \upharpoonright_n$ . Then,  $Z \in \mathcal{C}_{n,s_n}$  and according to the choice of the constant  $c$  the set  $Z$  also belongs to  $\mathcal{C}'_{n,s_n}$ . So it is clear that  $Z \in \bigcap_n \mathcal{U}_n$  and since  $Z$  is not in  $\mathcal{U}$  it is clear that  $Z \in \mathcal{F} \cap \bigcap_n \mathcal{U}_n$ . Moreover the elements of  $\mathcal{F} \cap \mathcal{U}_n$  are only those which belong to  $\mathcal{C}'_{n,s_n}$  and by definition we have  $\lambda(\mathcal{C}'_{n,s_n}) \leq 2^{-n}2^c$ . So  $\mathcal{F} \cap \bigcap_n \mathcal{U}_n$  is a difference test containing  $Z$ .

Suppose now that  $Z$  is captured by a difference test  $\mathcal{F} \cap \bigcap_n \mathcal{U}_n$ . For all  $n$  let  $\mathcal{V}_n$  be the following  $\Sigma_1^0$  class: if  $n$  is enumerated in  $\emptyset'$  at step  $t$  then  $\mathcal{V}_n$  is defined as  $\mathcal{U}_n[t] \cap \mathcal{F}[s_t]$  where  $s_t$  is the smallest integer such that  $\lambda(\mathcal{U}_n[t] \cap \mathcal{F}[s_t]) \leq 2^{-n}$ . If  $n$  is never enumerated then  $\mathcal{V}_n$  remains empty. The sequence  $(\mathcal{V}_n)_{n \in \mathbb{N}}$  forms a Solovay test. In particular there exists  $n$  such that for all  $m > n$ ,  $Z \notin \mathcal{V}_m$ . We can then compute  $\emptyset'$  using  $Z$  as follows: for any  $m > n$  it suffices to look for the smallest  $t$  such that  $Z \in \mathcal{U}_m[t]$  and see if  $m \in \emptyset'[t]$ . If this is not the case then  $m \notin \emptyset'$  because otherwise we would have  $Z \in \mathcal{V}_m$ . ■

#### left-c.e. MLR

Theorem 4.2's proof works when replacing  $\Omega$  by any left-c.e. MLR. In particular, any left-c.e. MLR is captured by a difference test and is then complete, as we discussed it in Section 16-2.4.

We now see an application of difference tests and incomplete randomness. With measure theory, came the idea of satisfying a property “almost everywhere” in analysis. A property over the reals is true almost everywhere if

the set of reals for which it is false has the measure 0. A classical theorem tells us for instance that a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  of Borel graph is almost everywhere the limit of a sequence of continuous function.

One of the fields of study of Algorithmic Randomness consists in determining the exact level of randomness that is required for properties which are true almost everywhere. We see an example of this here, with Lebesgue's density theorem and difference randomnesses. This study will also be useful in the forthcoming chapter on the K-trivial sets.

Let us first present Lebesgue's density theorem. For the moment, we have only exposed Lebesgue's density lemma, that we recall here:

**Lemma (18-3.1, Lebesgue density).** Let  $\mathcal{B}$  be a Borel class of positive measure. Then, for all  $\varepsilon > 0$  there exists a cylinder  $[\sigma]$  such that  $\lambda(\mathcal{B} \mid [\sigma]) > 1 - \varepsilon$ . ★

Lebesgue's density theorem is a reinforcement of this lemma. We introduce for this the concept of asymptotic value of a Borel class density along prefixes of a set.

**Definition 4.5.** Let  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  be a Borel class and  $Z \in 2^{\mathbb{N}}$ .

- (1) We note  $\underline{\rho}(\mathcal{B} \mid Z)$  the lower limit of the density occupied by  $\mathcal{B}$  within the prefixes of  $Z$ , that is to say

$$\underline{\rho}(\mathcal{B} \mid Z) = \liminf_{\sigma \prec Z} \lambda(\mathcal{B} \mid [\sigma]).$$

- (2) We note in a similar way  $\bar{\rho}(\mathcal{B} \mid Z)$  the upper limit of the density occupied by  $\mathcal{B}$  within the prefixes of  $Z$ , that is to say

$$\bar{\rho}(\mathcal{B} \mid Z) = \limsup_{\sigma \prec Z} \lambda(\mathcal{B} \mid [\sigma]).$$

We will say that  $\underline{\rho}(\mathcal{B} \mid Z)$  is the *lower density* of  $Z$  inside  $\mathcal{B}$  and  $\bar{\rho}(\mathcal{B} \mid Z)$  its *higher density*. ◇

We can now state Lebesgue's density theorem:

**Theorem 4.6 (Lebesgue density)**

Let  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  be a Borel class of positive measure. Then the class  $\{Z \in \mathcal{B} : \underline{\rho}(\mathcal{B} \mid Z) = 1\}$  has the same measure as that of  $\mathcal{B}$ .

We will use the following lemma to show it.

**Lemma 4.7.** Let  $\mathcal{F}$  be a closed class and  $\sigma \in 2^{<\mathbb{N}}$  a string such that  $\lambda(\mathcal{F} \mid [\sigma]) \geq \varepsilon$ . Then, there exists  $X \in \mathcal{F} \cap [\sigma]$  such that  $\lambda(\mathcal{F} \mid [X \upharpoonright_n]) \geq \varepsilon$

for all  $n > |\sigma|$ . ★

PROOF. It suffices to see that for any string  $\tau$  and any Borel class  $\mathcal{A}$ , if  $\lambda(\mathcal{A} \mid [\tau]) \geq \varepsilon$  then for  $i = 0$  or  $i = 1$  we will have  $\lambda(\mathcal{A} \mid [\tau i]) \geq \varepsilon$ . Indeed if  $\mathcal{A}$  occupies for less than  $\varepsilon$  in  $[\tau 0]$  and in  $[\tau 1]$ , it will also occupy less than  $\varepsilon$  in  $[\tau]$ . We then construct in this way a set  $X \succ \sigma$  such that  $\lambda(\mathcal{F} \mid [X \upharpoonright_n]) \geq \varepsilon$  for all  $n > |\sigma|$ . Since  $\mathcal{F}$  is a closed class and  $\mathcal{F} \cap [X \upharpoonright_n]$  is non-empty for all  $n$  then  $X \in \mathcal{F}$ . ■

PROOF OF THEOREM 4.6. Suppose by contradiction that the class  $\mathcal{B}' = \{Z \in \mathcal{B} : \underline{\rho}(\mathcal{B} \mid Z) < 1\}$  has a positive measure. Let  $\mathcal{B}'_\varepsilon = \{Z \in \mathcal{B} : \underline{\rho}(\mathcal{B} \mid Z) < 1 - \varepsilon\}$ . As  $\mathcal{B}' = \bigcup_{\varepsilon \in \mathbb{Q} \cap [0,1]} \mathcal{B}'_\varepsilon$ , by countable additivity there must exist  $\varepsilon$  such that the measure of  $\mathcal{B}'_\varepsilon$  is positive. Finally, using Theorem 17-4.4, let  $\mathcal{F} \subseteq \mathcal{B}'_\varepsilon$  be a closed class of positive measure. Note that for any  $Z \in \mathcal{F}$  we also have  $\underline{\rho}(\mathcal{F} \mid Z) < 1 - \varepsilon$  because  $\underline{\rho}(\mathcal{F} \mid [\sigma]) \leq \underline{\rho}(\mathcal{B} \mid [\sigma])$  for any string  $\sigma$ .

Now by Lebesgue's density lemma 18-3.1 there must be a string  $\sigma$  such that  $\lambda(\mathcal{F} \mid [\sigma]) > 1 - \varepsilon$ . According to Lemma 4.7 there exists then  $Z \in \mathcal{F}$  such that  $\lambda(\mathcal{F} \mid [Z \upharpoonright_n]) \geq 1 - \varepsilon$  for all  $n > |\sigma|$  and therefore such that  $\underline{\rho}(\mathcal{F} \mid Z) \geq 1 - \varepsilon$ , which contradicts  $\underline{\rho}(\mathcal{F} \mid Z) < 1 - \varepsilon$ . So the class  $\mathcal{B}' = \{Z \in \mathcal{B} : \underline{\rho}(\mathcal{B} \mid Z) < 1\}$  is of measure 0. ■

We cannot of course expect to satisfy, with a fixed randomness notion, Lebesgue's density theorem for any Borel class, but we can do so for a given level in the effective Borel hierarchy. We will limit ourselves to the simplest case:  $\Pi_1^0$  classes.

**Definition 4.8.** A set  $Z$  is said to be of *lower (resp. upper) positive density* if  $\underline{\rho}(\mathcal{F} \mid Z) > 0$  (resp.  $\bar{\rho}(\mathcal{F} \mid Z) > 0$ ) for any  $\Pi_1^0$  class  $\mathcal{F}$  containing  $Z$ . A set  $Z$  is said to be of *lower (resp. upper) density 1* if  $\underline{\rho}(\mathcal{F} \mid Z) = 1$  (resp.  $\bar{\rho}(\mathcal{F} \mid Z) = 1$ ) for any  $\Pi_1^0$  class  $\mathcal{F}$  containing  $Z$ . ◇

**Exercise 4.9. (★★)** Show that the MLR sets are of upper density 1. ◇

We now give a characterization of incomplete random sets, related to Lebesgue's density theorem.

**Theorem 4.10 (Bienvenu, Hölz, Miller, Nies [18])**

Let  $Z$  be an MLR set. Then, the following statements are equivalent:

1.  $Z$  is an incomplete set (i.e. with  $Z \not\leq_T \emptyset'$ ).

2.  $Z$  is a set of positive lower density.

PROOF. Suppose  $Z$  of lower density zero and let  $\mathcal{F}$  be a  $\Pi_1^0$  class such that  $\underline{\rho}(\mathcal{F} \mid Z) = 0$ . Let  $\mathcal{U}_n$  be the  $\Sigma_1^0$  class generated by the strings  $\sigma$  such that  $\lambda(\mathcal{F} \mid [\sigma]) < 2^{-n}$ . It is clear that  $\lambda(\mathcal{U}_n \cap \mathcal{F}) \leq 2^{-n} \lambda(\mathcal{F}) \leq 2^{-n}$ . As  $\underline{\rho}(\mathcal{F} \mid Z) = 0$  then  $Z$  is captured by the test  $\mathcal{F} \cap \bigcap_n \mathcal{U}_n$ . So  $Z$  is captured by a difference test. According to Theorem 4.2,  $Z \geq_T \emptyset'$ .

Now suppose that  $Z \geq_T \emptyset'$ . Then, according to Theorem 4.2,  $Z$  is captured by a difference test  $\mathcal{F} \cap \bigcap_n \mathcal{U}_n$ . Let us show that we have  $\underline{\rho}(\mathcal{F} \mid Z) = 0$ . For each  $r \in \mathbb{N}$  we will build a Martin-Löf test  $\bigcap_n \mathcal{V}_n$  such that for all  $X \in 2^{\mathbb{N}}$ , if  $X \in \mathcal{F} \cap \bigcap_n \mathcal{U}_n$  and  $X \notin \bigcap_n \mathcal{V}_n$  then there exists a prefix  $\sigma \prec X$  such that  $\lambda(\mathcal{F} \mid [\sigma]) < 2^{-r}$ . We define  $\mathcal{V}_0 = \mathcal{U}_0$ , then once  $\mathcal{V}_n$  is defined, for any string  $\sigma$  enumerated in  $\mathcal{V}_n$  we enumerate in  $\mathcal{V}_{n+1}$  the strings of  $\mathcal{U}_{|\sigma|+r+1} \cap [\sigma]$  while blocking the enumeration if necessary in order not to exceed a measure of  $2^{-|\sigma|}(1 - 2^{-r-1})$ . We have  $\lambda(\mathcal{V}_{n+1}) \leq (1 - 2^{-r-1})\lambda(\mathcal{V}_n)$  and we can therefore limit the measure of  $\mathcal{V}_n$  by a computable and decreasing function which goes towards 0. We can then easily transform  $\bigcap_n \mathcal{V}_n$  into a Martin-Löf test. Since  $Z$  is MLR then  $Z \notin \bigcap_n \mathcal{V}_n$ . Let  $n$  be the smallest integer such that  $Z \notin \mathcal{V}_n$ . Let  $\sigma \prec Z$  be such that  $\sigma$  is enumerated in  $\mathcal{V}_{n-1}$ . As  $Z \in \bigcap_n \mathcal{U}_n$  this means that the enumeration of  $\mathcal{V}_n$  has been blocked and we have  $\lambda(\mathcal{U}_{|\sigma|+r+1} \mid [\sigma]) > 1 - 2^{-r-1}$ . Suppose now by contradiction  $\lambda(\mathcal{F} \mid [\sigma]) > 2^{-r}$ . Then,  $\lambda(\mathcal{U}_{|\sigma|+r+1} \cap \mathcal{F} \mid [\sigma]) > 2^{-r-1}$  and therefore  $\lambda(\mathcal{F} \cap \mathcal{U}_{|\sigma|+r+1}) > 2^{-|\sigma|-r-1}$  which contradicts  $\lambda(\mathcal{F} \cap \mathcal{U}_{|\sigma|+r+1}) < 2^{-|\sigma|-r-1}$ . So  $\lambda(\mathcal{F} \mid [\sigma]) < 2^{-r}$ .

As we can do the same thing for all  $r$  there exists for all  $r$  a prefix  $\sigma \prec Z$  such that  $\lambda(\mathcal{F} \mid [\sigma]) < 2^{-r}$  and  $Z$  is therefore of lower density zero in  $\mathcal{F}$ . ■



# Chapter 20

## The K-trivials

K-trivials constitute one of the most fascinating classes of Algorithmic Randomness, and which has not ceased to surprise through the discoveries concerning it. The central theorem is undoubtedly the following: a set  $A$  is K-trivial iff it is low for randomness. We start our adventure with a study of the low for randomness notion, which will culminate with the remarkable “hungry sets” proof.

### 1. Lowness and bases for randomness

We have seen with Definition 19-2.8 the concept of oracle which do not modify the class of Martin-Löf randoms. We see here an a priori even stronger concept: oracles not modifying prefix-free complexity.

#### 1.1. Lowness

In general, any relativizable property  $P$  induces a lowness notion, where  $X$  is low for  $P$  iff  $P^X$  and  $P^\emptyset$  coincide. For example, a set  $X$  is low (for the Turing jump) if  $X' = \emptyset'$ . Likewise, a set  $X$  is low for Martin-Löf randomness if any MLR set is  $\text{MLR}(X)$ . We define the corresponding notion for prefix-free complexity.

**Definition 1.1.** A set  $A \in 2^{\mathbb{N}}$  is *low-for-K* if  $K^A(\sigma) \leq^+ K(\sigma)$  for any string  $\sigma$ .  $\diamond$

It is clear that any computable set is low-for-K, and it is possible to show that these are not the only ones.

**Exercise 1.2. (★★)** Using the same technique as that of the proof of Theorem 16-4.5, show that there are non-computable c.e. sets which are low-for-K.  $\diamond$

Let us see two direct and easy to show implications: if  $A$  is low-for-K then it is K-trivial and also low for Martin-Löf randomness.

**Proposition 1.3.** If  $A$  is low-for-K then  $A$  is K-trivial.  $\star$

PROOF. For any oracle  $X$  we have  $K^X(X \upharpoonright_n) \leq^+ K^X(n) \leq^+ K(n)$  for all  $n$ . For the first inequality, given  $n$  it suffices to use the oracle  $X$  to produce  $X \upharpoonright_n$ . For the second inequality, if a machine can produce  $n$  with no oracle, it can also do so with an oracle. If  $A$  is low-for-K, we also have  $K(A \upharpoonright_n) \leq^+ K^A(A \upharpoonright_n)$ , so  $K(A \upharpoonright_n) \leq^+ K(n)$  for all  $n$ .  $\blacksquare$

**Proposition 1.4.** If  $A$  is low-for-K then  $A$  is low-for-MLR.  $\star$

PROOF. Suppose  $A$  is low-for-K and consider an  $A$ -Martin-Löf test  $\bigcap_n \mathcal{U}_n$ . According to Theorem 18-2.1 relativized, for all  $X$  in  $\bigcap_n \mathcal{U}_n$ , for all  $c$  there exists  $m$  such that  $K^A(X \upharpoonright_m) \leq m - c$  and therefore such that  $K(X \upharpoonright_m) \leq m - c + d$  for some constant  $d$  independent of  $m$  and  $c$ . So for all  $X$  in  $\bigcap_n \mathcal{U}_n$ , for all  $c$  there exists  $m$  such that  $K(X \upharpoonright_m) \leq m - c$ . According to Corollary 18-2.2, the  $A$ -Martin-Löf test  $\bigcap_n \mathcal{U}_n$  is then included in the universal Martin-Löf test (without oracle).  $\blacksquare$

**Exercise 1.5. (★)** Show that every low-for-K set is of low degree.  $\diamond$

## 1.2. Basis for Randomness and Hungry Sets

We now see that the class of low-for-K coincides with that of the low for Martin-Löf randomness. We use for this a third concept which has its own interest.

**Definition 1.6.** A set  $A \in 2^{\mathbb{N}}$  is a *basis for Martin-Löf randomness* if there exists a set  $Z \in \text{MLR}(A)$  which computes  $A$ .  $\diamond$

We have seen with Theorem 18-3.2 that for any non-computable set  $A$ , the class of sets computing  $A$  is of measure 0. Moreover for a fixed functional  $\Phi$  the class  $\{X \in 2^{\mathbb{N}} : \forall n \Phi(X, n) \downarrow = A(n)\}$  is a  $\Pi_2^0(A)$  class of measure 0. We deduce that for any non-computable set  $A$  no set computing  $A$  is weak 2-random relative to  $A$ . On the other hand, one cannot always show that such a set is not  $\text{MLR}(A)$ . It suffices to consider a non-computable set  $A$  which is also low for Martin-Löf randomness, and according to Theorem 18-3.4 there exists an  $\text{MLR}$  set and therefore  $\text{MLR}(A)$  which computes  $A$ . In

particular if  $A$  is low-for- $K$  then  $A$  is a basis for randomness. The following theorem shows that the converse is true, using a sophisticated and clever construction, the so called “hungry sets” proof.

**Theorem 1.7 (Hirschfeldt, Nies and Stephan [89])**

*If  $A$  is a basis for randomness, then  $A$  is low-for- $K$ .*

PROOF. The reader can use Figure 1.8 to follow the proof. Let  $Z$  be an  $\text{MLR}(A)$  set such that  $\Phi(Z) = A$  for a functional  $\Phi$ . Let  $U$  be the universal oracle prefix-free machine of Theorem 19-2.3. We can see  $U$  as an enumeration of triples  $(\rho, \tau, x)$  meaning then  $U^\rho(\tau) \downarrow = x$ . We can consider without loss of generality that if  $U^X(\tau) \downarrow = x$  for a set  $X$ , then there is a unique prefix  $\rho \prec X$  such that  $U^\rho(\tau) \downarrow = x$ .

We will describe an algorithm parameterized by an integer  $d$ . Also for all  $d$  the algorithm will enumerate a bounded requests set  $L_d$ , and we will show that for  $d$  sufficiently large, the prefix-free machine  $M_d$  resulting from this requests set will be such that  $K_{M_d}(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . During the algorithm, for each triplet  $(\rho, \tau, x)$ , we will enumerate “hungry sets”  $C_{\tau,x}^\rho \subseteq 2^{<\mathbb{N}}$  which ask to be fed by finite strings until they are sated. A hungry set will continue to “eat” finite strings as long as it is not full, but will also refuse to eat anything that would feed it more than reason: each set  $C_{\tau,x}^\rho$  wants to reach a weight of  $2^{-d}2^{-|\tau|}$ , but will always refuse to exceed that weight.

The algorithm with parameter  $d$  is as follows: at computation step  $t$  for any  $\langle \rho, \tau, x \rangle \leq t$ , if  $U^\rho(\tau)[t] \downarrow = x$  we consider the clopen class  $\mathcal{U} = \{X : \Phi(X)[t] \geq \rho\}$ . Let  $W$  be a prefix-free set of strings such that  $[W] = \mathcal{U}$ . The goal is to add each string  $\sigma \in W$  into  $C_{\tau,x}^\rho$ , but while keeping the hungry sets — seen as open classes — pairwise disjoint. So if a prefix of  $\sigma$  is in another hungry set we will enumerate nothing at all, and if an extension of  $\sigma$  is in another hungry set, we will only enumerate the extensions of  $\sigma$  that are not yet in any of them. Formally for each string  $\sigma \in W$  we look for two sets of finite strings  $B_{0,\sigma}$  and  $B_{1,\sigma}$  such that  $[B_{0,\sigma}] \cup [B_{1,\sigma}] = [\sigma]$ , such that  $[B_{0,\sigma}] \cap [B_{1,\sigma}] = \emptyset$ , and such that  $[\iota]$  is included in a hungry set for each  $\iota \in B_{0,\sigma}$  and  $[\iota]$  has an empty intersection with all hungry sets for each  $\iota \in B_{1,\sigma}$ . Finally for any  $\sigma \in W$  and any string  $\iota \in B_{1,\sigma}$  we enumerate in order  $\iota$  in  $C_{\tau,x}^\rho$ , provided that we always keep  $\lambda([C_{\tau,x}^\rho]) \leq 2^{-d}2^{-|\tau|}$ . If enumerating a string changes the measure of  $[C_{\tau,x}^\rho]$  above  $2^{-d}2^{-|\tau|}$  then it is not enumerated. Finally, if after the enumeration of all these strings we have  $\lambda([C_{\tau,x}^\rho]) \geq 2^{-d-1}2^{-|\tau|}$  we enumerate  $\langle x, |\tau| + d + 1 \rangle$  in the bounded requests set  $L_d$ . This concludes the construction.

Let us show that  $L_d$  is indeed a bounded requests set for all  $d$ . We enumerate at most one  $\langle x, |\tau| + d + 1 \rangle$  element in  $L_d$  for each hungry set, in which

case this enumeration occurs when the corresponding  $C_{\tau,x}^\rho$  set is “half full”, that is, if  $\lambda([C_{\tau,x}^\rho]) \geq 2^{-d-1-|\tau|}$ . We have in particular

$$\begin{aligned} \text{weight}(L_d) &= \sum_{\langle x, |\tau|+d+1 \rangle \in L_d} 2^{-d-1-|\tau|} \\ &\leq \sum_{\langle \rho, \tau, x \rangle} \lambda([C_{\tau,x}^\rho]) \end{aligned}$$

It suffices then to notice that by construction, the open classes  $[C_{\tau,x}^\rho]$  are pairwise disjoint. We then have in particular by countable additivity of the measure:

$$\begin{aligned} \text{weight}(L_d) &\leq \sum_{\langle \rho, \tau, x \rangle} \lambda([C_{\tau,x}^\rho]) \\ &= \lambda(\bigcup_{\langle \rho, \tau, x \rangle} [C_{\tau,x}^\rho]) \\ &\leq 1 \end{aligned}$$

The set  $L_d$  is therefore indeed a bounded requests set.

For  $d$  fixed we define  $\mathcal{U}_d^A = \bigcup_{\rho \prec A, \tau, x} [C_{\tau,x}^\rho]$  where the sets  $C_{\tau,x}^\rho$  are those produced by the algorithm with  $d$  as a parameter. Let us show that  $\bigcap_d \mathcal{U}_d^A$  is an  $A$ -Martin-Löf test. We have

$$\lambda(\mathcal{U}_d^A) \leq \sum_{\rho \prec A, U^\rho(\tau) \downarrow = x} \lambda([C_{\tau,x}^\rho]) \leq \sum_{\rho \prec A, U^\rho(\tau) \downarrow} 2^{-d} 2^{-|\tau|} \leq 2^{-d} \Omega^A \leq 2^{-d}$$

So  $\bigcap_d \mathcal{U}_d^A$  is indeed an  $A$ -Martin-Löf test.

Now let's finish the proof: since  $Z$  is  $\text{MLR}(A)$  then there exists  $d$  such that  $Z \notin \mathcal{U}_d^A$ . Let us show that for such an integer  $d$ , for any  $\tau, x$  such that  $U^\rho(\tau) \downarrow = x$  for  $\rho \prec A$ , the set  $C_{\tau,x}^\rho$  is always “half full”, that is, the measure of  $[C_{\tau,x}^\rho]$  necessarily reaches  $2^{-d-1-|\tau|}$ . Note that if this is the case then also for any  $\tau, x$  such that  $U^A(\tau) \downarrow = x$  we enumerate  $\langle x, |\tau|+d+1 \rangle$  in  $L_d$ , which implies indeed  $K_{M_d}(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . Let  $\tau, x$  and  $\rho \prec A$  such that  $U^\rho(\tau) \downarrow = x$ . Suppose by contradiction  $\lambda([C_{\tau,x}^\rho]) < 2^{-d-1-|\tau|}$ . Let  $\sigma \prec Z$  be large enough such that  $2^{-|\sigma|} < 2^{-d-1-|\tau|}$  and such that  $\Phi(\sigma) \succeq \rho$ . If at this moment a string  $\sigma'$  such that  $\sigma' \prec \sigma$  or such that  $\sigma \prec \sigma' \prec Z$  is already enumerated in a hungry set  $C_{\tau',x'}^{\rho'}$ , it will necessarily be for  $\rho' \prec A$  since  $\Phi(Z) = A$ . In this case  $Z \in \mathcal{U}_d^A$  which is a contradiction. Otherwise there will be a string  $\iota$  with  $\sigma \preceq \iota \prec Z$  such that  $[\iota]$  at this point has an empty intersection with all hungry sets. As the enumeration of  $\iota$  in  $C_{\tau,x}^\rho$  leaves the measure of  $[C_{\tau,x}^\rho]$  below  $2^{-d} 2^{-|\tau|}$ , then  $\iota$  is enumerated in  $C_{\tau,x}^\rho$  and  $Z \in \mathcal{U}_d^A$  which is still a contradiction. So for all  $\tau, x$  and for  $\rho \prec A$  such that  $U^\rho(\tau) \downarrow = x$ , the measure of the set  $[C_{\tau,x}^\rho]$  necessarily reaches  $2^{-d-1-|\tau|}$  and therefore  $\langle x, |\tau|+d+1 \rangle$  is enumerated in  $L_d$ . So the machine  $M_d$  built from  $L_d$  is such that  $K_{M_d}(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . So  $A$  is low-for-K.  $\blacksquare$

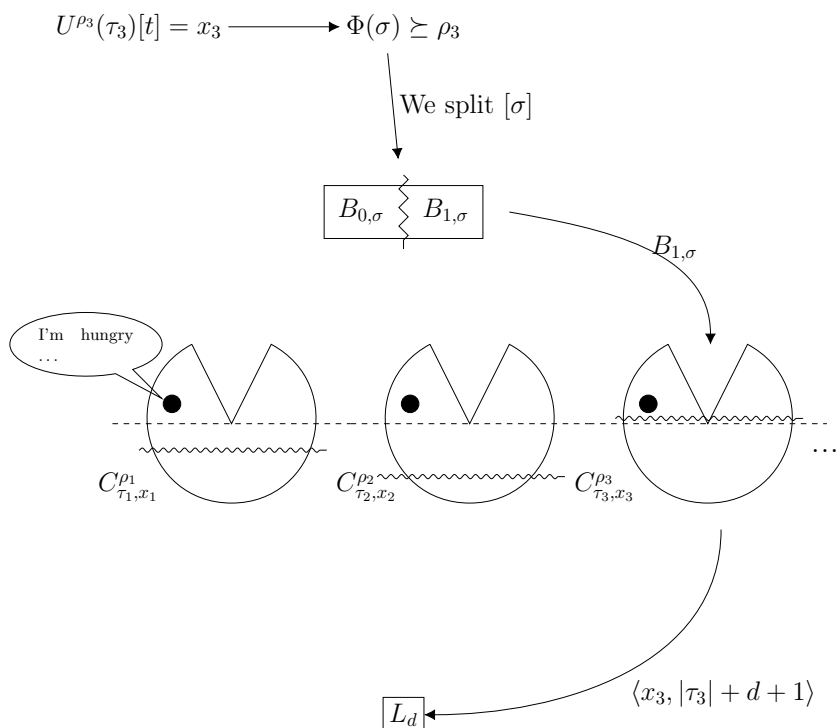


Figure 1.8: Illustration of the hungry sets proof. When  $U^{\rho_3}(\tau_3)[t] = x_3$  we try to feed the hungry set  $C^{\rho_3}_{\tau_3, x_3}$  with strings  $\sigma$  such that  $\Phi(\sigma) \succeq \rho_3$ , while keeping hungry sets pairwise disjoint. This is why we split  $\sigma$  in two parts  $B_{0,\sigma}$  and  $B_{1,\sigma}$ , where  $B_{1,\sigma}$  keeps the part of  $\sigma$  which is not already in some hungry set. If some day  $C^{\rho_3}_{\tau_3, x_3}$  is half full (that is, reaches a weight of  $2^{-|\tau_3|-d-1}$ ), we enumerate  $\langle x_3, |\tau_3| + d + 1 \rangle$  in our bounded requests set  $L_d$ .

**Corollary 1.9 (Hirschfeldt, Nies and Stephan [89])**

*The following three statements are equivalent:*

- (1) *A is low-for-K*
- (2) *A is low-for-MLR*
- (3) *A is a basis for randomness.*

PROOF. (1)  $\rightarrow$  (2) comes from Proposition 1.4. Let us show (2)  $\rightarrow$  (3). Suppose  $A$  is low-for-MLR. According to Theorem 18-3.4 there exists an MLR set  $Z$  such that  $Z \geq_T A$ . Since  $A$  is low-for-MLR then  $Z$  is also

$\text{MLR}(A)$  and so  $A$  is a basis for randomness. Finally,  $(3) \rightarrow (1)$  is given by the previous theorem. ■

Now it's time to get to the heart of the matter, and show that K-trivial sets are all low-for-K.

## 2. Golden run

The hungry sets proof may have whetted the reader's appetite for complex constructions. May he be reassured, as we now approach one of the most difficult result of Algorithmic Randomness, the so-called “golden run” proof, whose purpose is to show the following theorem.

**Theorem 2.1 (Hirschfeldt, Nies [165])**  
*K-trivial sets coincide with low-for-K sets.*

The reader preferring simplicity can however be relieved: there is another proof that all K-trivial sets are low-for-K, which does not rely on such a complicated construction and which we present at the end of this chapter. Let us insist however on the fact that this second proof is not really “simpler”: it uses a series of intermediate results — including the hungry sets proof — each simpler to show, and which put together arrive at the result. It is however perfectly impossible with this second proof to understand *why* K-trivial sets are all low-for-K. On the other hand, the first proof that we present now is admittedly difficult, but honest and straight to the point. The golden run construction, however complex it may be, perfectly shows the mechanisms at work, and anyone brave enough to understand its inner specificities will really know why and how the K-trivials sets are all low-for-K. The second proof of course also has its advantages, if only because each intermediate result of this other demonstration has its own interest.

This section is entirely devoted to the proof of Theorem 2.1. Let  $U$  be the universal prefix-free oracle machine of Theorem 19-2.3. Let  $A$  be a K-trivial set via a constant  $c$ , i.e., such that  $K(A \upharpoonright_n) \leq K(n) + c$  for all  $n$ . We are going to prove that such a set  $A$  is necessarily low-for-K, that is to say that we also have  $K(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . For that, we are going to build a prefix-free machine  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ .

**General idea.** Since  $A$  is K-trivial we know from Corollary 16-4.9 that it is  $\Delta_2^0$ . We will therefore use the approximation of  $A$  to build our machine  $M$ , the goal of which will be to do as well — up to constant —

as the universal machine  $U$  on the oracle  $A$ . To do this, as soon as we have  $U^\rho(\tau)[s] = \sigma$  at a computation step  $s$  for  $\rho \prec A_s$  and for strings  $\tau$  and  $\sigma$ , we would like to enumerate the pair  $(\sigma, |\tau|)$  in a bounded requests set. The problem is obviously that  $\rho \prec A_s$  — the current approximation of  $A$  used to send  $\tau$  to  $\sigma$  — may not be the correct one. If the latter changes, we will have so to speak “lost” our mapping of a string of length  $|\tau|$  to  $\sigma$ , as this mapping may not correspond to what happens with the real oracle  $A$ . We will then say that we lose the quantity  $2^{-|\tau|}$ , which will be deducted from our bounded requests set, the weight of which, let us remember, must not exceed 1.

In order to avoid too much loss, we will not immediately enumerate the pair  $(\sigma, |\tau|)$  in our bounded requests set. We will first ask the current approximation of  $A$ , “proofs” of its solidity. This will be done via the construction of a third-party machine  $M_d$ , also built via a bounded requests set  $L_d$ . Let  $c_d$  be an integer such that  $K(\sigma) \leq K_{M_d}(\sigma) + c_d$  for any string  $\sigma$ . We will enumerate in  $L_d$  the pair  $(n, l)$  for some length  $l$  and some  $n \geq |\rho|$ , then wait to have  $K(A_s \upharpoonright_n)[s] \leq l + c + c_d$  (remember that  $c$  is the K-triviality constant of  $A$ ). Note that if  $L_d$  is really a bounded requests set we must have  $K_{M_d}(n) \leq l$ . So  $K(n) \leq K_{M_d}(n) + c_d \leq l + c_d$ . As  $K(A \upharpoonright_n) \leq K(n) + c$  we must have  $K(A \upharpoonright_n) \leq l + c + c_d$ .

If during this validation waiting, the prefix  $\rho \prec A_s$  remains unchanged, then it will have proved its solidity, and only then will we enumerate the pair  $(\sigma, |\tau|)$  in our bounded requests set. Of course once  $\rho$  has been validated, that does not mean that it will not change anymore, but each time it changes we have forced  $K(A_s \upharpoonright_n)$  to go below some value, for some bad prefix of  $A$ , the goal is to make sure that this cannot happen too often without having  $\sum_\sigma 2^{-K(\sigma)} > 1$ .

The construction will use a process tree, infinitely branching, but of finite height. Each of these processes will attempt to build a machine  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$ . If each process will work on its own attempt to show that  $A$  is low-for K, via its own machine, all the processes will share the same bounded set of requests  $L_d$  whose role is discussed above. The algorithm as a whole uses the fixed point theorem to have access to the constant  $c_d$  associated with the machine  $M_d$  that we are building.

**The notion of  $i$ -set.** We say that a set  $E \subseteq \mathbb{N} \times \mathbb{N}$  is an  $i$ -set at step  $s$  if all  $(n, l) \in E$  have been enumerated in  $L_d$  at one step  $t \leq s$  and if we have  $i$  distinct approximations  $A_r \upharpoonright_n$  with  $t \leq r \leq s$  which are each such that  $K(A_r \upharpoonright_n) \leq l + c + c_d$ . The *weight* of an  $i$ -set  $E$  is given by  $\text{weight}(E) = \sum_{(n,l) \in E} 2^{-l}$ . We further suppose that for two distinct elements  $(n_1, l_1), (n_1, l_2) \in E$  we have  $n_1 \neq n_2$ .

We set  $k = 2^{c+c_d+1}$ . Each of our processes will create an  $i$ -set for  $i \leq k$ . The starting point of the future **gold run** argument is the following lemma.

**Lemma 2.2.** If  $E$  is a  $k$ -set, then  $\text{weight}(E) < \frac{1}{2}$ . ★

PROOF. For all  $(n, l)$  in  $E$  we have  $k$  distinct strings  $\rho_1, \dots, \rho_k$  of length  $n$  such that  $K(\rho_i) \leq l + c + c_d$  for  $i \leq k$  and therefore such that  $2^{-l-c-c_d} \leq 2^{-K(\rho_i)}$  for  $i \leq k$ . In particular  $k \times 2^{-l-c-c_d} \leq \sum_{i \leq k} 2^{-K(\rho_i)}$ . Moreover the elements of  $E$  are all pairwise distinct on their first coordinate. We therefore have

$$k \times \sum_{(n,l) \in E} 2^{-l-c-c_d} \leq \sum_{\rho} 2^{-K(\rho)} < 1$$

So  $k \times 2^{-c-c_d} \times \text{weight}(E) < 1$ . As  $k = 2^{c+c_d+1}$  we therefore have  $\text{weight}(E) < \frac{1}{2}$ . ■

We will shortly describe  $k$  separate programs  $P_1, \dots, P_k$ , where each instance of  $P_i$  will be a process responsible for producing an  $i$ -set.

**Processes tree.** The algorithm will consist of a dynamic tree of processes, where each node of height  $i$  will be an instance of the program  $P_{k-i}$ . In this tree, the child processes of a node correspond to the processes it has called. Each node remains in the tree as long as its corresponding process is running, and disappears with all its children when the process stops, or is stopped by one of its ascending processes.

The algorithm starts by executing a single instance of  $P_k$ , which will be the root of our tree. This instance of  $P_k$  will call infinitely many child processes, each being an instance of  $P_{k-1}$ , each of its child processes will in turn call instances of  $P_{k-2}$  and so on. The leaves of this tree will therefore be instances of  $P_1$ .

Each instance of  $P_i$  will be responsible for enumerating its own  $i$ -set, with the help of the child processes it calls. So each instance of  $P_1$  — the leaves of the tree — will begin the enumeration of a 1-set. Each instance of  $P_2$  will begin the enumeration of a 2-set, with the help of the 1-sets enumerated by its child processes, then each instance of  $P_3$  will begin the enumeration of a 3-set, with the help of the 2-sets enumerated by its child processes, and so on.

**Process parameters.** Each program  $P_i$  takes three parameters as input.

1. The *threshold* parameter  $p$ : This is a rational number smaller than 1, which corresponds to a *threshold* to be reached by  $P_i$ , that is,  $P_i$  will attempt to enumerate an  $i$ -set  $F_i$  of weight  $p$ .

2. The *step* parameter  $\delta$ : This is a rational number smaller than  $p$  and corresponding to “the speed” at which  $P_i$  will try to reach its threshold  $p$ , i.e., it will enumerate in its  $i$ -set pairs of the form  $(n, -\log_2(\delta))$ . For this reason we therefore ask for  $p$  to be a multiple of  $\delta$ .
3. The *length* parameter  $w$ : The last parameter of  $P_i$  corresponds to some prefix length of some approximation of  $A$ ; this length having to gradually increase as recursive calls are made. This will be clarified later.

Let us briefly explain the idea behind the threshold parameter  $p$ : when the weight of a process’  $i$ -set reaches its threshold  $p$ , this process stops and recursively stops all its child processes: the elements of its  $i$ -set are now ready to be promoted into elements of the  $(i+1)$ -set of its parent process. This will happen if the approximation of some prefix of  $A$  — of a specific length, this will be detailed later — ever changes again.

Let’s briefly explain the idea behind the step parameter  $\delta$ : by definition before enumerating  $(n, l)$  into an  $i$ -set, we must first enumerate it in  $L_d$ . The concern is that the weight  $2^{-l}$  corresponding to  $(n, l)$  is likely to be “spent for nothing”. This happens if after the enumeration of  $(n, l)$  in  $L_d$ , the corresponding approximation  $A_s \upharpoonright_n$  of  $A \upharpoonright_n$  changes before we have  $K(A_s \upharpoonright_n) \leq l + c + c_d$ . An instance of  $P_i$  will therefore fill its  $i$ -set little by little with a step of  $\delta$ , where  $\delta$  corresponds to a potential loss. The set of all the parameters  $\delta$  to choose must therefore be such that the total weight that can be lost — that is to say the sum of the parameters  $\delta$  — is sufficiently low.

Finally, let’s explain the idea behind the length parameter  $w$ : this is the length of the current prefix of  $A$  which is awaiting validation by the parent process. It will be necessary for the current process to validate only prefixes larger than  $w$ , and so on, up to leaf processes.

**The golden run.** The heart of the proof lies in the idea of the **golden run**. Each instance of  $P_i$  will try to create its own machine  $M$  — with a bounded requests set  $L$  — such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . The  $i$ -set of an instance of  $P_i$  is not only used as a step in creating the  $(i+1)$ -set of its parent process. It also matches the weight which is “lost” for this instance when attempting to create a machine  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ .

We have seen that the weight of the tree’s root process’  $k$ -set always remains below  $1/2$ . So what is lost by the root process is limited, and this is the whole point: the lost weight is what has been spent trying to do as well as the universal machine  $U^A$ , but on bad prefixes of  $A$ . If this wasted quantity is always smaller than  $1/2$ , then “half” of what we can spend remains to actually do as well as  $U^A$  — up to some constant of course. However,

we cannot conclude that the root process will succeed in enumerating the desired machine  $M$ : it could be that the validation of a certain prefixes of  $A$  never comes: for that it suffices that a child process itself never reaches its threshold. However if this happens, this child process will itself be a candidate for creating a machine  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ . This will work if all of its validation requests succeed — as long as  $A$  does not change during validation of course — that is to say if all of its child processes are either stopped before validation or they themselves reach their threshold. If one of them never reaches its threshold we then descend further down the tree until we reach  $P_2$  instances, for which we will show that the validations always succeed.

Thus, a process which is never stopped by one of its ascending nodes, which never reaches its threshold and for which all its validation requests are successful, will be a **gold run**, which will result in the creation of a machine  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ .

**The special stages of computation.** In order to simplify the algorithm description, we will work with special computation steps, i.e., steps for which the set  $A$  *seems* K-trivial. Formally  $s_0 = 1$  and  $s_{n+1}$  is the smallest computation time strictly greater than  $s_n$  and such that for all  $m \leq s_n$  we have  $K(A_{s_{n+1}} \upharpoonright_m)[s_{n+1}] \leq K(m)[s_{n+1}] + c$ . We will denote *computation steps* (in italics) these special computation steps  $s_0, s_1, s_2, \dots$ , and the algorithm will restrict itself to these special steps.

The reason for these *computation steps* is the following: once we have  $K(A_s \upharpoonright_n)[s] \leq l + c + c_d$  for some  $l$ , if  $A_{s+1} \upharpoonright_n \neq A_s \upharpoonright_n$  we want to put  $(n, l)$  in a 2-set, but this is only possible if we also have  $K(A_{s+1} \upharpoonright_n)[s+1] \leq l + c + c_d$ . For this reason, instead of looking for a computation step  $s$  such that  $K(A_s \upharpoonright_n)[s] \leq l + c + c_d$ , we will instead look for a *step*  $s > n$  such that  $K(n)[s] \leq l + c_d$ . In this way, as long as we are only working along our *computation steps*, we will always have  $K(A_t \upharpoonright_n)[t] \leq K(n)[t] + c \leq l + c + c_d$  for any *step*  $t \geq s$ .

**Description of the algorithm.** We recall here the different notations used for the algorithm:  $A$  is our  $\Delta_2^0$  set which is also K-trivial with the constant  $c$ ,  $U^A$  is the universal machine which defines our complexity  $K^A$  and for which we try to construct  $M$  such that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ .

The global algorithm creates a tree of processes by first calling an instance of the program  $P_k$  described below, with the threshold parameter  $p = \frac{3}{4}$ , the step parameter  $\delta = \frac{1}{4} \times 2^{-1}$  and the length parameter  $w = 0$ . The instance of  $P_k$  will in turn call instances of the program  $P_{k-1}$ , which in turn will call instances of the program  $P_{k-2}$ , and so on.

Each called process participates in the creation of a shared bounded requests set  $L_d$ , which is itself used to build a machine  $M_d$ . Using the fixed point theorem, the global algorithm uses a constant  $c_d$  such that  $K(\sigma) \leq K_{M_d}(\sigma) + c_d$  and for which we have  $k = 2^{c+c_d+1}$ .

We end with a last consideration on the stages of computation and the way in which the parallelism between the various processes is carried out. In practice, a “global scheduler” simulates parallelism by distributing the different *stages* of computation between the different processes to be executed. To simplify the presentation, this arrangement is implicit, and the flow of each process is described *step by step*. We just give the following constraint for the global scheduler: a *step*  $s$  of a parent process is executed before the same *step*  $s$  of its child processes.

Finally, it will be useful, for the creation of the bounded requests set  $L_d$  which will be shared between all the processes, to have access to a unique integer per process and computation time. For this, we will use the computation time of the global scheduler and for a *computation step*  $s$  within a process, we will denote by  $s^*$  the corresponding global computation step. Without further ado here is the algorithm:

---

Program  $P_1$

---

**Input :** The threshold  $p$ , the step  $\delta$  and the length  $w$

**Output:** A 1-set  $F_1$

---

**For** each *step*  $s > w$  **do**

Let  $n = s^*$  (which we also suppose to be greater than  $w$ ).

We enumerate  $(n, -\log_2(\delta))$  into  $L_d$ .

We wait for a *step*  $t > n$  such that  $K(n)[t] \leq -\log_2(\delta) + c_d$

We enumerate  $(n, -\log_2(\delta))$  into  $F_1$

**If**  $\text{weight}(F_1) \geq p$  **then**  
     | We stop the program.

**End**

**End**

---

---

Program  $P_i$  for  $1 < i \leq k$

---

**Input** : The threshold  $p$ , the step  $\delta$  and the length  $w$

**Output**: An  $i$ -set  $F_i$  and a bounded requests set  $L$

---

**For** each *step*  $s$  **do**

**For** each sub-step  $\langle \tau, \sigma \rangle \leq s$  **do**

**If**  $\langle \tau, \sigma \rangle$  is marked as available **then**

**If**  $\exists \rho \preceq A_s$  such that  $U^\rho(\tau)[s] = \sigma$  **then**

                Let  $\rho_{\tau, \sigma}$  be the smallest prefix of  $A_s$  of length greater than  $\max(w, |\rho|)$ .

                Let  $w_{\tau, \sigma}$  be the length of  $\rho_{\tau, \sigma}$ .

                Let  $\delta'$  be the greatest rational less than  $\frac{1}{4} \times 2^{-s^*}$  such that  $2^{-|\tau|} \times \delta$  is a multiple of  $\delta'$ .

                We then call the program  $P_{i-1}(2^{-|\tau|} \times \delta, \delta', w_{\tau, \sigma})$ . We denote by  $P_{i-1, \tau, \sigma}$  the corresponding process and  $F_{i-1, \tau, \sigma}$  its  $(i-1)$ -set.

                We mark  $\langle \tau, \sigma \rangle$  as unavailable.

**End**

**End**

**If**  $\langle \tau, \sigma \rangle$  is marked as unavailable after calling  $P_{i-1, \tau, \sigma}$  **then**

**If** the process  $P_{i-1, \tau, \sigma}$  has ended **then**

            We enumerate  $\left( \sigma, |\tau| - \log_2 \left( \frac{\delta}{p + \delta} \right) \right)$  in  $L$ .

**End**

**If**  $A_s \upharpoonright_{w_{\tau, \sigma}}$  is different from  $\rho_{\tau, \sigma}$  **then**

        We mark  $\langle \tau, \sigma \rangle$  as available.

        If the process  $P_{i-1, \tau, \sigma}$  is not finished, we cancel it as well as all its sub-processes (while keeping  $F_{i-1, \tau, \sigma}$ )

        We enumerate the elements of  $F_{i-1, \tau, \sigma}$  into  $F_i$ .

**If**  $\text{weight}(F_i) \geq p$  **then**

            We stop the program and all its sub-processes.

**End**

**End**

**End**

**End**

---

Now let's move on to the formal verification.

**Let us show that the output  $F_i$  of each instance of  $P_i$  is an  $i$ -set.** Let's start with the following fact.

- (1) Let  $(n_1, l_1), (n_2, l_2)$  be enumerated in any 1-set during the algorithm at different stages or by different processes. Then,  $n_1 \neq n_2$ .

This comes from the fact that such pairs  $(n_1, l_1), (n_2, l_2)$  are enumerated in two distinct global steps  $s_1^* \neq s_2^*$  which will be by construction such that  $n_1 = s_1^*$  and  $n_2 = s_2^*$ . Let us show that each set  $F_1$  enumerated by an instance of  $P_1$  is a 1-set at each *step* of computation. First of all (1) makes it possible to satisfy one of the conditions to be a 1-set, that is to say to have their elements pairwise distinct on their first coordinate.

For the other condition, if at a *step*  $s$  we enumerate  $(n, l)$  in a 1-set, this means that  $K(n)[t] \leq l + c_d$ . Since  $t$  is a *computation step* then  $K(A_t \upharpoonright_n)[t] \leq K(n)[t] + c \leq l + c + c_d$  and we therefore satisfy the other condition to be a 1-set.

Suppose now that each set  $F_{i,\tau,\sigma}$  associated with an instance  $P_{i,\tau,\sigma}$  of  $P_i$  is an  $i$ -set at each *computation step*. Suppose that in a step  $s$  we promote an element  $(n, l)$  of an  $i$ -set  $F_{i,\tau,\sigma}$  into the  $(i+1)$ -set  $F_{i+1}$  of a  $P_{i+1}$  instance. According to (1), all the elements of  $F_{i+1}$  after this enumeration remain pairwise distinct on their first coordinate. By construction we also have:

- A *step*  $t < s$  and an integer  $w_{\tau,\sigma} < n$  such that  $A_t \upharpoonright_{w_{\tau,\sigma}} \neq A_s \upharpoonright_{w_{\tau,\sigma}}$ .
- A *step*  $r$  with  $t < r < s$  such that  $K(n)[r] \leq l + c_d$  (and therefore such that  $(n, l)$  enters a 1-set).
- A *step* between  $r$  and  $s$  such that the pair  $(n, l)$  enters the  $i$ -set  $F_{i,\tau,\sigma}$ .

We have  $K(A_s \upharpoonright_n)[s] \leq K(n)[s] + c \leq K(n)[r] + c \leq l + c + c_d$  because  $s$  is a *computation step* greater than  $r$ . So  $A_s \upharpoonright_n$  is a good candidate for making  $(n, l)$  part of a  $(i+1)$ -set. However, we must check that  $A_s \upharpoonright_n$  is different from all the other strings  $\rho_1, \dots, \rho_i$  of length  $n$  which make  $(n, l)$  the element of the  $i$ -set  $F_{i,\tau,\sigma}$ . Let us not forget the fact that a  $\Delta_2^0$  approximation can return to a value already encountered in the past. It suffices to show that  $A_t$  shares its first  $w_{\tau,\sigma}$  bits with each  $\rho_j$  for  $j \leq i$ . If this is the case, since we have  $A_t \upharpoonright_{w_{\tau,\sigma}} \neq A_s \upharpoonright_{w_{\tau,\sigma}}$  then  $A_s \upharpoonright_n$  will be distinct from each  $\rho_j$ . It suffices to see that  $s$  is, within our instance of  $P_{i+1}$ , the smallest *step* such that  $A_t \upharpoonright_{w_{\tau,\sigma}} \neq A_s \upharpoonright_{w_{\tau,\sigma}}$  and to remember that the *steps* of the parent processes are processed before the steps of the *child* processes.

**Let us show that  $L_d$  is indeed a bounded requests set.** The total weight that is enumerated into  $L_d$  by a  $P_{1,\tau,\sigma}$  process executed with

parameter  $(p, \delta, w)$  is bounded by the weight of its 1-set plus  $\delta$ . Indeed by construction for each  $(n, -\log_2(\delta))$  enumerated in  $L_d$ , we also enumerate  $(n, -\log_2(\delta))$  in the 1-set unless the process is stopped by one of its ascending nodes, which happens only once. The weight we lose is then  $\delta$ .

Given an instance of  $P_{i+1}$  executed with parameter  $(p, \delta, w)$ , let  $C$  be the set of elements that are enumerated in the  $i$ -set of a child process of this instance of  $P_{i+1}$ , but which are not enumerated in its  $(i+1)$ -set. Note that some elements are in  $C$  if the instance of  $P_{i+1}$  is canceled by one of its ascending nodes. Let us show that the weight of  $C$  is bounded by  $\delta$ . When the instance of  $P_{i+1}$  is canceled, let  $(P_{i,\tau_j,\sigma_j})_{j \in \mathbb{N}}$  be the set of child processes that are not terminated. Note that the  $\tau_j$  are necessarily in the domain of  $U^\rho$  for pairwise compatible strings  $\rho$ : these strings  $\rho$  are prefixes of a certain approximation of  $A_s$ , and when this approximation changes the corresponding child process is canceled by  $P_{i+1}$  itself. Let  $\rho$  be the longest of all these strings. Since the weight of the  $i$ -set corresponding to  $P_{i,\tau_j,\sigma_j}$  is bounded by  $2^{-\tau_j} \delta$ , then the total weight is bounded by  $\delta \sum_{U^\rho(\tau) \downarrow} 2^{-|\tau|} \leq \delta$ .

By iterating this idea, we get that the sum of the total weights of all the 1-sets is bounded by the weight of the  $k$ -set of the root process of the tree, plus the sum of all the step parameters  $\delta$ . These parameters are therefore chosen so that their sum is less than  $1/4$ . Since the weight of the root process'  $k$ -set is always less than  $1/2$ , the total weight of what is listed in  $L_d$  is less than 1.

**End of the proof: the golden run.** Let us now show that there is a process  $P_{i,\tau,\sigma}$  called with parameter  $(p, \delta, w)$ , which is never stopped by one of its ascending nodes, which never reaches its threshold  $p$ , and such that every process it calls ends unless stopped by  $P_{i,\tau,\sigma}$ .

We know the root process cannot be stopped by another process and never reaches its threshold, by Lemma 2.2. If each of its child processes ends or is canceled, then the root process is the golden run. Otherwise at least one of the root's child processes is never canceled and never reaches its threshold. Either this is the golden run, or one of his children is never stopped and never reaches his threshold. If the induction continues until a process of the form  $P_{2,\tau,\sigma}$ , then this process is necessarily the golden run, because it is never canceled, never reaches its threshold, and all the leaf processes of the tree necessarily reach their threshold unless they are stopped by an ascending process.

Let now be a golden run given by an instance of  $P_i$ , called with parameter  $(p, \delta, w)$ . Let  $F_i$  be its  $i$ -set and  $L$ , the set enumerated by this process.

**Let us show that  $L$  is indeed a bounded requests set.** For each pair

$$\left( \sigma, |\tau| - \log_2 \left( \frac{\delta}{p + \delta} \right) \right)$$

enumerated in  $L$  we have  $U^\rho(\tau) = \sigma$  for  $\rho \prec A_s$  for some  $s$ . This pair corresponds to a child process  $P_{i-1,\tau,\sigma}$ . Two cases arise:

*Case 1:*  $\rho$  is a prefix of  $A$ . If we consider the sum of the weights of the pairs associated with the prefixes of  $A$ , we then obtain

$$\sum_{U^A(\tau) \downarrow} 2^{-|\tau| + \log_2 \left( \frac{\delta}{p + \delta} \right)} = \frac{\delta}{p + \delta} \sum_{U^A(\tau) \downarrow} 2^{-|\tau|} \leq \frac{\delta}{p + \delta}$$

*Case 2:*  $\rho$  is not a prefix of  $A$ . Then, during a future *step*  $s$ ,  $A_s \upharpoonright_{|\rho|} \neq \rho$ , so the  $(i-1)$ -set created by the child process  $P_{i-1,\tau,\sigma}$  will be integrated into the  $i$ -set  $F_i$  created by the **golden run**  $P_i$ . The sum of the weight of the pairs associated with  $\rho$  which are not prefixes of  $A$  is bounded by

$$\begin{aligned} \sum_{P_{i-1,\tau,\sigma}} 2^{-|\tau| + \log_2 \left( \frac{\delta}{p + \delta} \right)} &= \frac{\delta}{p + \delta} \frac{\sum_{P_{i-1,\tau,\sigma}} 2^{-|\tau| + \log_2(\delta)} }{\delta} \\ &= \frac{\delta}{p + \delta} \frac{\sum_{P_{i-1,\tau,\sigma}} \text{weight}(F_{i-1,\tau,\sigma})}{\delta} \\ &= \frac{\delta}{p + \delta} \times \frac{\text{weight}(F_i)}{\delta} \end{aligned}$$

The total weight of  $L$  will therefore be bounded by

$$\frac{\delta}{p + \delta} + \frac{\delta}{p + \delta} \times \frac{\text{weight}(F_i)}{\delta}$$

Since  $F_i$  never reaches its threshold which is  $p$  we therefore have:

$$\frac{\delta}{p + \delta} \left( 1 + \frac{\text{weight}(F_i)}{\delta} \right) \leq \frac{\delta}{p + \delta} (1 + p/\delta) \leq 1.$$

So  $L$  is indeed a bounded requests set. Let  $M$  be the corresponding machine.

**Let us show that  $K_M(\sigma) \leq^+ K^A(\sigma)$  for any string  $\sigma$ .** Now if  $U^\rho(\tau)[s] = \sigma$  such that  $A_s \upharpoonright_{|\rho|}$  will never change again, then the  $P_{i-1,\tau,\sigma}$  process will never be canceled. It will therefore reach its threshold and we will then enumerate  $(\sigma, |\tau| - \log_2(\delta/(p + \delta)))$  into  $L$ . So  $K_M(\sigma) \leq K^A(\sigma) - \log_2(\delta/(p + \delta))$  for any string  $\sigma$ . This concludes the proof.

### 2.1. Consequences of the theorem and its proof

We have now completed the **golden run** proof. Let us see immediately an interesting consequence.

**Theorem 2.3** (Nies [165])

*The class of K-trivials forms a Turing ideal:*

(1) *If  $A$  is K-trivial then every  $B \leq_T A$  is K-trivial*

(2) *If  $A_0, A_1$  are K-trivial then  $A_0 \oplus A_1$  is K-trivial.*

PROOF. We have seen with Exercize 16-4.7 that K-trivial sets are closed under Turing join. It is clear that the low-for-K sets are downward-closed in the Turing degrees. ■

Note that in the previous theorem, neither (1) is obvious for K-trivial sets, nor (2) for low-for-K sets. The fact that the two classes coincide is remarkable and gives us this nice result.

Let us remember for a moment the proof of Theorem 16-4.5 where we construct a non-computable c.e. set  $A$  such that  $K(A \upharpoonright_n) \leq^+ K(n)$  for all  $n$ . During the approximation we assign the weight  $2^{-K(n)[s]}$  to each prefix  $A_s \upharpoonright_n$  of the current approximation of  $A$  for  $n \leq s$ . This assignment corresponds to what we put in a bounded requests set to produce  $A_s \upharpoonright_n$ , that is to say the length at step  $s$  of the smallest string producing  $n$ . If at step  $s + 1$  an integer  $x_s < s$  is enumerated into  $A$  then we have lost everything we had assigned to  $A_s \upharpoonright_n$  for  $x_s < n \leq s$ . The construction then guarantees that the total sum of what is lost is finite.

To build a non-computable and low-for-K c.e. set  $A$ , we can proceed in the same way, but where we assign a weight of  $\Omega_s^{A_s \upharpoonright_n} - \Omega_s^{A_s \upharpoonright_{n-1}}$  to each prefix  $A_s \upharpoonright_n$  of the current approximation of  $A$ . Here  $\Omega_s^{A_s \upharpoonright_n} - \Omega_s^{A_s \upharpoonright_{n-1}}$  is the sum of  $2^{-|\tau|}$  such that  $U^{A_s}(\tau)[s] \downarrow$  with a use of  $A_s$  of exactly  $n$ . As for the construction of a c.e. K-trivial, if  $x_s < s$  is enumerated in  $A$  at step  $s + 1$ , we then see that we lose for all  $x_s < n \leq s$  the quantity  $\Omega_s^{A_s \upharpoonright_n} - \Omega_s^{A_s \upharpoonright_{n-1}}$ , that is, anything that has been assigned to the wrong prefixes of  $A$ . We can abstract this reasoning and consider what we call a *cost function*.

**Definition 2.4.** A *cost function* is given by  $c : \mathbb{N} \rightarrow \mathbb{Q}$ , a  $\Delta_2^0$  function such that  $\sum_n c(n) < 1$  and such that  $\sum_n c_s(n) < 1$  for any approximation  $s$ . The approximation  $(c_s)_{s \in \mathbb{N}}$  of a cost function can possibly depend on a  $\Delta_2^0$  approximation  $(A_s)_{s \in \mathbb{N}}$  of a set  $A$  in which case the function will be called *adaptive*. ◇

The cost function to build a K-trivial set is given by  $c(n) = 2^{-K(n)}$ . The one used to build a low-for-K set is adaptive and is given by the approximation  $c_s(n) = \Omega_s^{A_s \upharpoonright n} - \Omega_s^{A_s \upharpoonright n-1}$ . In both cases we can construct a non-computable c.e. set such that what we lose is given by the cost function, therefore is finite. This gives us the following definition.

**Definition 2.5.** Let  $(A_s)_{s \in \mathbb{N}}$  be a  $\Delta_2^0$  approximation of a set  $A$ . Let  $S = \{s \in \mathbb{N} : A_s \neq A_{s+1}\}$  and for  $s \in S$  let  $x_s < s$  be the smallest integer such that  $A_{s+1} \upharpoonright_{x_s} \neq A_s \upharpoonright_{x_s}$ . The approximation of  $A$  satisfies a cost function  $c$  —possibly adaptive— if  $\sum_{s \in S} \sum_{x_s < n \leq s} c_s(n)$  is finite.  $\diamond$

The golden run proof can be adapted to show that we can accelerate the  $\Delta_2^0$  approximation of any K-trivial set, so as to show that it satisfies any cost function fixed in advance. This gives us the following theorem.

**Theorem 2.6 (Nies [165])**

Let  $c$  be a possibly adaptive cost function. Any K-trivial set  $A$  admits a  $\Delta_2^0$  approximation  $(A_s)_{s \in \mathbb{N}}$  which satisfies  $c$ .

We can then use this to show several interesting properties, for example the fact that every K-trivial admits an approximation with few changes:

**Theorem 2.7 (Nies [165])**

Any K-trivial set admits a  $\Delta_2^0$  approximation such that each prefix of length  $n$  changes at most  $n^2 \times d$  times for some constant  $d$ .

PROOF. By Proposition 16-2.5, there exists  $d$  such that  $K(n) \leq 2 \log_2(n) + d$  for all  $n$ . We can consider without loss of generality that for all  $s$  we have  $K(n)[s] \leq 2 \log_2(n) + d$ . It is then sufficient to use the cost function given by  $c_s(n) = 2^{-K(n)[s]}$ . Let  $(A_s)_{s \in \mathbb{N}}$  be an approximation of  $A$  which satisfies this cost function and let  $e$  be the total sum of the cost that is lost. Every time a prefix of  $A$  of length  $n$  changes at a step  $s$  it costs at least  $2^{-K(n)[s]} \geq 2^{-2 \log_2(n) - d} = n^{-2} \times 2^{-d}$ . So this cannot happen more than  $n^2 \times 2^d \times e$  times without raising the cost above  $e$ . ■

It is also possible to use cost functions technique to show that any K-trivial set is bounded in the Turing degrees by a c.e. K-trivial. For this, we define the c.e. set  $C$  such that  $C$  enumerates  $\langle n, i \rangle$  at step  $s$  if the prefix of length  $n$  of  $A$  changes for the  $i$ -th time at step  $s$ . We easily verify that  $C \geq_T A$ . Indeed, to determine the initial segment of  $A$  of length  $n$ , it suffices to find the largest  $i$  such that  $\langle n, i \rangle \in C$ , and to perform the approximation of  $A$  until  $A_s \upharpoonright_n$  has changed  $i$  times. We then show that if

an approximation of  $A$  satisfies the cost function given by  $c(n) = 2^{-K(n)}$ , then the approximation of  $C$  must also satisfy this cost function, which we can use to show that  $C$  is also K-trivial. We will see an alternative proof of this fact with Theorem 4.4.

### 3. Characterization of the c.e. K-trivials

We continue here our study of the K-trivials with the presentation of a question which remained open for some time, and whose study led to many developments, which among other things allowed to give another proof of the fact that any K-trivial is low-for-K.

#### 3.1. The question

The question arised from the study of bases for randomness, after the discovery of the following proposition and its corollary.

**Proposition 3.1.** Let  $A$  be a c.e. set and let  $Z$  be a set which is MLR but not MLR( $A$ ). Then,  $Z \oplus A \geq_T \emptyset'$ . ★

PROOF. Let  $\bigcap_n \mathcal{U}_n^A$  be an  $A$ -Martin-Löf test such that  $Z \in \bigcap_n \mathcal{U}_n^A$ . Note that we can assume from Theorem 19-2.3 that  $\bigcap_n \mathcal{U}_n^X$  is a Martin-Löf test for any set  $X$ . We now describe  $\Sigma_1^0$  classes  $(\mathcal{V}_n)_{n \in \mathbb{N}}$  used to create a Solovay test. If  $n$  is enumerated in  $\emptyset'$  at step  $t$ , then we enumerate  $\mathcal{U}_n^{A_t}[t]$  in  $\mathcal{V}_n$ . If  $n$  is never enumerated in  $\emptyset'$  then  $\mathcal{V}_n$  is left empty. Since  $Z$  is MLR then there exists  $n$  such that  $Z \notin \bigcup_{m \geq n} \mathcal{V}_m$ . We can now decide if an integer  $m \geq n$  belongs to  $\emptyset'$  using  $Z \oplus A$  as follows: let  $t$  be the smallest computation time such that  $Z \in \mathcal{U}_m^A[t]$ . Let  $\rho$  be the prefix of  $A$  such that  $\mathcal{U}_m^\rho[t] = \mathcal{U}_m^A[t]$ . We then seek the smallest computation time  $t'$  such that  $A_{t'} \upharpoonright_{|\rho|} = A \upharpoonright_{|\rho|}$ . We then have  $m \in \emptyset'$  iff  $m \in \emptyset'[\max(t, t')]$ . Indeed if  $m$  enters into  $\emptyset'$  at a computation time  $s > \max(t, t')$  then we will still have  $A_s \upharpoonright_{|\rho|} = A \upharpoonright_{|\rho|}$  because  $A$  is c.e. and therefore  $Z \in \mathcal{U}_m^\rho[s] \subseteq \mathcal{U}_m^{A_s}[s] = \mathcal{V}_m$  which contradicts  $Z \notin \mathcal{V}_m$ . ■

#### Corollary 3.2 (Hirschfeldt, Nies and Stephan [89])

*Let  $Z$  be an incomplete MLR set and  $A$  a c.e. set such that  $Z \geq_T A$ . Then,  $A$  is K-trivial.*

PROOF. Since  $Z \geq_T A$  and  $Z$  is incomplete then  $Z \oplus A \not\geq_T \emptyset'$ . So according to Proposition 3.1,  $Z$  is MLR( $A$ ) and  $A$  is then a basis for randomness. According to Theorem 1.7,  $A$  is therefore low-for-K and therefore K-trivial. ■

What about the converse of corollary 3.2 ? Are the c.e. K-trivials necessarily computable by an incomplete MLR set ? This question has been the subject of numerous works and developments [13] [17] [18] [19] [42] resulting in a positive answer, allowing also, together with other results, to give an alternative proof of the fact that K-trivial sets are all low-for-K.

**Exercize 3.3. (★)** Show that Corollary 3.2 cannot be extended to  $\Delta_2^0$  sets.  
 $\diamond$

### 3.2. Oberwolfach Randomness

Oberwolfach is a small town of a few thousand inhabitants, lost in the heart of the black forest in the Baden-Württemberg state of Germany. It was here that a mathematical research center emerged during World War II and should become one of the most famous and important in the world. All year round, mathematicians from all over the world meet there for conferences. The calm of the place and its surrounding nature make it suited to concentration and mathematical work. Therefore, in addition to its conferences, the Oberwolfach research center also organizes “research in pairs” programs: a small group of 2 to 5 mathematicians meet there for a few weeks to work on a specific subject. It is with one of these programs that in 2012, Laurent Bienvenu (researcher at the CNRS), Noam Greenberg (professor at the University of Wellington), Antonín Kučera (professor at the University of Prague), André Nies (professor at the Auckland University) and Dan Turetsky (postdoctoral fellow at the Kurt Gödel research center in Vienna), meet in Oberwolfach to tackle the question of corollary 3.2’s converse.

If their work will not immediately lead to a solution, it would nevertheless constitute a fundamental progress, through the concept they baptized in honor of the place hosting them: *Oberwolfach randomness*. We have seen with Theorem 19-1.5 that if  $X$  is MLR but not weakly 2-random, then  $X$  computes a non-computable c.e. set. We have seen with Theorem 19-4.2 that if  $X$  is MLR but not difference random, then  $X$  computes the halting problem. Given a c.e. K-trivial set  $A$ , the idea is then to search for a randomness notion that would lie between difference randomness and weak 2-randomness, such that all the MLR sets which are not random for this notion would necessarily compute  $A$ .

**Definition 3.4 (BGKNT [15]).** An *Oberwolfach test* is given by a  $\Pi_2^0$  class  $\bigcap_n \mathcal{U}_n$  and a positive left-c.e. real  $r < 1$  such that  $\lambda(\mathcal{U}_n) \leq r - r_n$ , where  $(r_n)_{n \in \mathbb{N}}$  is the approximation of  $r$ . A set  $Z$  is *Oberwolfach-random* if  $Z$  does not belong to any Oberwolfach test.  $\diamond$

The convergence condition of the measure of open classes towards 0 is therefore relaxed in an Oberwolfach test compared to that in a Martin-Löf test. Indeed if  $r$  is not computable then neither is the bound  $r - r_n$ . There is however still a restriction on the convergence of the open classes towards 0, and it is in fact possible to show that this notion of randomness is strictly weaker than weak 2-randomness, for which there are no restrictions. We will see with Theorem 3.7 that any Oberwolfach random set is also difference random. The following theorem shows that Oberwolfach randomness objective is fulfilled at once for all K-trivials.

**Theorem 3.5 (BGKNT [15])**

*Let  $Z$  be a set which is MLR but not Oberwolfach random. Then,  $Z$  computes all the c.e. K-trivials.*

In order to show Theorem 3.5 we need a lemma which at first may seem cryptic, but which takes on its full meaning when placed in the context of cost functions that we have seen with Definition 2.4. The following lemma tells us that for any left-c.e. real  $r$ , any c.e. K-trivial admits a c.e. approximation which satisfies the computable cost function given by  $c(s) = r_s - r_{s-1}$ . This is of course a consequence of Theorem 2.6, but which can be shown here directly.

**Lemma 3.6 (BGKNT [15]).** Let  $A$  be a c.e. K-trivial with a constant  $d$ . Let  $r < 1$  be a positive left-c.e. real with its approximation  $(r_s)_{s \in \mathbb{N}}$ . Then, there exists an approximation  $(A_s)_{s \in \mathbb{N}}$  of  $A$  such that

$$\sum_{s \in S} r_s - r_{x_s} \text{ is finite,}$$

where  $S = \{s \in \mathbb{N} : A_s \neq A_{s+1}\}$  and where  $x_s < s$  is the smallest integer such that  $A_{s+1} \upharpoonright_{x_s} \neq A_s \upharpoonright_{x_s}$  ★

PROOF. We define the function  $f : \mathbb{N} \rightarrow \mathbb{N}$  as being  $f(s) = -\log_2(r_{s+1} - r_s)$ . We have in particular  $\sum_s 2^{-f(s)} = r < 1$ . According to Theorem 16-3.1 there exists a constant  $c$  such that  $2^{-f(s)} \leq 2^{-K(s)} 2^c$  for all  $s$ . As  $A$  is K-trivial with constant  $d$ , we can therefore speed up its approximation  $(A_s)_{s \in \mathbb{N}}$  in order to have  $K(A_s \upharpoonright_t)[s] \leq f(t) + c + d$  for all  $s$  and all  $t \leq s$ . We have then for  $s \in S$ :

$$r_s - r_{x_s} = \sum_{x_s \leq t < s} 2^{-f(s)} \leq \sum_{x_s \leq t < s} 2^{-K(A_s \upharpoonright_t)} 2^{c+d}$$

Since  $A$  is a c.e. set then if  $A_s \upharpoonright_{x_s} \neq A_{s+1} \upharpoonright_{x_s}$  we also have  $A_s \upharpoonright_{x_s} \neq A_{s'} \upharpoonright_{x_s}$  for all  $s' > s + 1$ . In particular we have  $A_s \upharpoonright_{x_s} \neq A_{s'} \upharpoonright_{x_{s'}}$  for  $s, s'$  distinct

belonging to  $S$  and therefore

$$\sum_{s \in S} r_s - r_{x_s} \leq \sum_{s \in S} \sum_{x_s \leq t < s} 2^{-K(A_s \upharpoonright t)} 2^{c+d} \leq \sum_{U(\tau) \downarrow} 2^{-|\tau|} 2^{c+d} \leq 2^{c+d} \Omega \leq 2^{c+d}.$$

This concludes the proof.  $\blacksquare$

PROOF OF THEOREM 3.5. Let  $Z$  be an MLR set captured by an Oberwolfach test  $\bigcap_n \mathcal{U}_n$ , with  $\lambda(\mathcal{U}_n) \leq r - r_n$  for a positive left-c.e. real  $r < 1$ . Let  $A$  be a c.e. K-trivial. Using Lemma 3.6 we can assume that the enumeration  $(A_s)_{s \in \mathbb{N}}$  of  $A$  is such that  $\sum_{s \in S} r_s - r_{x_s}$  is finite where  $S = \{s \in \mathbb{N} : A_s \neq A_{s+1}\}$  and where for  $s \in S$  the integer  $x_s$  is the smallest such that  $A_{s+1} \upharpoonright_{x_s} \neq A_s \upharpoonright_{x_s}$ .

We slow down the enumeration of each open class  $\mathcal{U}_n$  so as to have

$$\lambda(\mathcal{U}_n[s]) \leq \max(0, r_s - r_n)$$

for all  $s$ . We may also assume without loss of generality that  $\mathcal{U}_{n+1}[s] \subseteq \mathcal{U}_n[s]$  for every  $n, s$ . We define the functional  $\Phi$  by  $\Phi(X, n) = A_s(n)$  for all  $X \in \mathcal{U}_{n+1}[s+1] \setminus \mathcal{U}_{n+1}[s]$ . To see that we have  $\Phi(Z, n) = A(n)$ , we define the Solovay test  $\mathcal{V}_s = \mathcal{U}_{x_s}[s]$  for all  $s \in S$ . If  $s \notin S$  then  $\mathcal{V}_s$  remains empty. Note that we have  $\sum_{s \in S} \lambda(\mathcal{V}_s) = \sum_{s \in S} \lambda(\mathcal{U}_{x_s}[s]) \leq \sum_{s \in S} r_s - r_{x_s}$ . So  $\sum_{s \in S} \lambda(\mathcal{V}_s)$  is a finite quantity and  $(\mathcal{V}_s)_{s \in \mathbb{N}}$  is indeed a Solovay test.

As  $Z \in \bigcap_n \mathcal{U}_n$  then the functional is total on the oracle  $Z$ . Suppose that  $\Phi(Z, n) \neq A(n)$  for a fixed integer  $n$ . Let  $s$  be such that  $Z \in \mathcal{U}_{n+1}[s+1] - \mathcal{U}_{n+1}[s]$ . Then, there must exist  $t > s$  such that  $A_t(n) \neq A_{t+1}(n)$ . Let  $x_t \leq n+1$  be the smallest such that  $A_t \upharpoonright_{x_t} \neq A_{t+1} \upharpoonright_{x_t}$ . We have  $\mathcal{V}_t = \mathcal{U}_{x_t}[t] \supseteq \mathcal{U}_{n+1}[s+1]$  and therefore  $Z \in \mathcal{V}_t$ . Since  $Z$  is MLR this cannot happen infinitely often. So  $\Phi(Z, n) = A(n)$  for  $n$  large enough.  $\blacksquare$

The protagonists of Oberwolfach Randomness have also shown that their notion is the best possible for the given objective: it is possible to construct a K-trivial set such that any set computing it is captured by an Oberwolfach test. So no Oberwolfach random can compute all the K-trivials, and there exist in particular K-trivials more sophisticated than others, called “smart K-trivials”, such that any random computing them can compute all K-trivials.

### 3.3. Question resolution

We are now close to solving the question of whether any c.e. K-trivial is computed by an incomplete random. It suffices to build a set which passes all the difference tests — in order to be incomplete — but which is not Oberwolfach random. However, such a construction turns out to be very

complex, and could not be completed directly. The solution will come instead of an unexpected detour by Lebesgue's density theorem. Let us remember that an MLR set is incomplete iff it is positive density point in any  $\Pi_1^0$  class containing it. Bienvenu, Greenberg, Kučera, Nies and Turetsky also showed during their stay in Oberwolfach that their randomness notion is sufficient to satisfy Lebesgue's density theorem for any  $\Pi_1^0$  class:

**Theorem 3.7 (BGKNT [15])**

*Let  $Z$  be a Oberwolfach random. Then,  $\rho(\mathcal{P} \mid Z) = 1$  for any  $\Pi_1^0$  class  $\mathcal{P}$  containing  $Z$ .*

PROOF. Let  $\mathcal{P}$  be a  $\Pi_1^0$  class containing  $Z$ . Suppose  $\rho(\mathcal{P} \mid Z) < 1$  by contradiction. Let  $\alpha < 1$  be such that  $\rho(\mathcal{P} \mid Z) < \alpha$ . Let  $T$  be the computable tree whose infinite paths are the elements of  $\mathcal{P}$ . We define

$$U_n = \{\sigma \in T : |\sigma| \geq n \text{ and } \lambda(\mathcal{P} \mid [\sigma]) < \alpha\}.$$

Each open class  $\mathcal{U}_n = [U_n]$  is  $\Sigma_1^0$  and it is clear that  $Z \in \bigcap_n \mathcal{U}_n$ . We claim that  $\bigcap_n \mathcal{U}_n$  is an Oberwolfach test. For that, we need to find a left-c.e. real  $r$  such that  $\lambda(\mathcal{U}_n) \leq r - r_n$ .

Let  $\mathcal{P}[n]$  be the approximation of  $\mathcal{P}$  given by the clopen consisting of strings  $\sigma \in T$  of length  $n$ . It is clear that we have  $\mathcal{U}_n \subseteq \mathcal{P}[n]$ . Now let's look for a bound to the measure of  $\mathcal{U}_n \subseteq \mathcal{P}[n]$ . We have

$$\lambda(\mathcal{P} \setminus \mathcal{U}_n) \leq \lambda(\mathcal{P}[n] \setminus \mathcal{U}_n) = \lambda(\mathcal{P}[n]) - \lambda(\mathcal{U}_n).$$

Moreover the elements of  $\mathcal{P}$  are either in  $\mathcal{U}_n$  or in  $\mathcal{P} \setminus \mathcal{U}_n$ . So  $\lambda(\mathcal{P}) = \lambda(\mathcal{P} \cap \mathcal{U}_n) + \lambda(\mathcal{P} \setminus \mathcal{U}_n)$ . Moreover if  $\sigma \in U_n$  then  $\lambda(\mathcal{P} \mid [\sigma]) < \alpha$ . We therefore have  $\lambda(\mathcal{P} \cap \mathcal{U}_n) \leq \alpha \lambda(\mathcal{U}_n)$ . This gives

$$\lambda(\mathcal{P}) = \lambda(\mathcal{P} \cap \mathcal{U}_n) + \lambda(\mathcal{P} \setminus \mathcal{U}_n) \leq \alpha \lambda(\mathcal{U}_n) + \lambda(\mathcal{P}[n]) - \lambda(\mathcal{U}_n).$$

We then have  $\lambda(\mathcal{P}) - \lambda(\mathcal{P}[n]) \leq (\alpha - 1)\lambda(\mathcal{U}_n)$  and therefore

$$(1 - \alpha)\lambda(\mathcal{U}_n) \leq \lambda(\mathcal{P}[n]) - \lambda(\mathcal{P}) = \lambda(\mathcal{P}^c) - \lambda(\mathcal{P}[n]^c),$$

where  $\mathcal{P}^c$  and  $\lambda(\mathcal{P}[n]^c)$  denote respectively the complements of the sets  $\mathcal{P}$  and  $\mathcal{P}[n]$ . Therefore

$$\lambda(\mathcal{U}_n) \leq \frac{1}{1 - \alpha} (\lambda(\mathcal{P}^c) - \lambda(\mathcal{P}[n]^c)).$$

Our left-c.e. real will therefore be  $r = \frac{1}{1 - \alpha} \lambda(\mathcal{P}^c)$  with the approximation  $r_n = \left( \frac{1}{1 - \alpha} \lambda(\mathcal{P}[n]^c) \right)_{n \in \mathbb{N}}$ . It is of course possible for this real to be greater than 1, in which case it suffices to start the approximation from  $n$  sufficiently large such that  $r - r_n < 1$ . ■

**Corollary 3.8**

*Let  $X \in 2^{\mathbb{N}}$ . Then,  $X$  weakly 2-random implies  $X$  Oberwolfach random implies  $X$  difference random.*

PROOF. It is clear that Oberwolfach tests are all  $\Pi_2^0$  classes of measure 0. So if  $X$  is weakly 2-random it is Oberwolfach random. Now if  $X$  is Oberwolfach random, it has density 1 and therefore positive density in any  $\Pi_1^0$  class of positive measure and containing it. According to Theorem 19-4.10, it is therefore difference random. ■

To solve the question, it would therefore suffice to construct a Martin-Löf random which is a point of positive density in any  $\Pi_1^0$  class, and which is not a point of density 1 in at least one  $\Pi_1^0$  class, in order to exhibit an incomplete Martin-Löf random computing all K-trivials. This is what was done by Day and Miller, who then brought a final answer to the question, with the help of an appropriate forcing.

**Theorem 3.9 (Day and Miller [42])**

*There exists an MLR set  $Z \in 2^{\mathbb{N}}$  such that:*

- (1)  $\underline{\rho}(\mathcal{P} \mid Z) > 0$  for any  $\Pi_1^0$  class  $\mathcal{P}$ .
- (2)  $\underline{\rho}(\mathcal{Q} \mid Z) < 1$  for a specific  $\Pi_1^0$  class  $\mathcal{Q}$ .

PROOF. Let  $\mathcal{P}$  be a  $\Pi_1^0$  class containing only MLR sets. We define the forcing conditions  $\mathbb{P}$  by  $\langle \sigma, \mathcal{Q} \rangle \in \mathbb{P}$  if:

- (1)  $\sigma \in 2^{<\mathbb{N}}$
- (2)  $\mathcal{Q} \subseteq \mathcal{P}$  is a  $\Pi_1^0$  class
- (3)  $[\sigma] \cap \mathcal{Q} \neq \emptyset$  (and therefore  $\lambda(\mathcal{Q} \mid [\sigma]) > 0$  because  $\mathcal{Q}$  only contains MLRs)
- (4) There exists  $\delta < 1/2$  such that for any  $\tau \succeq \sigma$ , if  $[\tau] \cap \mathcal{Q}$  is not empty then  $\lambda(\mathcal{P} \mid [\tau]) \leq \lambda(\mathcal{Q} \mid [\tau]) + \delta$

We say that  $\langle \tau, \mathcal{R} \rangle$  extends  $\langle \sigma, \mathcal{Q} \rangle$  if  $\tau \succeq \sigma$  and  $\mathcal{R} \subseteq \mathcal{Q}$ . Note that  $(\epsilon, \mathcal{P})$  — where  $\epsilon$  is the empty string — is a condition with  $\delta = 0$ . Likewise, if  $(\sigma, \mathcal{Q})$  is a condition satisfying (4) with  $\delta$ , then for any  $\tau \succeq \sigma$  such that  $[\tau] \cap \mathcal{Q} \neq \emptyset$ ,  $(\tau, \mathcal{Q})$  is an extension of  $(\sigma, \mathcal{Q})$  satisfying (4) with the same  $\delta$ . Thus, any sufficiently generic filter will contain conditions  $(\sigma, \mathcal{Q})$  with  $\sigma$  of arbitrary long length.

Let  $G \subseteq \mathbb{P}$  be a sufficiently generic filter. Let  $X_G$  be the unique limit point of  $\sigma_i$  for  $\langle \sigma_i, \mathcal{Q}_i \rangle \in G$ . Since  $\mathcal{P}$  is a closed class it is clear that  $X_G \in \mathcal{P}$  and therefore  $X_G$  is an MLR set.

Let us now show that if  $G$  is sufficiently generic then  $\rho(\mathcal{P} \mid X_G) < 1/2$ . Let  $\langle \sigma, \mathcal{Q} \rangle$  be a forcing condition satisfying condition (4) with  $\delta$ . Let  $X$  be the leftmost path of  $\mathcal{Q} \cap [\sigma]$ . The class  $\mathcal{Q} \cap [\sigma] \subseteq \mathcal{P}$  is a class containing only MLR sets and therefore  $X$  is an MLR set. According to Exercize 18-1.3, MLR sets contain arbitrarily long sequences of 1. Let  $m$  be such that  $2^{-m} + \delta < 1/2$  and let  $\tau$  be such that  $\sigma \prec \tau 1^m \prec X$ . Since  $X$  is the leftmost path of  $\mathcal{Q} \cap [\sigma]$  we have  $\lambda(\mathcal{Q} \mid [\tau]) \leq 2^{-m}$  and therefore  $\lambda(\mathcal{P} \mid [\tau]) \leq 2^{-m} + \delta \leq 1/2$ . Then the forcing condition  $\langle \tau, \mathcal{Q} \rangle$  extends  $\langle \sigma, \mathcal{Q} \rangle$ . So if  $G$  is sufficiently generic we will have  $\lambda(\mathcal{P} \mid X_G \restriction_m) < 1/2$  for infinitely many integers  $m$  and therefore  $\rho(\mathcal{P} \mid X_G) < 1/2$ .

Finally, we show that if  $G$  is sufficiently generic then for any  $\Pi_1^0$  class  $\mathcal{S}$  we have  $X_G \in \mathcal{S}$  implies  $\rho(\mathcal{S} \mid X_G) > 0$ . Let  $\langle \sigma, \mathcal{Q} \rangle$  be a forcing condition satisfying condition (4) with  $\delta$ . If there exists  $\tau \succeq \sigma$  such that  $\mathcal{Q} \cap [\tau] \neq \emptyset$  and such that  $\mathcal{S} \cap [\tau] = \emptyset$  we take  $\langle \tau, \mathcal{Q} \rangle$  as the forcing extension and we will then have  $X_G \notin \mathcal{S}$ . Otherwise it means  $\mathcal{Q} \subseteq \mathcal{S}$ . Let  $\varepsilon < \min(1/2 - \delta, \lambda(\mathcal{Q} \mid [\sigma]))$ . Consider the class

$$\mathcal{Q}' = \{X \in \mathcal{Q} \cap [\sigma] : \forall n \geq |\sigma| \lambda(\mathcal{Q} \mid [X \restriction_n]) \geq \varepsilon\}.$$

It is clear that  $\rho(\mathcal{Q} \mid X) \geq \varepsilon$  for all  $X \in \mathcal{Q}'$ . As  $\mathcal{Q} \subseteq \mathcal{S}$  then also  $\rho(\mathcal{S} \mid X) \geq \varepsilon$  for all  $X \in \mathcal{Q}'$ . We still have to show that  $\langle \mathcal{Q}', \sigma \rangle$  is a valid extension for which we will then have  $\rho(\mathcal{S} \mid X_G) \geq \varepsilon$ .

Let us first show that  $\mathcal{Q}' \cap [\sigma]$  is non-empty. By the choice of  $\varepsilon$  we have  $\lambda(\mathcal{Q} \mid [\sigma]) \geq \varepsilon$ . There must therefore exist according to Lemma 19-4.7 an infinite sequence  $Y \succ \sigma$  with  $Y \in \mathcal{Q}$  such that  $\lambda(\mathcal{Q} \mid [Y \restriction_n]) \geq \varepsilon$  for all  $n \geq |\sigma|$ . So  $\mathcal{Q}' \cap [\sigma]$  is non-empty. Let now  $\tau \succeq \sigma$  be a string such that  $\mathcal{Q}' \cap [\tau]$  is non-empty. Let us show that we then have  $\lambda(\mathcal{P} \mid [\tau]) \leq \lambda(\mathcal{Q}' \mid [\tau]) + \delta + \varepsilon$  making  $\delta + \varepsilon < 1/2$  the rational satisfying condition (4). Let

$$\mathcal{U} = \{X \succeq \sigma : \exists n \lambda(\mathcal{Q} \mid [X \restriction_n]) < \varepsilon\}.$$

Since  $\mathcal{Q}' \cap [\tau]$  is non-empty we have  $\lambda(\mathcal{Q} \mid [\tau \restriction_n]) \geq \varepsilon$  for all  $n$  with  $|\sigma| < n \leq |\tau|$  and therefore  $[\tau] \not\subseteq \mathcal{U}$ . Let  $W$  be the minimal prefix-free set of strings describing  $\mathcal{U}$ . As  $[\tau] \not\subseteq \mathcal{U}$  we then have

$$\lambda(\mathcal{Q} \cap \mathcal{U} \cap [\tau]) = \sum_{\rho \in W, \rho \succ \tau} \lambda(\mathcal{Q} \cap [\rho]).$$

Moreover for any string  $\rho \in W$  we have  $\lambda(\mathcal{Q} \mid [\rho]) < \varepsilon$  by definition of  $\mathcal{U}$ . Therefore

$$\lambda(\mathcal{Q} \cap \mathcal{U} \cap [\tau]) \leq \varepsilon \sum_{\rho \in W, \rho \succ \tau} 2^{-|\rho|} \leq \varepsilon 2^{-|\tau|}.$$

As we have  $\mathcal{Q}' \cap [\tau] = \mathcal{Q} \cap [\tau] \setminus (\mathcal{Q} \cap \mathcal{U} \cap [\tau])$  then  $\lambda(\mathcal{Q}' \cap [\tau]) = \lambda(\mathcal{Q} \cap [\tau]) - \lambda(\mathcal{Q} \cap \mathcal{U} \cap [\tau])$  and therefore  $\lambda(\mathcal{Q}' \cap [\tau]) \geq \lambda(\mathcal{Q} \cap [\tau]) - \varepsilon 2^{-|\tau|}$ .

So  $\lambda(\mathcal{Q}' \mid [\tau]) \geq \lambda(\mathcal{Q} \mid [\tau]) - \varepsilon$ . Since  $\mathcal{Q}' \cap [\tau]$  is non-empty then also  $\mathcal{Q} \cap [\tau]$  is non-empty and we therefore have  $\lambda(\mathcal{P} \mid [\tau]) \leq \lambda(\mathcal{Q} \mid [\tau]) + \delta \leq \lambda(\mathcal{Q}' \mid [\tau]) + \delta + \varepsilon$ . So  $\langle \sigma, \mathcal{Q}' \rangle$  is a valid condition with  $\delta + \varepsilon < 1/2$ . ■

**Corollary 3.10 (Day and Miller [42])**

*Every c.e. K-trivial is computable by an incomplete MLR.*

PROOF. It suffices according to Theorem 3.9 to consider an MLR set  $Z$  which is a point of positive lower density without being a point of lower density 1. According to Theorem 19-4.10 such a set cannot compute  $\emptyset'$ . According to Theorem 3.7 such a set cannot be Oberwolfach random, because otherwise it would be a point of lower density 1. By Theorem 3.5 this set  $Z$  therefore computes all the c.e. K-trivials. ■

## 4. New proof that K-trivial implies low-for-K

We now show how to combine the results seen so far to prove that any c.e. K-trivial is low-for-K, without using the golden run proof. Let  $A$  be a c.e. K-trivial. Then, according to Corollary 3.10,  $A$  is computed by an incomplete MLR  $Z$ . Suppose by contradiction that  $Z$  is not  $\text{MLR}(A)$ . Then, according to Proposition 3.1 we must have  $A \oplus Z \geq_T \emptyset'$  and therefore  $Z \geq_T \emptyset'$  which contradicts the fact that  $Z$  is incomplete. So  $Z$  is  $\text{MLR}(A)$  and  $A$  is therefore a basis for randomness. According to Theorem 1.7,  $A$  is thus low-for-K.

We just have to show that every K-trivial is computed by a c.e. K-trivial, And this without using the golden run proof. As any c.e. K-trivial is low-for-K and as the low-for-K are downward-closed in the Turing degrees, we can easily deduce that any K-trivial is also low-for-K. We appeal for that to a clever use of a concept introduced by Solovay: computable compression functions which do as well as  $K$  infinitely often.

**Definition 4.1.** A *Solovay function* is a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $K(n) \leq^+ g(n)$  for all  $n$  and such that  $g(n) \leq^+ K(n)$  for infinitely many integers  $n$ . ◇

**Proposition 4.2.** There exists a Solovay function. ★

PROOF. Consider the function  $g_S : \mathbb{N} \rightarrow \mathbb{N}$  such that  $g_S(\langle \sigma, n, t \rangle) = |\sigma|$  if  $t$  is the smallest computation time such that  $U(\sigma)[t] \downarrow = n$ . Otherwise  $g_S(\langle \sigma, n, t \rangle) = 2\langle \sigma, n, t \rangle$ . As  $K(\langle \sigma, n, t \rangle) \leq^+ 2\langle \sigma, n, t \rangle$  we have  $K(n) \leq^+$

$g_S(n)$  for all  $n$ . Let us show that we have  $K(n) =^+ g_S(n)$  for infinitely many integers  $n$ .

Let  $M$  be the prefix-free machine which on  $\sigma$  such that  $t$  is the smallest for which  $U(\sigma)[t] \downarrow = n$ , returns  $\langle \sigma, n, t \rangle$ . With the machine  $M$  we have  $K(\langle \sigma, n, t \rangle) \leq^+ |\sigma|$ . As we can compute  $n$  from  $\langle \sigma, n, t \rangle$  we also have  $K(n) \leq^+ K(\langle \sigma, n, t \rangle)$ . Finally if  $\sigma$  is the smallest string such that  $U(\sigma) \downarrow = n$  we have by definition  $|\sigma| = K(n)$ .

So in this case we have  $|\sigma| = K(n) \leq^+ K(\langle \sigma, n, t \rangle) \leq^+ |\sigma|$ , so  $g_S(\langle \sigma, n, t \rangle) =^+ K(\langle \sigma, n, t \rangle)$ . ■

Bienvenu and Downey showed that the function defined by Solovay could be used to characterize K-triviality:

**Proposition 4.3 (Bienvenu and Downey, [14]).** Let  $g_S$  be the Solovay function of Proposition 4.2. If  $K(A \upharpoonright_n) \leq g_S(n) + c$  for all  $n$  then  $A$  is K-trivial via some constant  $c + d$ . ★

PROOF. For any  $n$ , let  $\sigma_n$  be the smallest string such that  $U(\sigma_n) = n$  and let  $t_n$  be the smallest computation time such that  $U(\sigma_n)[t_n] = n$ . We have  $K(A \upharpoonright_{\langle \sigma_n, n, t_n \rangle}) \leq g_S(\langle \sigma_n, n, t_n \rangle) + c = |\sigma_n| + c = K(n) + c$ . As we can compute  $A \upharpoonright_n$  from  $A \upharpoonright_{\langle \sigma_n, n, t_n \rangle}$  we therefore have a constant  $d$  such that  $K(A \upharpoonright_n) \leq K(A \upharpoonright_{\langle \sigma_n, n, t_n \rangle}) + d$  and therefore such that  $K(A \upharpoonright_n) \leq K(n) + c + d$ . ■

Later Bienvenu, Merkle and Nies [20] will show that the previous proposition works for all Solovay functions, but this is a much more difficult proof. We now have all the necessary ingredients to show that any K-trivial is bounded in the Turing degrees by a c.e. K-trivial, a result which was first demonstrated by Nies from the technique of cost functions and the **golden run** proof, and which can now be shown in a completely different way.

**Theorem 4.4 (Nies [165])**

*Every K-trivial is computed by a c.e. K-trivial.*

PROOF. Let  $g_S : \mathbb{N} \rightarrow \mathbb{N}$  be the Solovay function of Proposition 4.2. Let  $T = \{\sigma \in 2^{<\mathbb{N}} : \forall n < |\sigma| \ K(X \upharpoonright_n) \leq g_S(n) + c\}$  be the c.e. tree which represents the  $\Pi_1^0(\emptyset')$  class of the set  $\{X : K(X \upharpoonright_n) \leq g_S(n) + c\}$ . It is clear that  $[T]$  contains all K-trivials with constant  $c$ , and according to Proposition 4.3,  $[T]$  only contains K-trivials with some constant  $c + d$  and therefore in particular contains only a finite number of elements. Let  $A$  be one of these K-trivials and let  $\sigma \prec A$  be such that  $A$  is the only infinite path of  $T$  extending  $\sigma$ . Let  $T \upharpoonright_\sigma$  be the tree restricted to strings comparable with  $\sigma$ . We modify the enumeration of  $T \upharpoonright_\sigma$  as follows: given  $T_s \upharpoonright_\sigma$  at

step  $s$ , we enumerate in  $T_s \upharpoonright_\sigma$  a new string  $\tau \succeq \sigma$  only if it is greater than or equal to all strings currently in  $T_s \upharpoonright_\sigma$ , in which case we also list all prefixes of  $\tau$  in  $T_s \upharpoonright_\sigma$ . Let  $T_\sigma$  be the c.e. tree resulting from this enumeration.

Note that since  $T$  contains extensions of  $\sigma$  of arbitrarily long length, then also our new enumeration  $T_\sigma$  also contains extensions of  $\sigma$  of arbitrarily long length. In particular by König's lemma  $T_\sigma$  contains an infinite path. As  $A$  is the unique infinite path of  $T$  which extends  $\sigma$  then this infinite path is necessarily  $A$ . Since  $A$  is the unique infinite path of  $T_\sigma$  then  $T_\sigma$  allows us to compute  $A$ .

Let us show that  $T_\sigma$  is K-trivial via its representation  $X_\sigma$  where  $X_\sigma(n) = 1$  iff  $n$  encodes a string enumerated in  $T_\sigma$ . Note that  $X_\sigma \upharpoonright_{2^n}$  encodes all strings in  $T_\sigma$  of length less than or equal to  $n$ . Let us show  $K(X_\sigma \upharpoonright_{2^n}) \leq^+ g_S(n)$  for all  $n$ . For any  $n$  let  $\sigma_n$  be the last string of length  $n$  which is enumerated in  $T$ .

Note that  $\sigma_n$  allows to uniformly compute  $X_\sigma \upharpoonright_n$ : it suffices to wait until  $\sigma_n$  is enumerated in  $T_\sigma$  and then return all the strings of length less than or equal to  $|\sigma_n|$  enumerated in  $T_\sigma$  so far. By the enumeration property of  $T_\sigma$ , we know that no other string of length less than or equal to  $n$  will be enumerated subsequently. As  $\sigma_n \in T$ , by definition of  $T$ ,  $K(\sigma_n) \leq g_S(n) + c$ . Finally, since  $X_\sigma \upharpoonright_n$  is computable from  $X_\sigma \upharpoonright_{2^n}$  we have  $K(X_\sigma \upharpoonright_n) \leq^+ K(X_\sigma \upharpoonright_{2^n})$ . This gives us  $K(X_\sigma \upharpoonright_n) \leq^+ K(X_\sigma \upharpoonright_{2^n}) \leq^+ K(\sigma_n) \leq^+ g_S(n)$  for all  $n$ . So  $X_\sigma$  is K-trivial. ■

# Summary Diagram

Here is a table which summarizes the different notions of randomness discussed so far.

Randomness	Tests	Characterization	Density in $\Pi_1^0$ containing it
Weakly 2-randomness	$\Pi_2^0$ of measure 0	MLR and computes no non-computable $\Delta_2^0$	
Oberwolfach randomness	$\bigcap_n \mathcal{U}_n$ such that $\lambda(\mathcal{U}_n) < r - r_n$ with $\lim_n r_n = r$	MLR and does not compute all the K-trivials	density lower than 1
Difference randomness	$\mathcal{F} \cap \bigcap_n \mathcal{U}_n$ such that $\lambda(\mathcal{U}_n \cap \mathcal{F}) \leq 2^{-n}$	Incomplete MLR	positive lower density
MLR	$\bigcap_n \mathcal{U}_n$ such that $\lambda(\mathcal{U}_n) < 2^{-n}$		upper density 1

**Part III**

**Reverse Mathematics**



# Chapter 21

## Introduction

Reverse Mathematics is a set of tools from Proof Theory and Computability Theory to analyze the computational content of mathematical theorems. They make it possible to answer fundamental meta-mathematical questions such as: “What are the optimal axioms to prove ordinary theorems?” or else: “What meaning should be given to the implication of a theorem by another?”

### 1. Quest for optimal axioms

The quest for optimal axioms to prove ordinary theorems is the historical motivation of Reverse Mathematics, as conceived by Harvey Friedman in 1975–1976. As we saw in Chapter 9 on the crisis of foundations, the developments of Set Theory pushing back the limits of intuition, paradoxes showed up, sowing doubt in the solidity of the edifice of mathematics. The crisis of foundations reached its climax with the second incompleteness theorem of Gödel, stating that a consistent computably enumerable theory capable of speaking about natural integers, could not prove its own



Harvey Friedman, 1948–

consistency. Mathematicians are therefore doomed to relegate the consistency of mathematics to the level of belief, justified only by the absence of contradiction identified to date, despite intensive use of mathematics on a daily basis.

Theorems can therefore be seen as edifices of knowledge built on the belief in the consistency of a system of axioms. The stronger the axioms used in the proof of a theorem, the more fragile this knowledge will be. Friedman's approach therefore consisted in trying to control these risks, by determining which theorems would be invalidated if one had to renounce certain axioms. In other words, it was a question of determining the level of reliability of theorems, in direct response to the crisis of the foundations. The historical question of Reverse Mathematics was therefore the following:

What are the *optimal axioms* allowing to prove theorems of *ordinary mathematics*?

There are several important components to this question that should be considered.

First of all, what do we mean by “ordinary mathematics”? It is important to understand that Friedman's approach was above all empirical, and did not seek exhaustiveness: it is not a matter of being able to analyze the axiomatic power of the very last results of Set Theory, which is a meta-mathematical branch whose logical power is far removed from traditional mathematics. It is rather a matter of studying mathematics “of everyday life”, like the usual theorems of algebra or analysis.

How to prove that a set of axioms is optimal to prove a theorem? Carefully analyzing the proof of the theorem to identify its assumptions or axioms is not enough. There may very well be a new proof appealing to more elementary axioms. The notion of optimal axiom is therefore not a property relating to a proof, but to a theorem. Let  $A_1, \dots, A_n$  be axioms *sufficient* to prove a theorem  $T$ . In other words,  $A_1 \wedge \dots \wedge A_n \rightarrow T$ . To ensure that the axioms are *necessary*, it suffices to prove the reverse implication (hence the name of “Reverse Mathematics”). We then end up with the equivalence

$$A_1 \wedge \dots \wedge A_n \leftrightarrow T$$

### 1.1. Choice of a base theory

The process would probably be doomed to failure if one restricted oneself to logical implication without fixing a base theory. Indeed, the logical implication is a very fine relation which would place each theorem — and even different formulations of a theorem — in different “logical degrees”.

It is natural to try to get away from the details of formulation and to work modulo “elementary” operations: our logical implications can be established relative to a base theory allowing to carry out these elementary operations. One must be careful in the choice of this theory: if this one is too strong — for example if it already allows to prove most of the theorems — the informative value of an implication  $P \rightarrow Q$ , relatively to this theory, will be weakened.

This is where Computability Theory comes in: we are going to fix a base theory,  $\text{RCA}_0$ , capturing *computable mathematics*. The choice of this theory defines the semantic interpretation of the implication relation. For example, if we consider that the Zermelo-Fraenkel (ZF) theory represents consensual mathematics, i.e., mathematics widely accepted by the community, an implication  $\text{ZF} \vdash P \rightarrow Q$  means that by consensus, if we admit  $P$ , then we admit  $Q$ . Likewise, a proof of  $\text{RCA}_0 \vdash P \rightarrow Q$  means that if we trust computable mathematics, and if we admit  $P$ , then it is logical to consider  $Q$  to be true. In practice, the theory  $\text{RCA}_0$  is much weaker than ZF, and in particular many classical theorems are not provable in  $\text{RCA}_0$ , as we will see.

## 1.2. Second-order arithmetic

To carry out the historical program of Reverse Mathematics, namely the quest for optimal axioms, it is necessary to settle a formal framework, starting with the choice of a language. At first glance, the most obvious choice is that of a foundational theory such as Set Theory. Indeed, as we saw in Section 9-4, the language of sets allows a unified formalization of the totality of mathematics. The choice of Reverse Mathematics, however, fell on the language of second-order arithmetic, that is to say a language where one manipulates and quantifies on natural integers and sets of integers.

This choice may seem surprising: the sets of natural integers being by nature countable, second-order arithmetic can only handle countable objects. How then to speak of the usual objects which are not, like the functions from  $\mathbb{R}$  to  $\mathbb{R}$ ? This is in fact not possible in all generality, but the work of Hilbert and Bernays, *Grundlagen der Mathematik* (1934-1936), showed that a large part of mathematics could be formalized, modulo an appropriate coding, in second-order arithmetic. Thus, for example the *continuous* — and even Borel — functions from  $\mathbb{R}$  to  $\mathbb{R}$  all admit a countable representation. Remember that the purpose of Reverse Mathematics is not to be exhaustive, but to reflect a general trend in mathematics.

In return for this slight loss of generality, second-order arithmetic presents a considerable advantage: the manipulated mathematical objects being all represented by integers and sets of integers, we can benefit from the tools

of Computability Theory to talk about the complexity of axioms. There is indeed a robust notion of computability on integers, and in particular a characterization of computable sets as those definable by  $\Delta_1^0$  predicates. The various representations of the same theorem in second-order arithmetic, using different coding functions, will in practice have no impact on the strength of the theorem, because most codings are computable, and therefore equivalent in the base theory  $\text{RCA}_0$ .

### 1.3. Structure phenomenon

Since the inception of Reverse Mathematics, hundreds of theorems have been analyzed, from all branches of mathematics. We will find this approach in the excellent work of Steven Simpson, *Subsystems of Second-Order Arithmetics*. The systematic study of classical theorems from the angle of Reverse Mathematics has given rise to two empirical observations:

*Most ordinary theorems require low axiomatic power.* This observation can be seen as a partial response to the crisis of foundations, delivering an optimistic message: yes, the theory of arithmetic is incomplete and we have to be content with an experimental belief in its consistency, yes, adding axioms to this theory potentially only worsens the risks, but most of the usual theorems only require a “weak part” of these additional axioms. Mathematics is therefore robust in the face of possible inconsistencies due to too strong axioms.

*Mathematics is computationally very structured.* More precisely, there are four main systems of axioms, linearly ordered by the implication modulo  $\text{RCA}_0$ , such that if we take a classical theorem at random<sup>1</sup>, There is a good chance that it is equivalent to one of the four systems modulo  $\text{RCA}_0$ , or even already provable in  $\text{RCA}_0$ . This observation is called the *Big Five phenomenon*.

This second observation should however be put into perspective. There are counter-examples — coming in particular from Ramsey theory — behaving in a much more chaotic manner. This has led some researchers to raise the objection according to which this structural phenomenon observed in Reverse Mathematics is mostly due to a human bias, and that this structure is more that of the brains of mathematicians than of mathematics itself.

## 2. Comparison of theorems

The Reverse Mathematics evolved little by little, and many branches were born on the original ground of the search for the optimal axioms. Much

---

<sup>1</sup>“Chance” is to be taken in the informal sense of the term. This is an empirical observation and not a result of probabilities.

of the discipline consists of the comparison and classification of theorems. It is common in mathematics to hear claims like “Theorems  $A$  and  $B$  are equivalent”, or “Theorem  $A$  is not a consequence of Theorem  $B$ ”. Reverse Mathematics can give a precise meaning to these informal statements.

From a purely logical point of view, all theorems are equivalent, in the sense that they are all interpreted by the truth value **True**. What meaning can be given to the intuition of the implication or of the equivalence between two theorems? We could for example consider that a theorem  $T_0$  implies another theorem  $T_1$  if the proof of  $T_1$  involves the statement  $T_0$ . However, it is possible to replace each occurrence of  $T_0$  by its proof, in the proof of  $T_1$  to obtain a new proof of  $T_1$  not involving  $T_0$ . This formalization attempt is therefore not the right one.

If we come back to first intuition, a theorem  $T_0$  implies a theorem  $T_1$  if  $T_1$  can be proved *in an elementary way*, using  $T_0$  as a blackbox. Reverse Mathematics allows us to formalize the notion of elementary reasoning using the system  $\text{RCA}_0$ . It is then possible to give a formal meaning to the implication  $T_0 \rightarrow T_1$  by proving it in  $\text{RCA}_0$ . Reverse Mathematics, initially designed to identify the axioms necessary for mathematics, becomes a tool for classifying theorems.

**Example 2.1.** One will find in the literature statements such as “The Intermediate Value Theorem is a consequence of the Least Upper Bound property”. A weak version of the Intermediate Value Theorem — Bolzano’s theorem — states that if a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is continuous over an interval  $[a, b]$  with  $f(a) < 0$  and  $f(b) > 0$ , then there exists  $c \in ]a, b[$  such that  $f(c) = 0$ . The *Least Upper Bound property* states that any non-empty and bounded part of  $\mathbb{R}$  has a least upper bound.

Suppose the Least Upper Bound property in order to prove Bolzano’s theorem. Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a continuous function such that  $f(a) < 0$  and  $f(b) > 0$ . Let  $S = \{x \in [a, b] : f(x) < 0\}$ . The set  $S$  contains  $a$  and is bounded by  $b$ , so it admits a least upper bound  $c$ . Assuming by contradiction  $f(c) \neq 0$ , then for an open interval  $I$  such that  $f(c) \in I$  with  $\max I < 0$ , we have, by continuity of  $f$ , an open interval  $J$  containing  $c$  such that  $f(J) \subseteq I$ . The inequality  $c < \sup J$  contradicts the least upper bound quality of  $c$ . So  $f(c) = 0$ .

The previous example is an “elementary” proof ( $\text{RCA}_0$  will give a precise meaning to this) of Bolzano’s theorem from the Least Upper Bound property. From the point of view of Reverse Mathematics, these theorems must be formalized in the language of second-order arithmetic. A continuous function can be represented by an encoding of all open classes  $f^{-1}(]a, b[)$  for all rational  $a < b$ . The Least Upper Bound property cannot be stated

for any sets of real numbers, but we can state it for all Borel classes, which can be coded by countable objects.

The interest of the implication in  $\text{RCA}_0$  to classify theorems remains however relatively limited, insofar as most of the theorems are equivalent to five main sets of axioms. There are however refinements of the implication where one will control the uses of  $T_0$  in the proof of  $T_1$ , or even prohibit case analysis. These restrictions lead respectively to the *computable reduction* and the *Weihrauch reduction*, which we will see in detail in this part. Nowadays, Reverse Mathematics has taken on a broader meaning, and forms a banner under which the foundational approach of the search for optimal axioms is grouped, but also the classification of theorems through various computational reductions.

# Chapter 22

## Second-order arithmetic

Second-order arithmetic, denoted  $Z_2$ , is a theory introduced by Hilbert and Bernays in *Grundlagen der Mathematik*, allowing to speak of integers and sets of integers. Although the theory only handles countable objects, Hilbert and Bernays showed that a large part of classical mathematics was already provable in  $Z_2$ . The axioms of second-order arithmetic relative to integers (and which do not speak of sets of integers) coincide with those of Peano arithmetic. The theory  $Z_2$  is not for all that a *conservative extension* of PA (see Definition 23-6.1): there exist formulas involving only natural integers which are provable in  $Z_2$  but not in PA, starting with the consistency statement of PA.



Paul Bernays, 1888–1977

In this chapter, we will study several subsystems of second-order arithmetics which have been shown to be particularly important in reverse mathematics.

## 1. $Z_2$ language

The language of second-order arithmetic corresponds to that of Peano arithmetic, augmented with variables ranging over sets of integers, quantifiers on these sets, and the binary relation of membership between an integer and a set.

**Definition 1.1.** The language  $\mathcal{L}_{Z_2}$  of second-order arithmetic includes:

- (1) *First-order variable* symbols  $x, y, z, \dots$  for natural numbers.
- (2) *Second-order variable* symbols  $X, Y, Z, \dots$  for sets of integers.
- (3) *Logical connector* symbols:  $\wedge, \vee, \rightarrow, \neg$  and the parentheses  $()$ .
- (4) *Quantifier* symbols on integers and on sets of integers:  $\forall, \exists$ .
- (5) *Binary function* symbols on the following integers:  $+$ ,  $\times$ .
- (6) *Binary relation* symbols on the following integers:  $=$ ,  $<$ .
- (7) The membership relation symbol:  $\in$ .
- (8) *Integer constant* symbols  $\dot{0}, \dot{1}$ . ◇

The language of second-order arithmetic is *two-sorted*, in the sense that it handles two distinct types of objects: integers (first-order), and sets of integers (second-order). There are therefore two types of terms: first-order terms and second-order terms:

**Definition 1.2.** The *first-order terms* of  $Z_2$  are defined inductively as follows.

- (1) A first-order variable or constant is a first-order term.
- (2) If  $t_1, t_2$  are first-order terms, then so are  $(t_1 + t_2)$  and  $(t_1 \times t_2)$ .

The *second-order terms* are simply second-order variables. ◇

The simplicity of second-order terms comes from the absence of functions over sets of integers in the language of second-order arithmetic. Had we added the union symbol  $\cup$ , we would then have considered  $(X \cup Y) \cup Z$  as a second-order term. Standard set operations can however be defined using certain axioms.

**Example 1.3.** The following expressions are first-order terms:  $x, (((((x + \dot{1}) + \dot{1}) + \dot{1}) \times \dot{1}) + \dot{1}), (\dot{1} + \dot{0}), (x + (y \times z))$ . The only second-order terms are the second-order variables.

As with Peano arithmetic, if function symbols are used to create language terms, relation symbols are used to create language *formulas*:

**Definition 1.4.** The *formulas of second-order arithmetic* are defined as follows.

- (1) For all first-order terms  $t_1, t_2$  and any second-order term  $X$ , then  $t_1 = t_2$ ,  $t_1 < t_2$ , and  $t_1 \in X$  are formulas. These formulas are called *atomic formulas*.
- (2) For all formulas  $F_1, F_2$ , then  $(F_1 \wedge F_2)$ ,  $(F_1 \vee F_2)$ ,  $(F_1 \rightarrow F_2)$  and  $\neg F_1$  are formulas.
- (3) For any formula  $F$ , then  $\forall x F$ ,  $\exists x F$ ,  $\forall X F$  and  $\exists X F$  are formulas.  $\diamond$

**Example 1.5.** The formula

$$\forall X ([0 \in X \wedge \forall y (y \in X \rightarrow y + 1 \in X)] \rightarrow \forall z z \in X)$$

expresses induction.

A formula is *closed* if it does not have any free variable. Unless specified otherwise, a formula will not be assumed by default to be closed.

### Formulas and free variables

Let us remember the notation  $F(x)$  meaning that  $x$  is free in  $F$ . In order not to overload certain statements, one will not always make explicit all the free variables of  $F$ . This will for example be the case for formulas  $F(x) \equiv G(X_1, \dots, X_s, a_1, \dots, a_t, x)$  where  $X_1, \dots, X_s$  and  $a_1, \dots, a_t$  are free variables, respectively of the second and first order, which will be replaced by *parameters* in a certain structure.

Let us recall that the formulas of second-order arithmetic can be hierarchized according to their alternations of quantifiers. In particular, an *arithmetic* formula is a formula of  $\mathcal{L}_{\mathbb{Z}_2}$  where only quantifiers on integers are allowed, although the formula may contain free variables of the first or second order (see Section 10-1).

## 2. $\mathbb{Z}_2$ theory

Let us come to the axioms of second-order arithmetic. We find there the axioms of Robinson arithmetic to govern the behavior of natural numbers and usual operations (see Section 9-2.3). We give them back here for completeness:

- |   |   |
|---|---|
| (1) $\forall x \neg(x + \dot{1} = \dot{0})$<br>(2) $\forall x (x = \dot{0} \vee \exists y (x = y + \dot{1}))$<br>(3) $\forall x \forall y (x + \dot{1} = y + \dot{1} \rightarrow x = y)$<br>(4) $\forall x (x + \dot{0} = x)$ | (5) $\forall x \forall y (x + (y + \dot{1}) = (x + y) + \dot{1})$<br>(6) $\forall x (x \times \dot{0} = \dot{0})$<br>(7) $\forall x \forall y (x \times (y + \dot{1}) = (x \times y) + x)$<br>(8) $\forall x \forall y (x < y \leftrightarrow (\exists z (z \neq \dot{0} \wedge x + z = y)))$ |
|---|---|

Recall that we note  $\mathbf{Q}$  Robinson's theory of arithmetic, consisting of the axioms (1) - (8) above.

### 2.1. Comprehension scheme

So far, we have only specified the behavior of natural numbers. We are now going to define a scheme of axioms which will allow us to “construct” sets, in other words, to ensure that these sets will exist in any model. We have seen in the definition of the axioms of Set Theory (see Section 9-4) the restricted comprehension scheme. We will add a similar scheme for any formula (with free variables) of second-order arithmetic  $F(x)$ :

$$\exists X \forall y (y \in X \leftrightarrow F(y)) \quad (9)$$

We will suppose that the variable  $X$  does not appear freely in the formula  $F$ .

#### Universal closure of free variables

The comprehension scheme applies to *any formula with free variables*: any formula  $F(x) \equiv G(X_1, \dots, X_s, a_1, \dots, a_t, x)$  where  $X_1, \dots, X_s$  and  $a_1, \dots, a_t$  are free variables, respectively of the second and of the first order. The idea is then that the comprehension scheme must be true whatever the possible values of the free variables. To be completely formal, we should express (9) in the form

$$\forall Y_1 \dots \forall Y_s \forall a_1 \dots \forall a_t \exists X \forall y (y \in X \leftrightarrow G(Y_1, \dots, Y_s, a_1, \dots, a_t, y)).$$

In order to keep a little lightness in our notations, it will be understood below that we do not always explicitly note the free variables, which must then be universally quantified if necessary.

The comprehension scheme adds computably very complex sets to the models of second-order arithmetic. For example, we find the whole arithmetic hierarchy, starting with the halting problem:

**Example 2.1.** The predicate  $\Phi_z(X, z) \downarrow$  corresponds to a  $\Sigma_1^0(X)$  formula of arithmetic. So the comprehension scheme applied to the formula  $F(G, z) \equiv \Phi_z(G, z) \downarrow$  with a set which we suppose to exist  $X$ :

$$\exists Y \forall z (z \in Y \leftrightarrow \Phi_z(X, z) \downarrow)$$

guarantees the existence of  $X'$ .

Note that the formulas  $F$  can also contain second-order quantifications. That gives us a considerable definitional power, which we will approach in Part IV. Using second-order quantifications, it will for example be possible to show the existence of the Turing  $\omega$ -jump defined by  $\emptyset^{(\omega)} = \bigoplus_n \emptyset^{(n)}$ . Note that these second-order quantifications make it possible to make self-referencing definitions, in the sense that the formula  $F(x)$  can contain a universal quantifier on the sets of integers, and therefore which will take in particular as value the set that we are defining. The theory of second-order arithmetic is therefore a fundamentally *unpredicative* system in the sense of Poincaré (see Section 9-1 and Example 8.5).

## 2.2. Induction scheme

Recall that, to define Peano arithmetic, we added the induction scheme for any arithmetic formula. The temptation would be to extend this scheme of axioms to all the formulas of second-order arithmetic  $F(x)$  with free variables, namely:

$$(F(\dot{0}) \wedge (\forall x (F(x) \rightarrow F(x + \dot{1})))) \rightarrow \forall x F(x) \quad (10)$$

However, we will use an axiom which, combined with the comprehension scheme, implies the induction scheme (10) for any second-order formula.

$$\forall X ([\dot{0} \in X \wedge \forall y (y \in X \rightarrow y + \dot{1} \in X)] \rightarrow \forall z z \in X) \quad (11)$$

Note that (11) is not a scheme, but a simple axiom, insofar as it is not parameterized by a formula.

**Exercise 2.2.** Show that the axiom of induction on sets (11) and the comprehension scheme (9) imply the induction scheme (10) on second-order arithmetic formulas.  $\diamond$

### Notation

We note  $Z_2$  the theory of second-order arithmetic, composed of  $\mathbf{Q}$  (Robinson's axioms (1) - (8) above), the comprehension scheme (9), and the induction axiom (11) on sets.

As explained previously, despite the simplicity of its axioms, Hilbert and Bernays have shown that a large part of traditional mathematics can already be formalized in  $Z_2$ .

### 3. Semantics of second-order arithmetic

The intentional model of second-order arithmetic is of course

$$(\mathbb{N}, \mathcal{P}(\mathbb{N}), +, \times, <, 0, 1),$$

that is to say the natural integers and sets of integers, equipped with standard operations.

#### 3.1. Second order

There are two notions of structures to interpret second-order arithmetic: Henkin structures and full structures. In all its generality, a structure in the language of second-order arithmetic specifies two sets  $M$  and  $S$  representing the first and second order, respectively, as well as operations  $+$ ,  $\times$  on  $M$ , an order relation  $<$  on  $M$ , and a relation  $\in$  between  $M$  and  $S$ . Henkin structures are restricted to cases where  $M$  and  $S$  are disjoint and where  $S \subseteq \mathcal{P}(M)$  with  $\in$  denoting the true membership relation.

**Definition 3.1.** A (Henkin) structure in  $\mathcal{L}_{Z_2}$  is given by a tuple  $\mathcal{M} = (M, S, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  where  $M$  and  $S$  are two disjoint sets such that  $S \subseteq \mathcal{P}(M)$ ,  $+^{\mathcal{M}}, \times^{\mathcal{M}} : M \times M \rightarrow M$  are two operations on  $M$  and  $<^{\mathcal{M}} \subseteq M \times M$  is a relation on  $M$ . We also have the relation of equality on the elements of  $M$ , as well as an element  $0^{\mathcal{M}} \in M$  corresponding to the constant symbol  $\dot{0}$  and an element  $1^{\mathcal{M}} \in M$  corresponding to the constant symbol  $\dot{1}$ . ◇

Note that there is no equality relation on the sets. Leon Henkin proved that Theorem 9-2.22 of completeness was always valid in the structures which bear his name: we can essentially reduce Henkin structures to first-order structures as seen in Section 9-2.4, and for which completeness is verified. Usual first-order structures contain only one set of elements  $E$ . For such structures, we can add predicates  $M$  and  $S$  allowing to determine which elements of  $E$  are of first or second order, and axioms to simulate the fact that we have a Henkin's structure, eg  $\forall x (M(x) \wedge \neg S(x)) \vee (\neg M(x) \wedge S(x))$  indicates that any element is either of first or second order.

The standard semantics associated with second-order arithmetic is restricted to the particular case of Henkin structures where the second-order contains all the subsets of  $M$ .

**Definition 3.2.** A *full structure* in  $\mathcal{L}_{Z_2}$  is a Henkin structure of the form  $\mathcal{M} = (M, \mathcal{P}(M), +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$ , that is, where the second-order is the full power set  $\mathcal{P}(M)$ .  $\diamond$

In a Henkin structure with  $M$  and  $S \subseteq \mathcal{P}(M)$ , a sentence of the type  $\forall X \Phi(X)$  will be verified if it is true for all the elements of  $S$ . On the contrary in a full structure, the universal quantification is really done on all the subsets of  $M$ . We will see the kind of consequence that this distinction can have in Chapter 31.

We generally call *standard semantics* or *semantics of full models* the study of second-order arithmetic where interpretation is restricted to full structures. Standard semantics do not satisfy such good properties as the more general Henkin semantics. Indeed, there is a formula which fixes in a unique way what can be the set  $M$  of the first-order elements in a full structure. It is simply the conjunction of the axioms of  $\mathbf{Q}$  with the axiom of induction (11) above.

$$\forall X ((0 \in X \wedge \forall y (y \in X \rightarrow y + 1 \in X)) \rightarrow \forall z z \in X) \wedge \mathbf{Q} \quad (\text{a})$$

Let  $\mathcal{M} = (M, \mathcal{P}(M), +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  be a full structure satisfying (a). Let  $\omega^{\mathcal{M}} = \{0^{\mathcal{M}}, 1^{\mathcal{M}}, 1^{\mathcal{M}} + 1^{\mathcal{M}}, 1^{\mathcal{M}} + 1^{\mathcal{M}} + 1^{\mathcal{M}}, \dots\}$ . Note that  $\omega^{\mathcal{M}} \subseteq M$ . It is possible to show that the axioms of  $\mathbf{Q}$  imply that  $(\omega^{\mathcal{M}}, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}})$  is necessarily isomorphic to the standard integers. The induction axiom ensures that  $M$  has no other elements than those of  $\omega$ : by taking  $X = \omega$ , we obtain  $\forall z z \in \omega$ , therefore  $M = \omega$ . The model is then necessarily the standard model of integers with addition and multiplication. Thus, the only full model of (a) is, up to isomorphism, that of standard integers. A theory having only one model up to isomorphism is called *categorical*. A consequence of Corollary 9-2.26 is that any categorical theory is necessarily complete. The Incompleteness Theorem 9-3.10 of Gödel-Rosser asserts that there is no complete and consistent c.e. theory which extends PA. We deduce that the theory  $\text{PA} + (\text{a})$  is not c.e., which is absurd. The problem in this reasoning is Corollary 9-2.26, which follows from the completeness theorem 9-2.22 of Gödel: the latter fails for the standard semantics of second-order arithmetic.

We will therefore consider by default Henkin structures, which we will henceforth call quite simply *structures*, and which will in general not be full<sup>1</sup>.

<sup>1</sup>The so-called Löwenheim-Skolem theorem, well known in Model Theory, implies for example that there are countable models of second-order arithmetic.

### 3.2. First order

As we saw in Section 9-3.3, the first-order part of a model of  $Z_2$ , that is to say “the integers” of the model, does not necessarily correspond to the “true integers”. In any model  $\mathcal{M} = (M, S, +, \times, <, 0, 1)$  of  $Z_2$  — we have removed the exponent  $\mathcal{M}$  for clarity — we will necessarily have  $\omega \subseteq M$  for  $\omega = \{0, 1, 1+1, \dots\}$ . As mentioned in the previous section, by using the fact that  $(M, +, \times, <, 0, 1)$  satisfies the axioms **Q** of Robinson arithmetic, we easily construct from outside  $\mathcal{M}$  an isomorphism (for the order, addition and multiplication) between our true integers  $\mathbb{N}$  and  $\omega \subseteq M$ . It is perfectly possible in such a model to have  $\omega \subsetneq M$ , in which case  $\omega$  is necessarily an initial segment of  $M$  in the sense of  $<$  (see Theorem 9-3.13).

**Definition 3.3 ((Non-)standard integers).** Let  $\mathcal{M} = (M, S, +, \times, <, 0, 1)$  be a model of  $Z_2$ . The elements of  $\omega = \{0, 1, 1+1, \dots\} \subseteq S$  are called *standard integers*, as opposed to the elements of  $M \setminus \omega$  which are *non-standard integers*. A model satisfying  $M \setminus \omega \neq \emptyset$  is itself qualified as *non-standard*.  $\diamond$

#### Notation

In a model of  $Z_2$ , the standard integers will simply be denoted  $0, 1, 2, \dots$ . This notation is justified by the existence of an isomorphism between  $\mathbb{N}$  and the standard part of our model.

**Understanding non-standard models.** A non-standard model of  $Z_2$  contains by definition elements larger than our usual integers  $0, 1, 2, 3, \dots$ , and at the same time satisfies all the axioms of  $Z_2$ . It can be difficult at first to have clear ideas about such objects, for which many points can seem paradoxical. Let’s see an example. It should be perfectly clear that any (non-strictly) decreasing sequence of integers is constant above a certain rank. The immediate intuition that this statement is true rests on our vision of the standard model, but can of course also be proved: any non-empty set of integers has a smallest element (we will see with Proposition 23-3.4 that it is a consequence of induction). Let  $a_i$  be the smallest element of a sequence of integers  $(a_n)_{n \in \mathbb{N}}$ . By assumption on our sequence the following elements are either equal or strictly lower than  $a_i$ . By minimality of  $a_i$  they cannot be strictly inferior to it. They are therefore all equal to it and the sequence is then constant from the rank  $i$ .

Consider now a non-standard model  $\mathcal{M} = (M, S, +, \times, <, 0, 1)$  of  $Z_2$  and take a non-standard integer  $x \in M$ . As  $x > 1, x > 2, x > 3, \dots$ , then for any standard integer  $n$  the element  $x - n$  is different from 0. We can therefore use the axiom (2) of Robinson arithmetic to inductively construct the decreasing sequence  $x, x - 1, x - 2, x - 3, \dots$ , which is also strictly

decreasing. According to the completeness theorem and the proof of the previous paragraph, any decreasing sequence of elements in any model of  $Z_2$  should nevertheless be constant above a certain rank. So it seems that we are coming to a contradiction. What happened ? We can effectively define the sequence  $a_0 = x$  for  $x \in M$  and  $a_{n+1} = a_n - 1$  for all  $n \in \mathbb{N}$ . However, this sequence is only infinite (and even defined) from outside the model. Indeed, inside the model there is no distinction between standard and non-standard integers: the model “thinks” that all its elements are standard integers. Let’s give a more precise explanation. A sequence is a function  $f : \mathbb{N} \rightarrow \mathbb{N}$ . A second-order object therefore, whose existence is ensured by the axiom of comprehension, which must be based on a first-order formula  $F(n, y)$  such that for all  $n \in M$  there exists exactly one element  $y$  (that is to say the element  $a_n$  of our sequence) such that  $\mathcal{M} \models F(n, y)$ . In a proof intended to be read by mathematicians, the above sequence is defined via the primitive recursive function  $f(0) = x$  and  $f(n+1) = \max(0, f(n) - 1)$ . It is however advisable to formalize this definition in the language of arithmetic, for which we must use the coding of Gödel of the primitive recursive functions by formulas of arithmetic. By taking again the function  $\beta(a, b, i)$  of lemma 9-3.6 allowing to code the sequences of integers, the formula will be as follows (for more clarity we kept the functions  $\max$ ,  $\beta$  and  $-$  as is rather than substituting the formulas that represent them):

$$F(n, y) \equiv (n = 0 \wedge y = x) \vee \exists a, b \left( \begin{array}{l} y = \beta(a, b, n) \wedge \beta(a, b, 0) = x \wedge \forall i < n \\ \beta(a, b, i+1) = \max(0, \beta(a, b, i) - 1) \end{array} \right)$$

In our model of  $Z_2$ , the formula  $F$  above will indeed define a function whose domain is the entire set  $M$  and no longer simply the standard integers. The corresponding sequence will therefore have for initial segment  $a_0, a_1, a_2, a_3, \dots$  for all standard integers  $0, 1, 2, 3, \dots$ , and will indeed verify  $a_0 = x, a_1 = x - 1, a_2 = x - 2, a_3 = x - 3, \dots$ , but from the point of view of the model, it will also be defined for all the elements of  $M$ . Thus the assertion that  $(a_y)_{y \in M}$  is constant from a certain rank will indeed be verified, simply it will be constant from a non-standard rank, which from the point of view of the model is a perfectly valid integer. Note to finish that the standard part of the sequence  $(a_y)_{y \in M}$ , as initially defined from outside the model, is not distinguishable within the model. We will indeed see in Section 23-3.2 that the axiom of induction implies that it is impossible within a model to distinguish its non-standard elements from others.

**$\omega$ -structures.** In this part, we will be particularly interested with models of fragments of second-order arithmetic, within which the first-order corresponds to the standard integers  $\omega$ , equipped with the usual operations.

**Definition 3.4.** An  $\omega$ -structure is a  $\mathcal{M} = (\omega, S, +, \times, <, 0, 1)$  structure where  $\omega$  is the set of standard integers,  $S \subseteq \mathcal{P}(\omega)$ ,  $+$  and  $\times$  are the usual addition and multiplication operations, and  $<$  is the natural order. An  $\omega$ -structure is therefore entirely specified by the set  $S$ .  $\diamond$

In particular,  $(\omega, \mathcal{P}(\omega), +, \times, <, 0, 1)$  is an  $\omega$ -structure. We tend to identify a set  $S \subseteq \mathcal{P}(\omega)$  with the  $\omega$ -structure of which  $S$  is the second-order.

### 3.3. Parameter formulas

During this part, we will regularly use formulas *with parameters in a structure*. Intuitively, a formula with parameters in a structure is a formula that directly involves elements of the structure.

**Definition 3.5.** Let  $\mathcal{M} = (M, S, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  be a structure of second-order arithmetic. A *formula of  $\mathcal{L}_{Z_2}$  with parameters in  $\mathcal{M}$*  is a formula in the language  $\mathcal{L}_{Z_2} \cup \{c_n : n \in M\} \cup \{c_X : X \in S^{\mathcal{M}}\}$ , that is, a formula in the language of second-order arithmetic augmented with a constant symbol for each element of the structure.  $\diamond$

We tend to identify constant symbol and its interpretation in the structure, and directly write  $F(n, X)$  for  $F(c_n, c_X)$  with  $n \in M$  and  $X \in S^{\mathcal{M}}$ .

The parameters are closely related to the free variables of axiom schemes. Indeed, in these schemes, the free variables are universally quantified, but a structure  $\mathcal{M}$  satisfies a formula of the form  $\forall X F(X)$  iff  $\mathcal{M}$  satisfies the closed formula  $F(Z)$  for each parameter  $Z \in \mathcal{M}$ .

**Example 3.6.** Let  $F(X, y)$  be a formula of second-order arithmetic, with only free variables the set variable  $X$  and the integer variable  $y$ . Let  $\mathcal{M} \models Z_2$ . By the comprehension scheme, we have

$$\mathcal{M} \models \forall X \exists Y \forall z (z \in Y \leftrightarrow F(X, z))$$

In particular, for any parameter  $Z \in \mathcal{M}$ , we have

$$\mathcal{M} \models \exists Y \forall z (z \in Y \leftrightarrow F(Z, z))$$

Here,  $Z$  is a parameter and not a free variable, and  $F(Z, z)$  is a parameter formula.

The same nomenclature for the free variables and the parameters is justified by the fact that the first will normally be intended to be replaced by the second in the structures considered.

## 4. Formalizing Analysis in $Z_2$

As mentioned earlier, Hilbert and Bernays have shown that much of classical mathematics can be done in  $Z_2$ . We show here how to formalize some aspects of classical analysis. This will allow us to illustrate the strength of certain axiomatic subsystems of  $Z_2$  by different analysis theorems.

### 4.1. Rational numbers

It is common in Computability Theory to manipulate finite objects — like binary strings — via an integer coding system. We will use the following coding to talk about rationals.

**Definition 4.1.** In  $Z_2$ , a rational number is an integer of the form  $\langle i, n, m \rangle$  where  $i \in \{0, 1\}$  and  $m \neq 0$ . The corresponding rational number is  $n/m$  if  $i = 0$  and  $-n/m$  if  $i = 1$ .  $\diamond$

Note that we have several possible representations for the same rational number, which is not a problem, the important points being the following. First, there is a formula  $F_{\mathbb{Q}}(x)$  of  $Z_2$  which is true iff  $x$  is a rational number (i.e., iff  $x$  is an integer which satisfies the definition above). For  $x$  satisfying  $F_{\mathbb{Q}}$ , denote by  $\iota_{\mathbb{Q}}(x)$  the rational number represented by  $x$ . The usual operations of rational number manipulation must be able to be expressed in  $Z_2$ . For example, we need a formula  $F_{<}$  of  $Z_2$  such that for any  $n_1, n_2$  verifying  $F_{\mathbb{Q}}$ , we have  $\iota_{\mathbb{Q}}(n_1) < \iota_{\mathbb{Q}}(n_2)$  iff  $F_{<}(n_1, n_2)$ , or a formula  $F_{+}$  such that for any  $n_1, n_2$  satisfying  $F_{\mathbb{Q}}$ , the formula  $F_{+}(n_1, n_2, r)$  holds for a unique integer  $r$  such that  $F_{\mathbb{Q}}(r)$  and such that  $\iota_{\mathbb{Q}}(r) = \iota_{\mathbb{Q}}(n_1) + \iota_{\mathbb{Q}}(n_2)$ . The above conditions on the formulas  $F_{<}$  and  $F_{+}$  are of course expressed outside  $Z_2$  since  $\iota_{\mathbb{Q}}$  is not an object of  $Z_2$ . From inside  $Z_2$ , it will be necessary to show that the formulas allowing to define the usual functions and relations on  $\mathbb{Q}$  make it a totally ordered and dense field.

It should be clear that the usual functions and relations are primitive recursive over the set of rational codes. We will see that this automatically implies that the formulas of the type  $F_{<}$  or  $F_{+}$  above can be defined in  $Z_2$ , and even in the subsystem  $\text{RCA}_0$ , for which we will show with Theorem 23-4.6 that it is sufficient to define any primitive recursive function via an arithmetic formula.

As usual, we will handle our rational numbers without explicit recourse to coding, it being understood that the different operations are formalized via this coding in the language of arithmetic.

### 4.2. Real numbers

Real numbers are close to sets of integers. We can for example represent a real number by an ordered pair  $\langle n, X \rangle$  with  $n \in \mathbb{N}$  and  $X \in 2^{\mathbb{N}}$  where  $n$

indicates the integer part of the real number and  $X$  its fractional part. This kind of representation will not however suit us, because it does not allow to extend correctly the concepts of computability to functions over the real numbers. The following example is given via the representation of real numbers by their decimal rather than binary expansion, in order to simplify the presentation:

**Example 4.2.** Suppose that a functional, whose objective is to carry out the multiplication by 3, reads the digits  $0,333333\dots$ . It must decide after a while to extract the first digit of the result of this multiplication, but how do you know if you have to start with  $0,999999\dots$  or  $1,000000\dots$ ? If the functional opts for the first possibility, it could be then that a 4 occurs in the decimal expansion of our input, making the start of the computation false. If the machine opts for the second possibility, it could be that a 2 occurs in the decimal expansion of our input, with the same consequence  $\dots$

The pitfall of the above example comes from the topological difference between  $\mathbb{R}$  and  $2^{\mathbb{N}}$ , and in particular from the fact that some real numbers admit multiple possible representations via their decimal or binary expansion. In order to solve the problem, the commonly used solution is to work via the representation of a real  $r$  by a Cauchy sequence of rational numbers of sufficiently fast convergence. Formally:

**Definition 4.3.** In  $Z_2$ , a real number is a sequence of rational numbers  $(q_n)_{n \in \mathbb{N}}$  such that  $\forall n \forall m |q_n - q_{n+m}| \leq 2^{-n}$ . The number  $r \in \mathbb{R}$  corresponding to such a sequence is  $\lim_{n \rightarrow +\infty} q_n$ .  $\diamond$

It is clear that every real number  $r \in \mathbb{R}$  has a representation in  $Z_2$ , and reciprocally, any real number  $\mathbb{R}$  from the point of view of  $Z_2$ , that is, any sequence  $(q_n)_{n \in \mathbb{N}}$  respecting the definition above does indeed define a unique real number.

Why restrict yourself to Cauchy sequences with controlled convergence and not consider arbitrary Cauchy sequences instead, i.e., such that for all  $n$ , there exists some  $a$  for which  $|q_a - q_{a+b}| \leq 2^{-n}$  for all  $b$ ? Such sequences are also convergent, and therefore unambiguously define a unique real number. The reason is that we would like to keep a number of relations computable, in order to minimize the axioms of  $Z_2$  used in our proofs. If  $(q_n)_{n \in \mathbb{N}}$  and  $(p_n)_{n \in \mathbb{N}}$  are two arbitrary Cauchy sequences representing real numbers  $r_q$  and  $r_p$  that we suppose to be distinct, we cannot decide in a computable way, from the representations of our real number, if  $r_p < r_q$  or if  $r_q < r_p$ . On the other hand, the order becomes decidable as soon as we consider Cauchy sequences with controlled convergence.

Note that the relation  $r_p = r_q$  remains  $\Pi_1^0$  in all cases: it is expressed by  $\forall n |q_n - p_n| \leq 2^{-n+1}$ .

### 4.3. Sets of real numbers

The sets of real numbers are too big to be described by countable objects. We have in particular  $|\mathcal{P}(\mathbb{R})| > |2^{\mathbb{N}}|$ . On the other hand, we can indirectly speak of Borel classes in  $Z_2$ . An open class  $\mathcal{U} \subseteq \mathbb{R}$  can always be expressed as a union of rational intervals. For technical reasons, we consider the set  $\mathbb{Q}^\infty = \mathbb{Q} \cup \{-\infty, +\infty\}$  rather than  $\mathbb{Q}$ . Via a bijection from  $\mathbb{N}$  to  $\mathbb{Q}^\infty$ , we define a computable bijection from  $\mathbb{N}$  to open intervals of the form  $]a, b[$  for  $a, b \in \mathbb{Q}^\infty$  with  $a < b$ . The bijection induces a list  $(I_n)_{n \in \mathbb{N}}$  of these intervals. Just as we explained in the section on rational numbers, it is again understood that the chosen bijection guarantees us that the usual operations (like the union or the intersection of two intervals) as well as the usual relations (like the inclusion of one interval into another) are all primitive recursive, and therefore expressible in  $Z_2$  — and even in  $\text{RCA}_0$  — by Theorem 23-4.6 to come.

An open class  $\mathcal{U} \subseteq \mathbb{R}$  will then be represented by a set of integers  $X \in 2^{\mathbb{N}}$  such that  $\mathcal{U} = \bigcup_{n \in X} I_n$ . We define, from the outside of  $Z_2$ , the function  $\iota_\Sigma^1$  by  $\iota_\Sigma^1(X)$  as being the open class represented by  $X$ . A closed class being simply the complement of an open class, any set  $X$  also represents a closed class, and we define  $\iota_\Pi^1$  by  $\iota_\Pi^1(X) = \mathbb{R} \setminus \iota_\Sigma^1(X)$ . We can continue in this way to define the Borel classes. Once the representation of  $\Pi_n^0$  classes has been defined, a class  $\mathcal{B}$  which is  $\Sigma_{n+1}^0$  is represented by  $\bigoplus_m X_m$  such that  $\mathcal{B} = \bigcup_m \iota_\Pi^n(X_m)$ . We then have  $\iota_\Sigma^{n+1}(\bigoplus_m X_m) = \mathcal{B}$  and  $\iota_\Pi^{n+1}(\bigoplus_m X_m) = \mathbb{R} \setminus \iota_\Sigma^{n+1}(\bigoplus_m X_m)$ .

### 4.4. Continuous functions

The functions  $f : \mathbb{R} \rightarrow \mathbb{R}$  are in general not countable objects (in particular  $|\mathbb{R}^{\mathbb{R}}| > |2^{\mathbb{N}}|$ ). On the other hand, *continuous* functions are, and we can represent them unambiguously by an element of  $2^{\mathbb{N}}$ .

A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is continuous iff for any open class  $\mathcal{U}$ , the class  $f^{-1}(\mathcal{U})$  is open. Let us take our list  $(I_n)_{n \in \mathbb{N}}$  of intervals of the form  $]a, b[$  for  $a, b \in \mathbb{Q}^\infty$ . A continuous function  $f : \mathbb{R} \rightarrow \mathbb{R}$  will then be represented by a sequence  $((a_{n,m})_{m \in \mathbb{N}}, b_n)_{n \in \mathbb{N}}$  such that  $f^{-1}(I_{b_n}) = \bigcup_{m \in \mathbb{N}} I_{a_{n,m}}$  for all  $n$ . Note that an arbitrary sequence  $((a_{n,m})_{m \in \mathbb{N}}, b_n)_{n \in \mathbb{N}}$  does not always represent a continuous function. There are some rules to check:

**Definition 4.4.** In  $Z_2$ , a continuous function is a sequence

$$((a_{n,m})_{m \in \mathbb{N}}, b_n)_{n \in \mathbb{N}}$$

such that, by considering  $f^{-1}(I_{b_n}) = \bigcup_{m \in \mathbb{N}} I_{a_n, m}$  for all  $n$ , we have:

- (1)  $I_{n_1} \cap I_{n_2} = \emptyset \rightarrow f^{-1}(I_{n_1}) \cap f^{-1}(I_{n_2}) = \emptyset$ .
- (2) For every real number  $(q_n)_{n \in \mathbb{N}}$  and all  $n$ , there exists a sufficiently large  $k$  and an interval  $I_m$  with  $|I_m| \leq 2^{-n}$  such that  $]q_k - 2^{-k}, q_k + 2^{-k}[ \subseteq f^{-1}(I_m)$ .  $\diamond$

The first condition of the above definition guarantees that the relation induced by a representation  $((a_{n,m})_{m \in \mathbb{N}}, b_n)_{n \in \mathbb{N}}$  is indeed functional, and the second that the function thus defined is indeed total. Given a continuous function  $f$  represented by  $((a_{n,m})_{m \in \mathbb{N}}, b_n)_{n \in \mathbb{N}}$  and a real number  $r$  represented by  $(q_n)_{n \in \mathbb{N}}$ , the real number  $f(r)$  is unambiguously defined by the following algorithm.

- We are looking for  $k_0$  and  $]x_0, y_0[$  such that  $|x_0 - y_0| \leq 1$  and  $]q_{k_0} - 2^{-k_0}, q_{k_0} + 2^{-k_0}[ \subseteq f^{-1}(]x_0, y_0[)$ . By (2) of the above definition, the search is necessarily successful.
- Suppose  $k_n$  and  $x_n, y_n$  defined with  $|x_n - y_n| \leq 2^{-n}$ . Let  $m = (x_n + y_n)/2$ . Then, we look for  $k_{n+1} > k_n$  such that

$$\begin{aligned} & - ]q_{k_{n+1}} - 2^{-k_{n+1}}, q_{k_{n+1}} + 2^{-k_{n+1}}[ \subseteq f^{-1}(]x_n, m]) \text{ or} \\ & - ]q_{k_{n+1}} - 2^{-k_{n+1}}, q_{k_{n+1}} + 2^{-k_{n+1}}[ \subseteq f^{-1}(]m, y_n]) \text{ or} \\ & - ]q_{k_{n+1}} - 2^{-k_{n+1}}, q_{k_{n+1}} + 2^{-k_{n+1}}[ \subseteq f^{-1}(]m - 2^{-n-2}, m + 2^{-n-2}]). \end{aligned}$$

By (2) and (1) of the above definition, the search necessarily succeeds. Once  $q_{k_{n+1}}$  has been found, we define  $x_{n+1} = x_n$  and  $y_{n+1} = m$  in the first case,  $x_{n+1} = m$  and  $y_{n+1} = y_n$  in the second, and  $x_{n+1} = m - 2^{-n-2}$  and  $y_{n+1} = m + 2^{-n-2}$  in the third.

The computed real number  $f(r)$  is given by the sequence  $(x_n)_{n \in \mathbb{N}}$ , which satisfies  $|x_n - x_{n+m}| \leq 2^{-n}$  for all  $n, m$  (one could just as well choose  $(y_n)_{n \in \mathbb{N}}$  which represents the same real number). Note that, given a representation of  $f$  and a representation of  $r$ , the described algorithm always computes a representation of  $f(r)$ .

## 5. $\text{RCA}_0$ and computable mathematics

Second-order arithmetic suffers from the same problems as Set Theory with respect to the crisis of foundations: this system manipulates complex mathematical objects, for which it is difficult to trust intuition. In addition, the comprehension scheme allows quantifications on the sets, which makes the

system impredicative. We have already seen that impredicativity was sometimes a source of paradoxes (see Section 9-1), which is not very reassuring as regards the consistency of axioms.

The original goal of Reverse Mathematics being the analysis of the axioms necessary to prove a theorem, in order to restore confidence in our mathematics, it is necessary to work modulo a robust base theory, which leaves no room for doubt as to its consistency. Infinity being potentially a source of paradoxes, in particular because of certain counter-intuitive behaviors, our base theory,  $\text{RCA}_0$ , will confine itself to the axioms necessary to speak of computable sets, which in particular can be described by finitary means.

### 5.1. Understanding $\Delta_1^0$ comprehension

We have seen that the comprehension scheme makes it possible to construct all the sets of the arithmetic hierarchy, and well beyond. The first step consists in restricting it, so that only computable sets can be built. Recall that a second-order formula (with free variables) is  $\Sigma_n^0$  if it can be expressed in the form of an alternation of  $n$  quantifiers on integers, starting with an existential quantification, followed by a  $\Delta_0^0$  formula, that is to say a formula having only bounded quantifiers. In particular, the  $\Sigma_1^0$  formulas are of the form  $\exists x F(x)$  where  $F$  is  $\Delta_0^0$ . Note that in second-order arithmetic, the formula  $F$  can have free variables set. Likewise, a formula is  $\Pi_n^0$  if it can be expressed as an alternation of  $n$  quantifiers over integers, this time starting with a universal quantifier. Theorem 10-1.2 establishes a correspondence between computability and definability by the formulas of arithmetic. We recall it here:

**Theorem (10-1.2)**

Let  $A \subseteq \mathbb{N}$  and  $Z \in 2^{\mathbb{N}}$ . The following statements are equivalent:

- (1)  $A \subseteq \mathbb{N}$  is  $Z$ -c.e.
- (2) There is a  $\Sigma_1^0$  formula of second-order arithmetic  $F(X, n)$  such that  $A = \{n \in \mathbb{N} : \mathbb{N} \models F(Z, n)\}$ .
- (3) There exists a Turing functional  $\Phi(X, n)$  such that  $A = \{n \in \mathbb{N} : \Phi(Z, n) \downarrow\}$ .

We will restrict the comprehension scheme to  $\Delta_1^0$  predicates, in other words to predicates both  $\Sigma_1^0$  and  $\Pi_1^0$ . As the notion of  $\Delta_1^0$  predicate is not syntactic, unlike that of  $\Sigma_1^0$  or  $\Pi_1^0$  predicate, we will use the following trick. For any  $\Sigma_1^0$  formula  $F(x)$  and any  $\Pi_1^0$  formula  $G(x)$ , we define the axiom:

$$(\forall x (F(x) \leftrightarrow G(x))) \rightarrow \exists X \forall y (y \in X \leftrightarrow F(y)) \quad (12)$$

The premise  $\forall x (F(x) \leftrightarrow G(x))$  ensures that the predicate is  $\Delta_1^0$  before

defining the set  $X$ . Note that the formulas  $F$  and  $G$  can contain other free variables, of first or second order, on which we must universally quantify as explained in 2.1. We then obtain in substance: for any substitution of the parameters for the free variables, if  $F(x) \leftrightarrow G(x)$  with these parameters, then the set of elements which verify  $F(x)$  with these parameters does exist. We call (12) the  $\Delta_1^0$  *comprehension scheme*.

## 5.2. $\Sigma_1^0$ induction scheme

Let us now focus on the first order, which is governed by the axioms of Robinson arithmetic and the induction scheme. In a process of doubt in the face of what touches on infinity, it is legitimate to question this scheme, which makes it possible to deduce properties *for all integers*. It is used for example to show the fundamental theorem of arithmetic: any non-negative number decomposes in a unique way into product of prime numbers. It is impossible for us to check integer by integer that this is indeed the case. To do this, you have to trust the induction scheme. The reader may possibly be shocked to see induction thus called into question, and in fact, we will see in Section 23-2.1 that if we do not allow ourselves any form of induction, we cannot show much interesting. On the other hand, we can show that a relatively low level of induction is already sufficient for many usual theorems, and in particular to formalize the notion of computation and of computable functions:

**Definition 5.2.** We call  $\Sigma_1^0$  *induction scheme*, the induction scheme (10) restricted to any  $\Sigma_1^0$  formula with free variables.  $\diamond$

By following the principle of not using more than necessary, it is this induction scheme that will be used in our base system.

It is difficult at the beginning to develop a good intuition on what the absence implies, for example of induction for the  $\Sigma_2^0$  formulas. This difficulty arises from the fact that most mathematicians are used to working with standard integers, and to thinking based on this model rather than based on the axioms used. Then, for example, we admit without paying the slightest attention that any non-empty set of integers has a smallest element. We will see in Section 23-3.2 that this property is equivalent to induction. It is quite possible to build models which only satisfy  $\Sigma_1^0$  induction, and within which some non-empty  $\Pi_2^0$  sets have no smallest element.

In order to understand the consequences of the absence of certain levels of induction, it is in fact necessary, or at least very useful, to have in mind the non-standard models of arithmetic, in order to create an intuition which may otherwise be lacking. We will see in Section 23-3 more details on induction, absence of induction and the non-standard models that go with it.

**Exercise 5.3.** Show that the induction scheme (12) for formulas (with free variables)  $\Delta_0^0$  implies the induction axiom (11).  $\diamond$

### 5.3. $\text{RCA}_0$ theory

We now have the necessary elements to define our theory  $\text{RCA}_0$ .

**Definition 5.4.** We denote by  $\text{RCA}_0$  the system composed of the axioms of Robinson arithmetic augmented with the  $\Delta_1^0$  comprehension scheme (12) and the induction scheme (10) for  $\Sigma_1^0$  formulas with free variables.  $\diamond$

The acronym  $\text{RCA}_0$  stands for “Recursive Comprehension Axiom”. Recall that *recursive* is an old term for *computable*. This system owes its name to its  $\Delta_1^0$  comprehension scheme which only allows the construction of computable sets. The index “0” of  $\text{RCA}_0$  means that the induction scheme is restricted to  $\Sigma_1^0$  formulas. Indeed, Friedman had initially defined the system RCA with the induction scheme (10) for all the formulas.

Just as the restriction of the comprehension scheme allows to restrict the complexity of the *infinite* sets of the system, we will see that the induction scheme allows — in a sense which will be clarified in Section 23-5 — to restrict the complexity of *finite* sets. In the case of  $\text{RCA}_0$ , we will have the infinite computable sets, and the finite  $\Sigma_1^0$  sets. The reader may have the impression that finite sets can all be represented by an integer and that talking about their complexity does not really make sense. We insist once again on the fact that the restrictions of induction must be understood in the light of non-standard models. In such a model, we have an element  $a$  greater than all the integers in the standard part  $\omega$  of the model. From outside the model, there is an infinity of elements smaller than  $a$ , and therefore an uncountable infinity of subsets of elements smaller than  $a$ . In particular, if our non-standard model is countable, there will inevitably be some of these subsets which cannot be “coded” by an element of our model.

### 5.4. Models of $\text{RCA}_0$

Recall that an  $\omega$ -structure is a structure of second-order arithmetic where the first-order part consists of standard integers, equipped with the usual operations. The  $\omega$ -structures are therefore characterized by their the second-order part. The  $\omega$ -models of  $\text{RCA}_0$  admit a very simple characterization in terms of Turing ideals.

**Definition 5.5.** A *Turing ideal* is a class  $\mathcal{I} \subseteq 2^{\mathbb{N}}$  having the following two properties:

- (1) Closure under the Turing reduction:  $\forall X \in \mathcal{I} \forall Y \leq_T X \ Y \in \mathcal{I}$ ;
- (2) Closure under the effective join:  $\forall X \in \mathcal{I} \forall Y \in \mathcal{I} \ X \oplus Y \in \mathcal{I}$ . ◇

**Example 5.6.** The class of computable sets is a Turing ideal. Likewise, for any set  $A$ , the class  $\{X \in 2^{\mathbb{N}} : X \leq_T A\}$  is a Turing ideal. The  $K$ -trivials, seen in Chapter 20, form a Turing ideal.

**Exercise 5.7. (\*)** Show that the class of low sets does not form a Turing ideal.

Note: one could for example use Theorem 10-3.31. ◇

**Exercise 5.8. (\*\*)** Show that the class of computably dominated sets does not form a Turing ideal. ◇

Note that, by Theorem 14-2.7, every countable Turing ideal  $\mathcal{I}$  admits an exact pair, i.e., two sets  $A, B$  such that  $\mathcal{I} = \{X \in 2^{\mathbb{N}} : X \leq_T A \wedge X \leq_T B\}$ . It is easy to show the following equivalence:

**Proposition 5.9 (Friedman).** An  $\omega$ -structure  $\mathcal{M}$  is a model of  $\text{RCA}_0$  iff its second-order part is a Turing ideal. ★

PROOF. Suppose that  $\mathcal{M} \models \text{RCA}_0$ . Let  $\mathcal{I}$  be its second-order part. Let us show that  $\mathcal{I}$  is closed under the Turing reduction. Let  $X \in \mathcal{I}$  and  $Y \leq_T X$ . By Theorem 10-1.2, there exists a  $\Sigma_1^0$  formula  $F(X, x)$  and a  $\Pi_1^0$  formula  $G(X, x)$  such that  $Y = \{n \in \mathbb{N} : F(X, n)\} = \{n \in \mathbb{N} : G(X, n)\}$ . By the  $\Delta_1^0$  comprehension scheme, the set  $Y$  exists, therefore  $Y \in \mathcal{I}$ . Let us show that  $\mathcal{I}$  is closed under join. Let  $X, Y \in \mathcal{I}$ . Let  $F(X, Y, x)$  be the  $\Delta_0^0$  formula defined by

$$\exists y \leq x ((x = \langle 0, y \rangle \wedge y \in X) \vee (x = \langle 1, y \rangle \wedge y \in Y))$$

The formula  $F$  being  $\Delta_0^0$ , by the  $\Delta_1^0$  comprehension scheme, the set  $X \oplus Y = \{n \in \mathbb{N} : F(X, Y, n)\}$  exists, therefore  $X \oplus Y \in \mathcal{I}$ . The class  $\mathcal{I}$  is therefore a Turing ideal.

Conversely, suppose that  $\mathcal{I}$  is a Turing ideal, and let  $\mathcal{M}$  be the  $\omega$ -structure with  $\mathcal{I}$  as second-order part. Let us show that  $\mathcal{M} \models \text{RCA}_0$ . The axioms of Robinson arithmetic and the induction scheme are satisfied because  $\mathcal{M}$  is an  $\omega$ -structure. It suffices to show that  $\mathcal{M}$  satisfies the  $\Delta_1^0$  comprehension scheme. Let  $F(X_1, \dots, X_n, y)$  and  $G(X_1, \dots, X_n, y)$  be formulas respectively  $\Sigma_1^0$  and  $\Pi_1^0$ , with second-order parameters  $X_1, \dots, X_n \in \mathcal{I}$ , such that  $\mathcal{M} \models \forall x (F(X_1, \dots, X_n, x) \leftrightarrow G(X_1, \dots, X_n, x))$ . Since  $\mathcal{M}$  is an  $\omega$ -structure,  $\{m \in \mathbb{N} : F(X_1, \dots, X_n, m)\} = \{m \in \mathbb{N} : G(X_1, \dots, X_n, m)\}$ . Let  $A$  be this set. By Theorem 10-1.2,  $A \leq_T X_1 \oplus \dots \oplus X_n$ . As  $X_1, \dots, X_n \in$

$\mathcal{I}$  and by closure of  $\mathcal{I}$  under the effective join and the Turing reduction,  $A \in \mathcal{I}$ . ■

**Exercise 5.10.** Let  $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \dots$  be a sequence of Turing ideals. Show that  $\mathcal{I} = \bigcup_n \mathcal{I}_n$  is a Turing ideal. ◇

**Exercise 5.11.** Let  $Z_0 \leq_T Z_1 \leq \dots$  be an increasing sequence of sets for the Turing reduction. Show that  $\mathcal{I} = \{X \in 2^{\mathbb{N}} : \exists n \ X \leq_T Z_n\}$  is a Turing ideal. ◇

### 5.5. Mathematics in $\text{RCA}_0$

By Proposition 5.9,  $\text{RCA}_0$  has a minimal  $\omega$ -model for inclusion, namely, the  $\omega$ -structure  $\mathcal{M}_{\text{CALC}}$  whose second-order part is the Turing ideal of computable sets. According to Theorem 9-2.22 of completeness, it therefore suffices, to conclude that a theorem  $T$  is not provable in  $\text{RCA}_0$ , to show that Theorem  $T$  implies the existence of a non-computable set. Here is an example.

**Proposition 5.12.** König's lemma is not provable in  $\text{RCA}_0$ . ★

PROOF. Let us show that  $\mathcal{M}_{\text{CALC}}$  does not satisfy König's lemma. Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable infinite tree having no computable paths. For example, the tree whose paths are all of PA degree.  $T$  being computable,  $T \in \mathcal{M}_{\text{CALC}}$ . However,  $T$  does not have any path in  $\mathcal{M}_{\text{CALC}}$  since  $T$  does not have any computable path. It follows that  $\mathcal{M}_{\text{CALC}}$  is not a model of König's lemma, so that  $\text{RCA}_0$  does not prove König's lemma. ■

Note that, for a proof of separation in  $\omega$ -structures, it is not necessary to worry about the level of induction used in the proof. Indeed, the  $\omega$ -structures satisfy the complete induction scheme. The separation results are therefore often pure computability-theoretic arguments.

On the other hand, showing that  $\text{RCA}_0$  implies a theorem requires much more attention on the complexity of the objects manipulated, and the level of induction used. Recall that the *Intermediate Value Theorem*<sup>2</sup> states that if  $f : \mathbb{R} \rightarrow \mathbb{R}$  is continuous over the interval  $[0, 1]$  with  $f(0) < 0 < f(1)$ , then there exists  $x \in ]0, 1[$  such that  $f(x) = 0$ . We will show that the Intermediate Value Theorem is provable in  $\text{RCA}_0$ .

We refer the reader to Section 4 for some explanations on how to formalize Analysis in  $Z_2$ .

---

<sup>2</sup>Or under this version, Bolzano's theorem

**Proposition 5.13 (Simpson [202]).** The Intermediate Value Theorem is provable in  $\text{RCA}_0$ . ★

PROOF. We will formalize the standard proof of the theorem, namely the proof by dichotomous search. We suppose fixed a sequence  $(I_n)_{n \in \mathbb{N}}$  of all intervals of the form  $]a, b[$  for  $a, b \in \mathbb{Q} \cup \{-\infty, +\infty\}$ . Let  $f : [0, 1] \rightarrow \mathbb{R}$  be a continuous function satisfying  $f(0) < 0 < f(1)$  (the function is given to us via its representation, as explained in Section 4). Suppose that  $f(q) \neq 0$  for any rational number  $q \in \mathbb{Q} \cap ]0, 1[$ , otherwise  $q$  is the desired real number. By the  $\Delta_1^0$  comprehension scheme, sets  $X_-, X_+$  such that  $\bigcup_{n \in X_-} I_n = f^{-1}(]-\infty, 0[)$  and  $\bigcup_{n \in X_+} I_n = f^{-1}(]0, \infty[)$  exist (it suffices to extract the correct information from the representation of our function  $f$ ). Let  $]a_0, b_0[ \supseteq ]a_1, b_1[ \supseteq \dots$  be the sequence of rationally bound intervals, defined by  $]a_0, b_0[ = ]0, 1[$ , and

$$(a_{n+1}, b_{n+1}) = \begin{cases} \left( \frac{a_n + b_n}{2}, b_n \right) & \text{if } \frac{a_n + b_n}{2} \in \bigcup_{n \in X_-} I_n \\ \left( a_n, \frac{a_n + b_n}{2} \right) & \text{if } \frac{a_n + b_n}{2} \in \bigcup_{n \in X_+} I_n \end{cases}$$

By our hypothesis  $f(q) \neq 0$  for any rational number  $q \in \mathbb{Q} \cap ]0, 1[$ , the sequence  $\{(a_n, b_n) : n \in \mathbb{N}\}$  is well defined for all  $n$ . It is therefore provably  $\Delta_1^0(X_- \oplus X_+)$ , and exists by the  $\Delta_1^0$  comprehension scheme. By  $\Sigma_1^0$  induction,  $f(a_n) < 0 < f(b_n)$  for all  $n \in \mathbb{N}$ , and  $|a_n - b_n| = 2^{-n}$ . It follows that  $r = (a_n)_{n \in \mathbb{N}}$  is a real number.

Let us show that  $f(r) = 0$ . Suppose  $f(r) < 0$ . Let  $n \in \mathbb{N}$  be such that  $f(r) < -2^{-n}$ . We then search for an interval  $I_n$  with  $|I_n| < 2^{-n}$  and an integer  $m$  such that  $f(]a_m, b_m[) \subseteq I_n$  and  $a_m < b_{m+1} < b_m$ . The distance between any points  $x, y$  of  $I_n$  and therefore  $f(]a_m, b_m[)$  is bounded by  $2^{-n}$ . This is in particular the case for points  $f(b_{m+1})$  and  $f(r)$ . We then have  $f(b_{m+1}) < 0$ , which contradicts the fact that  $f(b_{m+1}) > 0$ . The case  $f(r) > 0$  is similar. ■

Note that the proof of Proposition 5.13 is not uniform, in the sense that it distinguishes the case of a rational solution or not. We will see in Chapter 24 the Weihrauch reduction which allows to express these uniformity considerations.

We direct the reader interested in a more detailed development of mathematics in  $\text{RCA}_0$ , to the very complete work of Simpson [202], *Subsystems of Second-Order Arithmetic*.

## 6. $\text{ACA}_0$ and the arithmetic hierarchy

As mentioned previously,  $\text{Z}_2$  is not a conservative extension of Peano arithmetic (PA), in the sense that there are statements formulated only in the language of first-order arithmetic, which are not provable in PA, but are provable in  $\text{Z}_2$ . We will now see an important restriction of second-order arithmetic, which gives us more power than  $\text{RCA}_0$  but which, unlike  $\text{Z}_2$ , is conservative over PA.

### Notation

We denote by  $\text{ACA}_0$  the system composed of the axioms of Robinson arithmetic, augmented with the comprehension scheme (9) for the arithmetic formulas, and the axiom of induction (11).

The acronym  $\text{ACA}_0$  stands for “Arithmetic Comprehension Axiom”. The induction axiom and the comprehension scheme for arithmetic formulas allow to prove the induction scheme (10) for arithmetical formulas. The system  $\text{ACA}_0$  therefore implies  $\text{RCA}_0$ . We had shown that  $\text{Z}_2$  is an impredicative system, because the comprehension scheme with arbitrary formulas allows to build sets which depend on themselves (see Section 9-1 and Example 8.5). Restricting the comprehension scheme to arithmetic formulas precisely avoids that, and makes the system predicative. Systems equivalent to  $\text{ACA}_0$  had already been studied by the predicativist movement, in particular by Weyl in 1918 in his book *Das Kontinuum*, i.e., almost 60 years before the start of Reverse Mathematics.

The system  $\text{ACA}_0$  is one of the big systems of Reverse Mathematics which constitute the Big Five phenomenon. Note that it is not necessary to add the comprehension scheme for all the arithmetic formulas to obtain the system  $\text{ACA}_0$ . It suffices to add the scheme for the  $\Sigma_1^0$  formulas, as shown by the following proposition:

**Proposition 6.1 (Simpson [202]).**  $\text{RCA}_0$  proves that  $\text{ACA}_0$  is equivalent to the  $\Sigma_1^0$  comprehension scheme. ★

PROOF. The  $\Sigma_1^0$  formulas being arithmetic,  $\text{ACA}_0$  implies the  $\Sigma_1^0$  comprehension scheme. Conversely, suppose that  $\text{ACA}_0$  proves the  $\Sigma_n^0$  comprehension scheme for a fixed  $n \in \mathbb{N}$ . Let us show that it implies the  $\Sigma_{n+1}^0$  comprehension scheme. Let  $F(x)$  be a  $\Sigma_{n+1}^0$  formula (with free variables).  $F(x)$  can be expressed in the form  $\exists y G(x, y)$ , where  $G(x, y)$  is a  $\Pi_n^0$  formula. By the  $\Sigma_n^0$  comprehension scheme, the set  $Y = \{(x, y) \in \mathbb{N} \times \mathbb{N} : \neg G(x, y)\}$  exists. By the  $\Sigma_1^0$  comprehension scheme parameterized by  $Y$ , the set  $Z = \{x \in \mathbb{N} : \exists y (x, y) \notin Y\}$  exists. In particular,  $Z = \{x \in \mathbb{N} : \exists y G(x, y)\}$  exists. Thus  $\text{ACA}_0$  proves the  $\Sigma_{n+1}^0$  comprehension scheme. ■

Note the induction of the previous proof: we use  $n$  successive applications of  $\Sigma_1^0$  comprehension to show  $\Sigma_n^0$  comprehension. In fact, we do not take into account the number of times an axiom is used in a proof. Thus, as soon as we allow ourselves the  $\Sigma_1^0$  comprehension, the different levels of the arithmetic hierarchy are not distinguishable from the point of view of provability. We will see in Chapter 24 tools allowing to measure in a more precise way the computational power of theorems, by controlling in particular the number of applications of an axiom.

The system  $\text{ACA}_0$  takes us out of computable mathematics. It is in fact equivalent to the existence of the Turing jump of any set.

**Exercise 6.2. (\*)** Show that  $\text{ACA}_0$  is equivalent over  $\text{RCA}_0$  to the statement  $\forall X \exists Y Y = X'$ , where  $Y = X'$  is a notation for  $\forall e (e \in Y \leftrightarrow \Phi_e^X(e) \downarrow)$ .  $\diamond$

### 6.1. Models of $\text{ACA}_0$

As with  $\text{RCA}_0$ , the  $\omega$ -models of  $\text{ACA}_0$  have a nice characterization in terms of Turing ideals.

**Definition 6.3.** A *jump ideal* is a Turing ideal  $\mathcal{I} \subseteq 2^{\mathbb{N}}$  such that for all  $X \in \mathcal{I}$ ,  $X' \in \mathcal{I}$ .  $\diamond$

The following equivalence is an analogue of Proposition 5.9.

**Proposition 6.4 (Simpson [202]).** An  $\omega$ -structure  $\mathcal{M}$  is a model of  $\text{ACA}_0$  iff its second-order part is a jump ideal.  $\star$

**PROOF.** Suppose  $\mathcal{M} \models \text{ACA}_0$ . Let  $\mathcal{I}$  be its second-order part. In particular,  $\mathcal{M} \models \text{RCA}_0$ , so by Proposition 5.9,  $\mathcal{I}$  is a Turing ideal. By Exercise 6.2, for all  $X \in \mathcal{I}$ ,  $X' \in \mathcal{I}$ , so  $\mathcal{I}$  is a jump ideal.

Conversely, suppose that  $\mathcal{I}$  is a jump ideal. In particular,  $\mathcal{I}$  is a Turing ideal, so by Proposition 5.9,  $\mathcal{M} \models \text{RCA}_0$ . Since  $\mathcal{I}$  is a jump ideal, by Exercise 6.2,  $\mathcal{M}$  is a model of  $\text{ACA}_0$ .  $\blacksquare$

In particular,  $\text{ACA}_0$  also has a minimal  $\omega$ -model for inclusion, namely, the  $\omega$ -model whose second-order part is exactly the jump ideal of arithmetic sets. It is easy to see that  $\text{RCA}_0$  does not imply  $\text{ACA}_0$ , considering the  $\omega$ -structure  $\mathcal{M}_{\text{CALC}}$  whose second-order part is the Turing ideal of computable sets.  $\mathcal{M}_{\text{CALC}}$  is a model of  $\text{RCA}_0$ , but the computable sets do not form a jump ideal, so  $\mathcal{M}_{\text{CALC}}$  is not a model of  $\text{ACA}_0$ .

## 6.2. Mathematics in $\text{ACA}_0$

The system  $\text{ACA}_0$  is very powerful, insofar as it suffices to prove an overwhelming majority of theorems. Let us start by showing that  $\text{ACA}_0$  is equivalent over  $\text{RCA}_0$  to a well-known analysis theorem. The *Bolzano-Weierstrass theorem* states that for any sequence  $(x_n)_{n \in \mathbb{N}}$  of points in  $[0, 1]$ , there exists a convergent subsequence, that is to say a strictly increasing function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that the sequence  $(x_{f(n)})_{n \in \mathbb{N}}$  converges to a point.

The Bolzano-Weierstrass theorem states two existences: that of a convergent subsequence, and that of the point of convergence.  $\text{RCA}_0$  proves that the existence of a point of convergence implies that of a convergent subsequence : it suffices to compute from the sequence and the point a subsequence closer and closer to the point of convergence. The reverse is however not true: the existence alone of a convergent subsequence is equivalent to a system named COH (see Section 25-4.1), strictly between  $\text{RCA}_0$  and  $\text{ACA}_0$ .

**Proposition 6.5 (Simpson [202]).**  $\text{ACA}_0$  is equivalent over  $\text{RCA}_0$  to the Bolzano-Weierstrass theorem. ★

**PROOF OF BOLZANO-WEIERSTRASS IN  $\text{ACA}_0$ .** Let  $(x_n)_{n \in \mathbb{N}}$  be a sequence of real numbers in  $[0, 1]$ . Let  $X \in 2^{\mathbb{N}}$  be a representation of this sequence. Using  $X''$ , we define a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $\forall n \forall i |x_{f(n)} - x_{f(n+i)}| \leq 2^{-n}$ . We compute for that, from  $X''$ , a sequence of intervals  $[a_0, b_0] \supseteq [a_1, b_1] \supseteq \dots$  such that each  $[a_n, b_n]$  contains an infinity of  $x_m$ , and such that  $b_n - a_n \leq 2^{-n}$ . We start with  $[a_0, b_0] = [0, 1]$ . Once  $[a_n, b_n]$  has been computed, if there is an infinity of points in  $[a_n, (a_n + b_n)/2]$ , we define  $[a_{n+1}, b_{n+1}] = [a_n, (a_n + b_n)/2]$ . Otherwise we define  $[a_{n+1}, b_{n+1}] = [(a_n + b_n)/2, b_n]$ . Note that the question: “there exists an infinity of points such that  $\dots$ ” is  $\Pi_2^0$  and can therefore be solved using  $X''$ . By using arithmetic induction, we jointly show that the sequence  $(a_n, b_n)_{n \in \mathbb{N}}$  is well defined, and that each  $[a_n, b_n]$  indeed contains an infinity of  $x_m$ . We can then define  $f(n)$  as being the first  $m$  found such that  $x_m \in [a_n, b_n]$ . Once  $f$  defined, we easily construct a rational representation  $(q_n)_{n \in \mathbb{N}}$  of  $\lim_{n \rightarrow \infty} f(n)$  such that  $|q_n - q_{n+i}| \leq 2^n$  for all  $n, i \in \mathbb{N}$ . ■

**PROOF OF  $\text{ACA}_0$  USING THE BOLZANO-WEIERSTRASS THEOREM.** Let  $X$  be a set and let  $(x_n)_{n \in \mathbb{N}}$  be the sequence of real numbers defined by  $x_n = 0.X'[n]$ , where  $X'[n]$  is the approximation of  $X'$  at the computation step  $n$ . Note that the sequence is bounded by 1. According to the Bolzano-Weierstrass theorem, this sequence has a convergent subsequence and a limit, which can only be  $0.X'$ . According to Exercize 6.2, we therefore have  $\text{ACA}_0$ . ■

The two previous proofs clearly illustrate the insensitivity of  $\text{ACA}_0$  to iterations of the Turing jump: we used the double Turing jump to prove the Bolzano-Weierstrass theorem. On the other hand, in the opposite direction, an application of the Bolzano-Weierstrass theorem only allowed to prove the existence of the simple jump. We then only obtain the double jump by applying the theorem twice in a row. It is natural to ask the question of the exact computational power of an application of the Bolzano-Weierstrass theorem. It turns out that its power is exactly that of a PA degree relative to  $\emptyset'$ , as shown in the following exercise.

**Exercise 6.6. (★)** Show that for any sequence  $(x_n)_{n \in \mathbb{N}}$  of real numbers in  $[0, 1]$ , there exists a  $\emptyset'$ -computable tree of which all infinite paths are exactly the points of convergence of subsequences of  $(x_n)_{n \in \mathbb{N}}$ . Finally, show that for any infinite  $\emptyset'$ -computable tree, there exists a sequence  $(x_n)_{n \in \mathbb{N}}$  of points in  $[0, 1]$ , whose points of convergence are exactly the infinite paths of the tree.  $\diamond$

## 7. $\text{WKL}_0$ and the compactness argument

The system  $\text{WKL}_0$  is undoubtedly the most significant system of Reverse Mathematics, which were the first to identify its importance within ordinary mathematics. Formalisms equivalent to the systems  $\text{Z}_2$  and  $\text{ACA}_0$  previously existed, while  $\text{RCA}_0$  captures the already well-established notion of computable mathematics.  $\text{WKL}_0$  constitutes one of the systems of the Big Five phenomenon, and intuitively corresponds to compactness arguments.

### Notation

We denote by  $\text{WKL}$  the statement “Any infinite binary tree admits an infinite path”. We denote by  $\text{WKL}_0$  the system composed of  $\text{RCA}_0$  augmented with the statement  $\text{WKL}$ .

$\text{WKL}_0$  is the acronym of “Weak König’s Lemma”, and corresponds to a well known power in Computability Theory, namely, that of PA degrees. We will see that this system is equivalent to important theorems of Analysis and Logic, in particular the Borel-Lebesgue compactness theorem and the completeness theorem of Gödel.

### 7.1. Models of $\text{WKL}_0$

As for  $\text{RCA}_0$  and  $\text{ACA}_0$ , the system  $\text{WKL}_0$  admits a characterization of its  $\omega$ -models, which follows directly from that of  $\text{RCA}_0$  in terms of Turing ideals (see Proposition 5.9).

**Definition 7.1.** A *Scott ideal* is a Turing ideal  $\mathcal{I} \subseteq 2^{\mathbb{N}}$  such that for every  $X \in \mathcal{I}$  and every infinite binary tree  $X$ -computable  $T \subseteq 2^{<\mathbb{N}}$ , there exists an infinite path of  $T$  belonging to  $\mathcal{I}$ .  $\diamond$

In other words, a Turing ideal  $\mathcal{I}$  is a Scott ideal if for all  $X \in \mathcal{I}$ , there exists a set  $Y \in \mathcal{I}$  of PA degree relative to  $X$ . Dana Scott was the first to study these ideals [193], which unsurprisingly characterize the  $\omega$ -models of  $\text{WKL}_0$ .

**Proposition 7.2.** An  $\omega$ -structure  $\mathcal{M}$  is a model of  $\text{WKL}_0$  iff its second-order part is a Scott ideal.  $\star$

PROOF. Immediate by Proposition 5.9 and the definition of  $\text{WKL}_0$ .  $\blacksquare$

Knowing that there is no computable PA degree, the Turing ideal of computable sets is not a Scott ideal. The  $\omega$ -structure  $\mathcal{M}_{\text{CALC}}$  is therefore a model of  $\text{RCA}_0$  which is not a model of  $\text{WKL}_0$ , which shows that  $\text{RCA}_0$  does not imply  $\text{WKL}_0$ .

Most of the  $\Pi_1^0$  basis theorems are relativizable and allow the construction of Scott ideals with weak computational properties. These ideals corresponding to  $\omega$ -models of  $\text{WKL}_0$ , these computability-theoretic results translate into separation results in Reverse Mathematics.

**Proposition 7.3.** Let  $A_0, A_1, \dots$  be a countable sequence of non-computable sets. There is a Scott ideal  $\mathcal{I}$  which does not contain any of the  $A_i$ .  $\star$

PROOF. We are going to define a sequence of sets  $Z_0 \leq_T Z_1 \leq_T \dots$  such that for all  $n \in \mathbb{N}$ ,

- (1)  $Z_{n+1}$  is of PA degree relative to  $Z_n$ ;
- (2)  $A_0, A_1, \dots$  are not  $Z_n$ -computable.

$Z_0 = \emptyset$ . Suppose we have defined  $Z_n$ . By the cone avoidance basis theorem (see Theorem 8-4.7) iterated, there exists a set  $Z_{n+1}$  of PA degree relative to  $Z_n$ , such that  $A_0, A_1, \dots$  are not  $Z_{n+1}$ -computable. In particular,  $Z_{n+1} \geq_T Z_n$  (see if necessary Exercise 8-7.6). Let  $\mathcal{I} = \{X \in 2^{\mathbb{N}} : \exists n \ X \leq_T Z_n\}$ . By Exercise 5.11,  $\mathcal{I}$  is a Turing ideal. Let us show that  $\mathcal{I}$  is a Scott ideal. Let  $X \in \mathcal{I}$  is  $T \subseteq 2^{<\mathbb{N}}$  an infinite  $X$ -computable tree. In particular, there exists an  $n \in \mathbb{N}$  such that  $X \leq_T Z_n$ , therefore  $Z_{n+1}$  is of PA degree relative to  $X$ . It follows that  $Z_{n+1}$  computes an infinite path of  $T$ , therefore  $\mathcal{I}$  contains an infinite path of  $T$ . The class  $\mathcal{I}$  is therefore a Scott ideal. By construction,  $\mathcal{I}$  does not contain any of the  $A_i$ .  $\blacksquare$

In particular, there is no minimum  $\omega$ -model of  $\text{WKL}_0$  for inclusion, unlike  $\text{RCA}_0$  and  $\text{ACA}_0$ . Indeed, by Proposition 7.3, the intersection of all

Scott's ideal is the Turing ideal of computable sets, which is not a Scott ideal.

We can in particular deduce from Proposition 7.3 that  $\text{WKL}_0$  does not imply  $\text{ACA}_0$  over  $\text{RCA}_0$ . Indeed, every  $\omega$ -model of  $\text{ACA}_0$  contains  $\emptyset'$ , but by Proposition 7.3, there exists an  $\omega$ -model of  $\text{WKL}_0$  not containing  $\emptyset'$ .

**Exercise 7.4.** Show that there exists a Scott ideal  $\mathcal{I}$  containing only sets of computably dominated degree.  $\diamond$

**Exercise 7.5.** Show that there exists a Scott ideal  $\mathcal{I}$  containing only sets of low degree.  $\diamond$

## 7.2. Mathematics in $\text{WKL}_0$

As we mentioned, the system  $\text{WKL}_0$  corresponds intuitively to compactness arguments. The compactness theorem par excellence is the Borel-Lebesgue theorem, which states that for any collection  $(\mathcal{U}_n)_{n \in \mathbb{N}}$  of open classes of  $\mathbb{R}$  such that  $[0, 1] \subseteq \bigcup_n \mathcal{U}_n$ , there exists  $k$  such that  $[0, 1] \subseteq \bigcup_{n \leq k} \mathcal{U}_n$ . We will show that this statement is equivalent to  $\text{WKL}_0$  over  $\text{RCA}_0$ .

Given  $[0, 1] \subseteq \bigcup_n \mathcal{U}_n$ , we can assume without loss of generality that each open class  $\mathcal{U}_n$  consists of a rational interval  $]a_n, b_n[$ , by isolating the intervals of each open class of our union. Let us move on to the proof of the theorem.

**PROOF OF BOREL-LEBESGUE IN  $\text{WKL}_0$ .** For any string  $\sigma \in 2^{<\mathbb{N}}$  we consider the interval  $I_\sigma = [0.\sigma 0^\infty, 0.\sigma 1^\infty] \subseteq [0, 1]$ . Note that the intervals  $I_\sigma$  for  $|\sigma| = n$  form a division of  $[0, 1]$  into  $2^n$  almost disjoint parts (for example  $[0.01010^\infty, 0.01011^\infty]$  and  $[0.01100^\infty, 0.01101^\infty]$  have exactly the point  $0.01011^\infty = 0.01100^\infty$  in common.)

We construct the tree  $T \subseteq 2^{<\mathbb{N}}$  such that  $\sigma \in T$  if  $I_\sigma$  is not included in any open class  $\mathcal{U}_n$ . Note that  $\sigma \in T$  is a  $\Pi_1^0$  predicate. Indeed  $\sigma \notin T$  iff  $[0.\sigma 0^\infty, 0.\sigma 1^\infty] \subseteq ]a_n, b_n[$  for some  $n$ .

So  $T$  is  $\Pi_1^0$  relative to the representation of open classes  $\mathcal{U}_n$  and we can build a  $T$ -computable  $T'$  containing  $T$  such that  $[T] = [T']$  (see Proposition 8-1.10). Let us show that  $[T]$  does not contain any infinite path. Consider  $X \in 2^\mathbb{N}$  and the real number  $r_x = 0.X$ . Formally,  $r_x = \sum_{i \in \mathbb{N}^*} 2^{-i} \times X(i)$ . By the theorem hypothesis, we have  $r_x \in ]a_n, b_n[$  for a certain  $n$ . Let  $\sigma \prec X$  be large enough such that  $r_x \in I_\sigma \subseteq ]a_n, b_n[$ . Then,  $\sigma \notin T$  and therefore  $X \notin [T]$ . So  $T$  does not have any infinite path and neither does  $T'$ . By weak König's lemma, we deduce that  $T'$  is finite and therefore so is  $T$ . So there exists  $n$  such that any interval  $I_\sigma$  for  $\sigma$  of size  $n$  is included in a certain  $\mathcal{U}_n$ . As the number of strings of size  $n$  is finite, we therefore have a certain  $k$  such that  $[0, 1] \subseteq \bigcup_{n \leq k} \mathcal{U}_n$ .

Before concluding, let us remember that  $WKL_0$  still only has induction for  $\Sigma_1^0$  formulas. It is not always easy to understand where induction is used and at what level. The point to which we must pay attention here is the following: for the last step, we use the fact that “for any string  $\sigma$  of size  $n$ , there exists  $m$  for which  $I_\sigma \subseteq \mathcal{U}_m$ ” implies “there exists  $k$  such that for any string  $\sigma$  of size  $n$ , there exists  $m \leq k$  such that  $I_\sigma \subseteq \mathcal{U}_m$ ”. This is a use of the  $\Sigma_1^0$  *collection scheme* that we will formally define in Section 23-3.1, and which is demonstrated from  $\Sigma_1^0$  induction. Close examination will reveal that this scheme is also needed to show  $[T] = [T']$  above. ■

Let us now show that  $WKL_0$  is necessary to prove the Borel-Lebesgue theorem.

**PROOF OF  $WKL_0$  USING BOREL-LEBESGUE.** Let  $T \subseteq 2^{<\mathbb{N}}$  be an infinite binary tree. Let us take again the notation  $I_\sigma = [0.\sigma 0^\infty, 0.\sigma 1^\infty]$  of the previous proof. Suppose that  $T$  does not have any infinite path. Then, for all  $X \in 2^\mathbb{N}$ , there exists  $\sigma \notin T$  such that  $X \in I_\sigma$ . It follows that  $(I_\sigma)_{\sigma \notin T}$  forms a cover of  $2^\mathbb{N}$  by open classes. By the Borel-Lebesgue theorem, there exists a finite set  $F \subseteq 2^{<\mathbb{N}} \setminus T$  such that  $2^\mathbb{N} \subseteq \bigcup_{\sigma \in F} I_\sigma$ . Let  $n$  be the maximum length of strings in  $F$ . The tree  $T$  being infinite, there exists  $\tau \in T$  of length  $n$ . Since  $\bigcup_{\sigma \in F} I_\sigma$  is a cover of  $2^\mathbb{N}$ , there exists  $\sigma \in F$  such that  $I_\tau \subseteq I_\sigma$ . In particular,  $\sigma \preceq \tau$ , but  $\sigma \notin T$  and  $\tau \in T$ , which contradicts the fact that  $T$  is a tree.

Here again, care must be taken not to use more than  $\Sigma_1^0$  induction, which is indeed the case. ■

The system  $WKL_0$  is equivalent to more general compactness theorems on compact metric spaces. In particular,  $WKL_0$  is equivalent to the statement “Any cover of a compact metric space by open classes admits a finite sub-cover”. The system  $WKL_0$  is also equivalent to Proposition 9-2.24: any consistent theory can be extended into a complete and consistent theory. The equivalence between the ability to compute a complete and consistent extension of Peano arithmetic, and the ability to compute a path in any infinite computable tree, illustrates this equivalence. The reader will find a more detailed development of mathematics in  $WKL_0$  in the work of Simpson [202].

### 7.3. $WWKL_0$ and randomness

It is advisable to stop on a particularly important subsystem of  $WKL_0$ , to the point that it would deserve, by its robustness, by its epistemological interpretation, and by the number of theorems which are equivalent to it, to be considered as the sixth honorary member of the Big Five. This is

the restriction of weak König's lemma to trees of positive measure. In this section, we will appeal to notions of Algorithmic Randomness introduced in Part II, and more precisely to the computational properties of Martin-Löf randoms. However, you do not need to understand this section to continue the chapter.

**Definition 7.6.** A tree  $T \subseteq 2^{<\mathbb{N}}$  is of *positive measure* if

$$\liminf_n \frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n} > 0$$

We have seen that there is a correspondence between  $\Pi_1^0$  classes and binary trees. The notion of tree of positive measure describes trees whose corresponding  $\Pi_1^0$  classes are of positive measure.

**Exercise 7.7. (★)** Let  $T \subseteq 2^{<\mathbb{N}}$  be a tree. To show that

$$\lambda([T]) = \liminf_n \frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n} \quad \diamond$$

The notion of positive measure tree turns out to be more practical to handle than the notion of  $\Pi_1^0$  class of positive measure in second-order arithmetic, because it avoids having to formalize the notion of measure which is a function on classes, and therefore of higher order. We will therefore use the formulation in terms of trees of positive measure to define the system  $\text{WWKL}_0$ .

### Notation

**WWKL** is the statement “Any binary tree of positive measure admits an infinite path”.  $\text{WWKL}_0$  is the system  $\text{RCA}_0$  augmented with the statement **WWKL**.

The acronym **WWKL** comes from “Weak weak König's lemma”. It follows directly from weak König's lemma because any binary tree of positive measure is infinite. The  $\Pi_1^0$  classes of positive measure have very strong links with Martin-Löf randomness. In particular, every MLR is, up to a prefix, a member of each  $\Pi_1^0$  class of positive measure, as shows the following lemma.

**Lemma 7.8 (Kučera [127]).** Let  $T \subseteq 2^{<\mathbb{N}}$  be an  $X$ -computable tree such that  $\lambda([T]) > 0$ . Then  $[T]$  contains the suffix of any  $\text{MLR}(X)$  set. That is, if  $Z$  is  $\text{MLR}(X)$  then we have  $Y \in [T]$  and  $\sigma$  such that  $Z = \sigma Y$ .★

**PROOF.** Consider the  $\Sigma_1^0(X)$  class  $\mathcal{U} = 2^{\mathbb{N}} \setminus [T]$ . In particular,  $\lambda(\mathcal{U}) < 1$ . Let  $W$  be a prefix-free  $X$ -c.e. set such that  $\mathcal{U} = \bigcup_{\sigma \in W} [\sigma]$ . We define  $\mathcal{U}^1 = \mathcal{U}$  and  $W^1 = W$ , then by induction on  $n$ , we define the class  $\mathcal{U}^{n+1}$  as being

that described by the set  $W^{n+1} = \{\sigma\tau : \sigma \in W \text{ and } \tau \in W^n\}$ . We then have  $\lambda([W^{n+1}]) = \sum_{\sigma \in W} 2^{-|\sigma|} \lambda([W^n]) = \lambda([W])\lambda([W^n])$ . Then  $\lambda(\mathcal{U}^n) = (\lambda(\mathcal{U}))^n$ . As  $\lambda(\mathcal{U}) < 1$ , then  $f : n \rightarrow \lambda(\mathcal{U}^n)$  tends towards 0 effectively, i.e., bounded by a computable function. In particular,  $\bigcap_n \mathcal{U}^n$  is an  $X$ -test of Martin-Löf. Consider now a set  $Z$  which has no suffix in  $2^{\mathbb{N}} \setminus \mathcal{U}$ . So for any  $\sigma$  such that  $Z = \sigma Y$ , we have  $Y \in \mathcal{U}$ . In particular,  $Z \in \mathcal{U} = \mathcal{U}^1$ . Let  $\sigma \prec Z$  be such that  $\sigma \in W^1$ . Then,  $Z$  truncated by its prefix  $\sigma$ , also belongs to  $\mathcal{U}$ , and therefore by definition  $Z$  also has a prefix in  $W^2$  and therefore belongs to  $\mathcal{U}^2$ . We show by continuing this way that  $Z \in \bigcap_n \mathcal{U}^n$  and therefore that  $Z$  is not  $\text{MLR}(X)$ . ■

Lemma 7.8 allows us to give a nice characterization of  $\omega$ -models of  $\text{WWKL}_0$ .

**Proposition 7.9.** An  $\omega$ -structure  $\mathcal{M}$  is a model of  $\text{WWKL}_0$  iff its second-order part is a Turing ideal  $\mathcal{I}$  such that for all  $X \in \mathcal{I}$ , there exists an  $\text{MLR}(X)$   $Z \in \mathcal{I}$ . ★

PROOF. Let  $\mathcal{M}$  be an  $\omega$ -model of  $\text{WWKL}_0$  and let  $\mathcal{I}$  be its second-order part. In particular,  $\mathcal{M} \models \text{RCA}_0$ , so by Proposition 5.9,  $\mathcal{I}$  is a Turing ideal. Let  $X \in \mathcal{I}$  and  $T \subseteq 2^{<\mathbb{N}}$  be an  $X$ -computable tree of positive measure, such that all infinite paths are  $\text{MLR}(X)$ . For example, we can fix  $c \in \mathbb{N}$  and define  $T = \{\sigma \in 2^{<\mathbb{N}} : \forall \tau \preceq \sigma \ K^X(\tau) \geq |\tau| - c\}$ , where  $K^X(\sigma)$  is the prefix-free Kolmogorov complexity, relativized to  $X$  (see Theorem 18-2.1). As  $\mathcal{M} \models \text{WWKL}$ , there exists  $Z \in [T]$  such that  $Z \in \mathcal{I}$ . In particular,  $Z$  is  $\text{MLR}(X)$ .

Conversely, let  $\mathcal{I}$  be a Turing ideal such that for all  $X \in \mathcal{I}$ , there exists an  $\text{MLR}(X)$   $Y \in \mathcal{I}$ . Let  $\mathcal{M}$ , the  $\omega$ -structure induced by  $\mathcal{I}$ . By Proposition 5.9,  $\mathcal{M} \models \text{RCA}_0$ . Let  $T \subseteq 2^{<\mathbb{N}}$  be a tree of positive measure encoded by a set  $X \in \mathcal{I}$ . By hypothesis, there exists an  $\text{MLR}(X)$   $Z \in \mathcal{I}$ . In particular,  $\lambda([T]) > 0$ , so there is a  $Y \in [T]$  and  $\sigma \in 2^{<\mathbb{N}}$  such that  $Z = \sigma Y$ . As  $Y \leq_T Z$  and  $Z \in \mathcal{I}$ , we have  $Y \in \mathcal{I}$ , so  $\mathcal{M} \models \text{WWKL}$ . ■

The system  $\text{WWKL}_0$  can be considered as capturing mathematics using probabilistic arguments. This view should however be put into perspective, because certain probabilistic arguments require stronger notions of randomness. As no  $\text{MLR}$  is computable, the minimal  $\omega$ -model  $\mathcal{M}_{\text{CALC}}$  of  $\text{RCA}_0$  is not a model of  $\text{WWKL}_0$ , which implies that  $\text{RCA}_0$  does not prove  $\text{WWKL}_0$ .

Let us now show that  $\text{WWKL}_0$  is strictly weaker than  $\text{WKL}_0$ , by building a model of  $\text{WWKL}_0$  which is not a model of  $\text{WKL}_0$  using the tools of Algorithmic Randomness.

**Proposition 7.10.**  $\text{WWKL}_0$  does not imply  $\text{WKL}_0$  over  $\text{RCA}_0$ . ★

PROOF. Let  $Z$  be a 2-random set. By Theorem 19-1.7 and Corollary 19-1.8,  $Z$  is not of PA degree. Let  $Y_0 \oplus Z_0 = Z$ ,  $Y_1 \oplus Z_1 = Z_0$ ,  $Y_2 \oplus Z_2 = Z_1$  and so on. By the relativized van Lambalgen theorem (see Theorem 19-2.5),  $Y_0$  is MLR and  $Z_0$  is  $\text{MLR}(Y_0)$ ,  $Y_1$  is  $\text{MLR}(Y_0)$  and  $Z_2$  is  $\text{MLR}(Y_0 \oplus Y_1)$ , and in general,  $Y_{n+1}$  is  $\text{MLR}(Y_0 \oplus \cdots \oplus Y_n)$  and  $Y_n \leq_T Z$  for all  $n \in \mathbb{N}$ . Let  $\mathcal{I} = \{X \in 2^\mathbb{N} : \exists n \ X \leq_T Y_0 \oplus \cdots \oplus Y_n\}$ . Let  $\mathcal{M}$  be the  $\omega$ -structure induced by  $\mathcal{I}$ . The class  $\mathcal{I}$  is a Turing ideal by Exercize 5.11, so  $\mathcal{M} \models \text{RCA}_0$  by Proposition 5.9. Let  $X \in \mathcal{I}$ , and let  $n \in \mathbb{N}$  be such that  $X \leq_T Y_0 \oplus \cdots \oplus Y_n$ . Since  $Y_{n+1} \in \mathcal{I}$  and  $Y_{n+1}$  is  $\text{MLR}(Y_0 \oplus \cdots \oplus Y_n)$ , then  $Y_{n+1}$  is  $\text{MLR}(X)$ , so  $\mathcal{M} \models \text{WWKL}$ . The  $\omega$ -structure  $\mathcal{M}$  is therefore a model of  $\text{WWKL}_0$ . However,  $\mathcal{I}$  does not contain any set of PA degree, because for all  $X \in \mathcal{I}$ ,  $X \leq_T Z$ , and  $Z$  is not of PA degree. So by Proposition 7.2,  $\mathcal{M}$  is not a model of  $\text{WKL}_0$ . ■

## 8. More powerful systems

We have so far defined five systems:  $\text{RCA}_0$ ,  $\text{WWKL}_0$ ,  $\text{WKL}_0$ ,  $\text{ACA}_0$ , and  $Z_2$ , listed in strictly increasing order of logical strength. The system  $\text{RCA}_0$  captures computable mathematics, which covers a large part of ordinary mathematics, but there are, however, many classical non-computable theorems. The system  $\text{WWKL}_0$  adds the probabilistic arguments, while the system  $\text{WKL}_0$  adds the compactness arguments to the mathematician's toolbox. The system  $\text{ACA}_0$  represents a leap in the logical force of mathematics. It captures predicative mathematics, which corresponds to the overwhelming majority of theorems. Very rare classical theorems escape the power of  $\text{ACA}_0$  and call on  $Z_2$ .

The vast majority of theorems falling below  $\text{ACA}_0$ , weak subsystems of second-order arithmetic, have received special attention. Indeed, the process of Reverse Mathematics is above all to understand trends in mathematics and not to seek exhaustiveness. However, it is worth mentioning some important systems which lie beyond  $\text{ACA}_0$ , and which we will study in more detail in Chapter 31 of Part IV on Higher Computability Theory.

### 8.1. $\text{ACA}'_0$ and the power of induction

We will start with a system that is slightly stronger than  $\text{ACA}_0$ , and which is not considered part of the Big Five phenomenon.

We have seen that  $\text{ACA}_0$  was equivalent to the statement  $\forall X \exists Y \ Y = X'$  (see Exercize 6.2). By iterating the statement of the existence of the Tur-

ing jump, it immediately follows that  $\text{ACA}_0$  is equivalent to the statement  $\forall X \exists Y Y = X^{(n)}$  for all  $n \in \mathbb{N}$ . Here, it is advisable to pay close attention to the quantification of  $n$ , which is an external quantification. The statement  $\forall n \forall X \exists Y Y = X^{(n)}$  uses an induction scheme which goes beyond the scope of  $\text{ACA}_0$ .

Many theorems that we will study in this part are expressed in the form

$$\forall X (F(X) \rightarrow \exists Y G(X, Y)),$$

where  $F$  and  $G$  are arithmetic formulas, possibly with free variables. We can then see a theorem of this type as a mathematical problem, formulated in terms of instances and solutions. We will say that a set  $X$  is an *instance* of the problem if  $F(X)$  is true, and  $Y$  is a *solution* to the instance  $X$  if  $G(X, Y)$  is true. For example, an instance of weak König's lemma is an infinite binary tree, and a solution to this instance is an infinite path. We saw that, by iterations of the Turing jump, it was possible to prove in  $\text{ACA}_0$  the existence of all the sets of the arithmetic hierarchy. On the other hand,  $\text{ACA}_0$  retains a form of weakness, in the sense that it is not able to prove the existence of solutions to a problem, which are arbitrarily high in the arithmetic hierarchy. More precisely the following theorem has been independently proved by Jockusch and Solovay (see Wang [229]):

**Theorem 8.1 (Jockusch and Solovay)**

*Let  $F(X)$  and  $G(X, Y)$  be arithmetic formulas having only these free explicit variables. If*

$$\text{ACA}_0 \vdash \forall X (F(X) \rightarrow \exists Y G(X, Y))$$

*then there exists a  $k \in \mathbb{N}$  such that*

$$\text{ACA}_0 \vdash \forall X (F(X) \rightarrow \exists Y \leq_T X^{(k)} G(X, Y))$$

PROOF COMMUNICATED TO WANG BY JOCKUSCH. Let  $F(X)$  and  $G(X, Y)$  be arithmetic formulas having only these free explicit variables. Let  $C$  be a second-order constant symbol. Let  $G_k(X)$  be the formula  $(F(X) \rightarrow \exists Y \leq_T X^{(k)} G(X, Y))$ . Let  $T$  be the theory in the language  $\mathcal{L}_{\text{PA}}$  augmented with the parameter  $C$ , composed of the axioms of PA, the induction scheme for arithmetic formulas with  $C$  as a parameter, and axioms  $\neg G_k(C)$  for all  $k \in \mathbb{N}$ . Suppose that  $\forall X G_k(X)$  is not provable by  $\text{ACA}_0$  for any  $k \in \mathbb{N}$ . By applying Theorem 9-2.20 and using  $\neg G_{k+1}(X) \rightarrow \neg G_k(X)$ , we deduce that, for all  $k$ , the system  $\text{ACA}_0 \cup \{\neg G_k(C)\}_{i \leq k}$  has a model. By compactness, we deduce that  $\text{ACA}_0 \cup \{\neg G_k(C)\}_{k \in \mathbb{N}}$  has a model. As  $\text{ACA}_0 \vdash \text{PA}$ , the theory  $T$  has a model  $\mathcal{M}$ . More precisely,  $\mathcal{M}$  is specified by a first-order part  $M$  and an interpretation  $C^M \subseteq M$  of the symbol  $C$ . Let  $\mathcal{I} \subseteq \mathcal{P}(M)$  be the class of subsets of  $M$  which can be defined in  $\mathcal{M}$  by an arithmetic

formula, with the only parameter  $C^M$ . In particular,  $C_M \in \mathcal{I}$ . Consider the structure  $\mathcal{M}'$  in the language of  $\mathcal{L}_{Z_2}$ , whose first-order part is  $M$  and whose second-order part is  $\mathcal{I}$ .

Let us show that  $\mathcal{M}' \models \text{ACA}_0$ . As  $\mathcal{M} \models \text{PA}$ , then  $\mathcal{M}'$  satisfies the axioms of Robinson arithmetic. Let  $H(x)$  be an arithmetic formula with parameters in  $\mathcal{M}'$ . Since any parameter of  $H$  is arithmetically definable in  $\mathcal{M}$ , then  $H$  can be transformed into a formula  $\hat{H}$  with the only parameter  $C^M$ , such that  $\{x \in M : M \models \hat{H}(x)\} = \{x \in M : \mathcal{M}' \models H(x)\}$ . As  $\{x \in M : M \models \hat{H}(x)\} \in \mathcal{I}$ , then  $\mathcal{M}'$  satisfies the arithmetic comprehension scheme with parameters. Finally, let us show that  $\mathcal{M}'$  satisfies the induction axiom (11). Let  $X \in \mathcal{I}$ . In particular,  $X$  can be defined by an arithmetic formula with the only parameter  $C^M$ , but  $T$  proves the induction scheme for these formulas, hence the induction axiom (11) is satisfied by  $\mathcal{M}'$ .

So  $\mathcal{M}' \models \text{ACA}_0$ , but as  $\mathcal{M} \models \neg G_k(C^M)$  for all  $k$ , and as any set of  $\mathcal{M}'$  is arithmetically definable in  $C^M$ , then  $\mathcal{M}'$  is a model of  $\neg F(C^M) \vee \forall Y \neg G(C^M, Y)$ . So  $\forall X (F(X) \rightarrow \exists Y G(X, Y))$  is not a theorem of  $\text{ACA}_0$ . ■

It follows directly from Theorem 8.1 that the statement  $\forall n \forall X \exists Y Y = X^{(n)}$  is not provable in  $\text{ACA}_0$ . Let us remember that the  $\omega$ -models of  $\text{ACA}_0$  are closed under Turing jump, and are therefore in particular models of

$$\forall n \forall X \exists Y Y = X^{(n)}.$$

The models of  $\text{ACA}_0$  in which this statement is false will therefore necessarily have a non-standard first-order part, within which the necessary level of induction to show for example  $(\exists X X = \emptyset^{(n)} \rightarrow \exists X X = \emptyset^{(n+1)}) \rightarrow \forall n \exists X X = \emptyset^{(n)}$  is missing. This leads us to define the following system.

### Notation

We denote by  $\text{ACA}'_0$  the system  $\text{RCA}_0$  augmented with the axiom “For any set  $X$  and any integer  $n \in \mathbb{N}$ , there exists a sequence  $Y_0, Y_1, \dots, Y_n$  such that  $Y_0 = X$  and  $Y_{s+1} = Y'_s$ .”

The  $\omega$ -models of  $\text{ACA}_0$  and  $\text{ACA}'_0$  coincide, given that any jump ideal satisfies the above statement. Any proof of separation between  $\text{ACA}_0$  and  $\text{ACA}'_0$  is therefore necessarily done in non-standard models. We will see in Chapter 25 an example of a classical theorem provable in  $\text{ACA}'_0$  but not in  $\text{ACA}_0$ .

## 8.2. $\text{ACA}_0^+$ and the Turing $\omega$ -jump

The system  $\text{ACA}_0^+$  is a reinforcement of  $\text{ACA}_0$  which appears naturally in the analysis of the proofs of countable combinatorial theorems, and in particular Hindman’s theorem 8.2.

### Notation

$\text{ACA}_0^+$  is the system  $\text{RCA}_0$ , augmented with the statement “For any set  $X$ , the Turing  $\omega$ -jump  $X^{(\omega)} = \bigoplus_{n \in \mathbb{N}} X^{(n)}$  of  $X$  exists”.

The system  $\text{ACA}_0^+$  is strictly stronger than  $\text{ACA}_0$ , because the  $\omega$ -structure whose second-order part is composed of the arithmetic sets, is a model of  $\text{ACA}_0$  (and  $\text{ACA}'_0$ ), but does not contain the Turing  $\omega$ -jump of  $\emptyset$ , so is not a model of  $\text{ACA}_0^+$ . This system was introduced by Blass, Hirst, and Simpson [21] to measure the logical power of Hindman’s theorem. It is about a theorem of combinatorial mathematics, and more particularly of Ramsey theory, which will be approached in Chapter 25. However, we will not have the opportunity to come back to Hindman’s theorem, which we will nevertheless state here, for the reader who wishes to measure its difficulty. Given a set  $X \subseteq \mathbb{N}$ , we denote by  $\text{FS}(X)$  the set

$$\left\{ \sum_{n \in F} n : F \subseteq X \text{ finite and non-empty} \right\}$$

#### **Theorem 8.2 (Hindman [85])**

*For any  $k \in \mathbb{N}$  and any function  $f : \mathbb{N} \rightarrow \{0, \dots, k\}$ , there exists an infinite set  $H \subseteq \mathbb{N}$  such that  $|f(\text{FS}(H))| = 1$ .*

Blass, Hirst, and Simpson [21] proved that Hindman’s theorem implies  $\text{ACA}_0$  and is provable in  $\text{ACA}_0^+$ . Hindman’s theorem is a typical example, in Reverse Mathematics, of a theorem for which computational analysis required finding new and more elementary proofs. To date, there are several proofs, including the original one by Hindman [85] which can be formalized in  $\text{ACA}_0^+$ , a short proof due to Baumgartner [11], but whose computational analysis is more complex, a proof of Galvin-Glazer using ultrafilters [38] which are third-order objects, and therefore more difficult to study in Reverse Mathematics, and more recently a simple proof due to Towsner [223] but whose logical complexity is the same as the original proof of Hindman. The following question remains one of the biggest open questions in Reverse Mathematics:

**Question 8.3.** What is the exact computational power of Hindman’s theorem? ★

It is not yet known whether Hindman’s theorem is equivalent over  $\text{RCA}_0$  to  $\text{ACA}_0$ , to  $\text{ACA}_0^+$ , or even strictly between the two systems. We do not know for the moment any classical theorem equivalent to the system  $\text{ACA}_0^+$ , which relativizes the importance of this system.

### 8.3. $\text{ATR}_0$ and transfinite induction

The systems  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$  are respectively the fourth and fifth big systems of the Big Five phenomenon studied in Reverse Mathematics. Their study as well as that of their models calls for more powerful tools than those of Classical Computability Theory, namely Higher Computability Theory, which is a field of study in its own right, and which we will present in Part IV. The systems  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$  will be studied in more detail in Chapter 31. However, we give a quick overview in this section to complete the overview of Reverse Mathematics systems.

The system  $\text{ATR}_0$  informally affirms the existence of the Turing  $\alpha$ -jump, for an ordinal  $\alpha$  in the model considered. More precisely, a *well ordered* set is a totally ordered set  $(A, <_A)$  with no infinite decreasing sequence (this notion will be formally presented and studied in great detail in Chapter 27). Let  $\theta(x, X)$  be an arithmetic formula, containing in particular the free variables  $x$  and  $X$ , and potentially other free variables. This formula induces an operator  $\Theta : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  on the sets defined by  $\Theta(X) = \{n \in \mathbb{N} : \theta(n, X)\}$ .

**Definition 8.4.** The *transfinite recursion scheme* states, for any arithmetic formula  $\theta(x, X)$  with free variables, and any well-ordered set  $(A, <_A)$ , the existence of a defined set  $Y = \bigoplus_{a \in A} Y_a$  for all  $a \in A$  by

$$Y_a = \Theta \left( \bigoplus_{b <_A a} Y_b \right)$$

For example, if we consider the formula  $\theta(x, X) = \Phi_x^X(x) \downarrow$  and the well-ordered set  $(\mathbb{N}, <_{\mathbb{N}})$ , then the operator  $\Theta$  is the Turing jump defined by  $\Theta(X) = X'$ . We have  $Y_0 = \Theta(\emptyset) = \emptyset'$ ,  $Y_1 = \Theta(Y_0) = \emptyset''$ ,  $Y_2 = \Theta(Y_0 \oplus Y_1) = (\emptyset' \oplus \emptyset'')$ , and so on. The resulting set  $Y$  is of the same degree as the Turing  $\omega$ -jump of  $\emptyset$ .

#### Notation

$\text{ATR}_0$  is the system  $\text{RCA}_0$ , augmented with the transfinite recursion scheme.

From the previous example, we see that  $\text{ATR}_0$  implies  $\text{ACA}_0^+$  and therefore  $\text{ACA}_0$ . We will see in Chapter 31 that  $\text{ATR}_0$  is a strictly more powerful system.

### 8.4. $\Pi_1^1\text{-CA}_0$ and impredicativity

It is possible to extend the classification of formulas beyond the arithmetic hierarchy, based on the alternation of quantifiers on sets. We then obtain

the *analytical hierarchy*. We will only define its lowest level. A second-order arithmetic formula is  $\Pi_1^1$  (resp.  $\Sigma_1^1$ ) if it is of the form  $\forall X F(X)$  (resp.  $\exists X F(X)$ ) where  $F$  is an arithmetic formula.

### Notation

$\Pi_1^1\text{-CA}_0$  is the system  $\text{RCA}_0$ , augmented with the comprehension scheme (9) for the  $\Pi_1^1$  formulas.

The system  $\Pi_1^1\text{-CA}_0$  is strictly more powerful than  $\text{ATR}_0$ , and very rare theorems are equivalent to it. Let us mention the Cantor-Bendixson theorem, which states that any uncountable closed class of  $\mathbb{R}^n$  is the union of a perfect closed class and a finite or countable class. In particular, the Cantor-Bendixson theorem asserts that the continuum hypothesis is provable for closed classes of  $\mathbb{R}^n$ . We will see the proof of a certain form of this theorem within Baire space  $\mathbb{N}^{\mathbb{N}}$  with Theorem 30-3.2.

Unlike the system  $\text{ATR}_0$ , the system  $\Pi_1^1\text{-CA}_0$  is fundamentally impredicative. Let us recall that an unpredicative definition is in substance a circular definition in which the object which is defined is itself capable of being used in the definition. Consider the following example.

**Example 8.5.** The following sets are defined in an impredicative way:

- (1)  $A = \{n \in \mathbb{N} : \forall X \ n \notin X\}$
- (2)  $W = \{e \in \mathbb{N} : W_e \text{ enumerates a tree } T \subseteq \mathbb{N}^{<\mathbb{N}} \text{ such that } \forall Y \ Y \notin [T]\}$

The sets  $A$  and  $W$  are defined via a quantification on all the sets, and are therefore defined in terms of themselves.

The set  $A$  of the previous example can of course be defined differently —  $A = \emptyset$  — but we will see in Section 29-3 that this is not the case for the set  $W$ . Note that the impredicativity of  $Z_2$  only calls on the comprehension scheme (9) for a  $\Pi_1^1$  formula, which makes  $\Pi_1^1\text{-CA}_0$  an impredicative system.

## 8.5. Summary

In this chapter, we have presented a strictly increasing hierarchy of subsystems of second-order arithmetic:

$$\text{RCA}_0 < \text{WKL}_0 < \text{ACA}_0 < \text{ACA}'_0 < \text{ACA}^+_0 < \text{ATR}_0 < \Pi_1^1\text{-CA}_0 < Z_2$$

The systems  $\text{RCA}_0$ ,  $\text{WKL}_0$ ,  $\text{ACA}_0$ ,  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$  form the Big Five. Most ordinary theorems are either provable in  $\text{RCA}_0$ , or equivalent to one

of the four other systems modulo  $\text{RCA}_0$ . Among these theorems, the vast majority of them are provable in  $\text{ACA}_0$ , or even  $\text{WKL}_0$ .

The study of the subsystems of second-order arithmetic is not, however, confined to this linearly ordered minimalist hierarchy. We will study in the following chapters several subsystems of intermediate power, which appear naturally in the study of mathematics. Chapter 25 in particular will focus on the meta-mathematical study of Ramsey's theorem, which has profoundly changed the face of Reverse Mathematics.

# Chapter 23

## Induction and conservation

In this chapter, we focus on the possibilities and limitations of the system  $\text{RCA}_0$  with regard to first-order statements. Recall that  $\text{RCA}_0$  restricts the induction scheme to  $\Sigma_1^0$  formulas.

Restricting induction to capture computable mathematics may seem surprising at first glance, as it is a principle governing the behavior of integers and not sets of integers. However, we will see that there is a link between the induction scheme and the bounded comprehension scheme, in other words, the existence of finite sets in the model. The restriction of induction is part of the program of Reverse Mathematics, which aims to find the optimal axioms to prove ordinary theorems. It is therefore reasonable to consider induction as a resource that one seeks to minimize.

Since the standard integers satisfy the induction scheme for all formulas, it is useful to fully understand the limits of  $\text{RCA}_0$  to base one's intuition on non-standard models, which, let us recall, contain elements larger than all standard integers.

### Notation: $\omega$ is the new $\mathbb{N}$

Within the framework of the study of potentially non-standard models, it is customary to denote the standard integers (those which we consider to be “true integers”, and denoted  $\mathbb{N}$  until then in this book) by the letter  $\omega$ . Our usual notation  $\mathbb{N}$  will then refer, within this chapter, to the syntactic set of integers: for example the sentence “ $\forall n \in \mathbb{N} F(n)$ ” means that any integer from the point of view of the considered model satisfies the formula  $F$ . The notation  $\mathbb{N}$  is then simply a syntactic symbol which is interpreted by  $M$  within a model  $\mathcal{M} = (M, S, +, \times, <, 0, 1)$ .

## 1. $\text{RCA}_0$ -provably computable functions

In order to ensure the validity of a proof in  $\text{RCA}_0$ , it is necessary in particular to check that only the  $\Sigma_1^0$  induction scheme is used. This is a restriction which should not be taken lightly, and which should be kept in mind, in particular before applying the comprehension scheme for  $\Delta_1^0$  formulas, which we recall here. For any  $\Sigma_1^0$  formula  $F(x)$  and any  $\Pi_1^0$  formula  $G(x)$ , we have:

$$(\forall x (F(x) \leftrightarrow G(x))) \rightarrow \exists X \forall y (y \in X \leftrightarrow F(y)) \quad (12)$$

In particular, it will be necessary to prove  $(\forall x (F(x) \leftrightarrow G(x)))$  within  $\text{RCA}_0$  before being able to apply the axiom of comprehension. It is in fact conceivable that some sets are computable, that is to say  $\Delta_1^0$ , without however being able to prove it within  $\text{RCA}_0$ . It therefore appears important to carry out a detailed study of the arithmetic statements that it is possible to prove in  $\text{RCA}_0$ , and in particular to understand which functions are provably computable there.

Recall that a function  $f : \omega \rightarrow \omega$  is encoded in the form of a set of integers by its graph  $G_f = \{\langle m, n \rangle : f(m) = n\}$ . A function  $f : \omega \rightarrow \omega$  is computable if and only if its graph is definable by a  $\Sigma_1$  formula of arithmetic (via the ingenious coding of Gödel presented in the proof of Theorem 9-3.4), in other words, there is a  $\Sigma_1$  formula  $F(x, y)$  such that  $G_f = \{\langle m, n \rangle : \omega \models F(m, n)\}$ , where  $\omega$  denotes a standard model of PA.

### Remark

The graphs of computable functions correspond to  $\Sigma_1$  formulas (in particular without second-order parameters). For the  $X$ -computable functions with oracle  $X$ , the correspondence still holds, but for a  $\Sigma_1^0$  formula using the parameter  $X$ .

**Definition 1.1.** We say that a formula  $F(x, y)$  of (first or second order) arithmetic is *functional* if  $\omega \models \forall x \exists! y F(x, y)$ . ◇

Recall that the notation  $\exists! y$  means “there is a unique  $y$ ”. Note that if a  $\Sigma_1^0$  formula  $F(x, y)$  is functional, the function  $f : \omega \rightarrow \omega$  that it defines is indeed of  $\Delta_1^0$  graph, because for any (total) function  $f : \omega \rightarrow \omega$ , we have  $f(x) = y$  iff  $f(x) \neq z$  for all  $z \neq y$ . Formally, the formula  $F(x, y)$  is expressed in a  $\Pi_1^0$  way via the formula  $\forall z \neq y \neg F(x, z)$ . Be careful, however, it is not because  $F(x, y)$  is functional that the system  $\text{RCA}_0$  can prove it; it could be that in some models the relation  $F(x, y)$  will not be functional because of non standard elements  $x$ . This leads us to the following definition.

**Definition 1.2.** A formula  $F(x, y)$  of (first or second order) arithmetic is *T-provably functional* for a theory  $T$  if  $T \vdash \forall x \exists! y F(x, y)$ , that is to say:

$$T \vdash \forall x \exists y F(x, y) \quad \text{and} \quad T \vdash \forall x \forall y_0 \forall y_1 (F(x, y_0) \wedge F(x, y_1) \rightarrow y_0 = y_1)$$

A function  $f : \omega \rightarrow \omega$  is *T-provably total* if there exists a  $T$ -provably functional formula  $F(x, y)$  such that  $F(n, f(n))$  is true for all  $n \in \omega$ . A function  $f : \omega \rightarrow \omega$  is *T-provably computable* if it is  $T$ -provably total via a  $\Sigma_1$  formula.  $\diamond$

### $\Sigma_1$ vs $\Sigma_1^0$ formulas

Note that if  $T \vdash \forall x \exists! y F(x, y)$ , where  $F$  is a  $\Sigma_1^0$  formula with a free variable  $Z$ , this is equivalent to a proof of  $T \vdash \forall Z \forall x \exists! y F(x, y)$ . The corresponding function will thus be a provably total functional on all oracles  $Z$ .

The above definition can easily be generalized to functions with several parameters. The main question of this chapter is therefore to characterize the  $\text{RCA}_0$ -provably computable functions. In particular, we will show:

#### **Theorem 1.3**

*The primitive recursive functions are  $\text{RCA}_0$ -provably computable.*

This is a theorem which will allow us to abstract a little from the axiom system  $\text{RCA}_0$ . Let us remember the characterization of primitive recursive functions as those which can be computed by structured programs using **for** loops but no **while** loops (see Theorem 6-3.22). Thanks to this, we can keep a certain lightness in the formalism, while remaining rigorous: to show that the recourse to a computable function is valid in  $\text{RCA}_0$ , it suffices to show that it is primitive recursive.

Theorem 1.3 actually goes further, and we have the following superb characterization.

#### **Theorem 1.4 (Parikh, see Hajek and Pudlak [80])**

*The  $\text{RCA}_0$ -provably computable functions are exactly the primitive recursive functions.*

The converse is unfortunately much more complex to prove, and goes beyond the scope of this work; however, we will give in the second part of this chapter (see Section 7) some elements of the proof.

We therefore devote the beginning of this chapter to the proof of Theorem 1.3. We are in fact going to show a slightly stronger result: the

primitive recursive functions definable using any oracle  $Z$  are provably computable, whatever the oracle in the theory  $\text{RCA}_0$ .

**Definition 1.5.** The class of  $Z$ -primitive recursive functions is the smallest class of functions containing the base functions (successor, projection, constants), the characteristic function of  $Z$ , and which is closed under composition and the primitive recursion scheme.  $\diamond$

In the case of a primitive recursive function without oracle, the first-order part of  $\text{RCA}_0$  — denoted  $\Sigma_1\text{-PA}$ , i.e., Robinson arithmetic and the induction scheme for  $\Sigma_1$  formulas — suffices to show that it is computable. This is not something very surprising, and we will see in the second part of this chapter that any first-order statement provable in  $\text{RCA}_0$  is already provable in  $\Sigma_1\text{-PA}$ .

## 2. Weak PA subsystems

In order to show Theorem 1.3, we will proceed to an encoding of the primitive recursive functions (possibly with oracle) by functional  $\Sigma_1^0$  formulas, similar to that which was presented in the proof of Theorem 9-3.4, except that this time, we will have to make sure that everything we do is formalized in  $\text{RCA}_0$ . The tedious nature of the developments to come will be compensated by a detailed understanding of the main axioms of weak subsystems of arithmetic, carried out in particular in Section 3.

Even if we are studying first-order theorems here, we are placing ourselves within the framework of second-order arithmetic. The different formulas used are likely to include free second-order variables. For the majority of the proofs that follows, this does not matter.

### 2.1. Robinson arithmetic

Robinson arithmetic (denoted  $\mathbf{Q}$ ) is, let us remember, a fragment of Peano arithmetic from which the induction scheme has been removed. Its axioms are detailed in Section 9-2.3. The axioms of Robinson arithmetic alone form a very weak system, within which hardly any statement of the type  $\forall x F(x)$  is provable. Here is for example a non-standard model of  $\mathbf{Q}$  within which the statement  $\forall x \neg(x < x)$  is false.

**Example 2.1.** Let  $\mathcal{M} = (\omega \cup \{\infty\}, +, \times, <, 0, 1)$  be the structure, where  $+$ ,  $\times$  and  $<$  have their usual meaning on  $\omega$ , and where

- $n + \infty = \infty + n = \infty$  for  $n \in \omega \cup \{\infty\}$

- $n \times \infty = \infty \times n = \infty$  for  $n \in (\omega \setminus 0) \cup \{\infty\}$
- $0 \times \infty = \infty \times 0 = 0$

$\mathcal{M}$  is a model of  $\mathbf{Q}$ . We leave it to the reader to verify that the axioms of  $\mathbf{Q}$  are all satisfied in this model.

Note that  $\mathbf{Q}$  will still be able to prove statements of the type  $\neg(\dot{n} < \dot{n})$  where  $\dot{n}$  is the term  $(\dots(1+1)+\dots+1)$  where 1 is repeated  $n$  times, but without being able to do a generalization to all integers.

**Exercise 2.2.** ( $\star$ ) Show that the statements  $\forall x, y \ x+y = y+x$  and  $\forall x, y \ x+y = y+x$  are not provable in  $\mathbf{Q}$ .  $\diamond$

In order to be able to use even the most basic facts about integers, such as the commutativity of addition or multiplication, it then appears necessary to use a minimum of induction.

## 2.2. The open induction scheme

Let us first see the most basic possible induction scheme, which allows us to demonstrate elementary facts about the integers.

### Notation

We denote by  $\mathbf{l}_{\text{open}}$  the induction scheme  $(F(0) \wedge \forall x (F(x) \rightarrow F(x+1))) \rightarrow \forall x F(x)$  restricted to quantifier-free formulas.

**Proposition 2.3.** The commutativity, associativity and injectivity of addition are provable in  $\mathbf{Q} + \mathbf{l}_{\text{open}}$ . Formally

$$\begin{aligned} \mathbf{Q} + \mathbf{l}_{\text{open}} &\vdash \forall x, y \ x + y = y + x \\ \mathbf{Q} + \mathbf{l}_{\text{open}} &\vdash \forall x, y, z \ (x + y) + z = x + (y + z) \\ \mathbf{Q} + \mathbf{l}_{\text{open}} &\vdash \forall x, y, z \ x + y = x + z \rightarrow y = z \end{aligned} \quad \star$$

**PROOF.** Let us start with commutativity. Let us first show  $0 + x = x$  for all  $x$  ( $F_1$ ). We have  $0+0 = 0$  by (4) of  $\mathbf{Q}$ . Suppose  $0+x = x$ . Then,  $0+(x+1) = (0+x)+1$  by (5) of  $\mathbf{Q}$ . By the induction hypothesis  $(0+x)+1 = x+1$ . By  $\mathbf{l}_{\text{open}}$ , we get  $0+x = x$  for all  $x$ . Let us now show  $(x+1)+y = (x+y)+1$  for all  $x, y$  ( $F_2$ ). We have  $(x+1)+0 = x+1 = (x+0)+1$  by (4) of  $\mathbf{l}_{\text{open}}$ . Suppose  $(x+1)+y = (x+y)+1$ . Then,  $(x+1)+(y+1) = ((x+1)+y)+1 = ((x+y)+1)+1 = (x+(y+1))+1$  by (5) of  $\mathbf{l}_{\text{open}}$  and by the induction hypothesis. By  $\mathbf{l}_{\text{open}}$ , we get  $(x+1)+y = (x+y)+1$  for all  $x, y$ . Let us finally show the commutativity of addition. We have  $x+0 = 0+x$  for all  $x$ , by (4) of  $\mathbf{Q}$  and ( $F_1$ ). Suppose  $x+y = y+x$ . Then,  $x+(y+1) = (x+y)+1$

by (5) of **Q**. By induction hypothesis,  $(x + y) + 1 = (y + x) + 1$ . By  $(F_2)$   $(y + x) + 1 = (y + 1) + x$ . By  $I_{\text{open}}$ , we get  $x + y = y + x$  for all  $x, y$ .

Let's move on to associativity. We have  $(x + y) + 0 = x + y = x + (y + 0)$ , by (4) of **Q**. Suppose  $(x + y) + z = x + (y + z)$ . Then,  $(x + y) + (z + 1) = ((x + y) + z) + 1 = (x + (y + z)) + 1 = x + ((y + z) + 1) = x + (y + (z + 1))$ , by (5) of **Q** and by induction hypothesis. By  $I_{\text{open}}$ , we get the associativity for all  $x, y, z$ .

Finally, let's show injectivity. As  $0 + x = x$  for all  $x$ , then  $0 + y = 0 + z \rightarrow y = z$ . Suppose  $x + y = x + z \rightarrow y = z$ . Then suppose  $(x + 1) + y = (x + 1) + z$ . Then,  $(x + y) + 1 = (x + z) + 1$  by associativity and commutativity of addition. By (2) of **Q**, we therefore have  $(x + y) = (x + z)$ . So  $y = z$  by induction hypothesis, which implies  $y + 1 = z + 1$ . By  $I_{\text{open}}$ , we get the injectivity for all  $x, y, z$ . ■

**Exercise 2.4.** (★) Show that the following statements are provable in  $\mathbf{Q} + I_{\text{open}}$ :

- (1) The commutativity of multiplication:  $x \times y = y \times x$
- (2) The distributivity of multiplication over addition:  $x \times (y + z) = x \times y + x \times z$
- (3) The associativity of multiplication:  $(x \times y) \times z = x \times (y \times z)$  ◇

The injectivity of the multiplication will be easier to show using  $<$ .

#### Notation

We write  $x \leq y$  as a shorthand for the statement  $x = y \vee x < y$ .

**Lemma 2.5.** The following statements are provable in  $\mathbf{Q} + I_{\text{open}}$ :

- (1) Total order:  $x = y \vee x < y \vee y < x$ . Each of these cases is furthermore mutually exclusive.
- (2) Transitivity of the order:  $(x < y \wedge y \leq z) \rightarrow x < z$
- (3) Addition of inequalities:  $(x_1 < y_1 \wedge x_2 \leq y_2) \rightarrow x_1 + x_2 < y_1 + y_2$
- (4) Growth of the multiplication:  $x < x \times y$  for  $x \neq 0$  and  $y \neq 0, 1$
- (5) Injectivity of the multiplication:  $(x \times y = x \times z \wedge x \neq 0) \rightarrow y = z$  ★

**PROOF.** (1) Let's show  $x = y \vee x < y \vee y < x$ . Recall that, by (8) of **Q**, we have  $x < y$  iff  $x + z = y$  for  $z \neq 0$ . For any  $x = 0$  and  $y$ , either  $y = 0$  in which case  $x = y$ , or  $y \neq 0$  in which case  $x + y = y$  by (4) of **Q**, and the

commutativity of addition, and therefore  $x < y$ . Suppose  $x = y \vee x < y \vee y < x$  and show  $(x+1) = y \vee (x+1) < y \vee y < (x+1)$ . If  $x = y$  then  $y+1 = x+1$  and therefore  $y < x+1$ . If  $x < y$ , then  $x+z = y$  for  $z \neq 0$ . If  $z = 1$ , then  $x+1 = y$ . If  $z \neq 1$ , then  $z = z' + 1$  by (3) of  $\mathbf{Q}$ , for  $z' \neq 0$  because  $z \neq 1$ . In particular,  $x + (z' + 1) = y$ , and therefore  $(x+1) + z' = y$  by commutativity and associativity of the addition. So  $x+1 < y$  by (8) of  $\mathbf{Q}$ . Finally, if  $y < x$ , then  $y+z = x$  for  $z \neq 0$ . So  $(y+z) + 1 = y + (z+1) = x+1$  by associativity of the addition. So  $y < x+1$  by (8) of  $\mathbf{Q}$ . By  $\mathbf{l}_{\text{open}}$ , we therefore have  $x = y \vee x < y \vee y < x$  for all  $x, y$ .

Let us now show that each case is mutually exclusive. If  $x = y$ , we cannot have  $x+z = y$  or  $y+z = x$  for  $z \neq 0$ . Indeed by injectivity of the addition, that would give  $z = 0$ . If  $x < y$  then  $x+z = y$  for  $z \neq 0$ . Suppose  $y \leq x$ . Then,  $y+z' = x$  for  $z'$ , which gives  $y+z'+z = y = y+0$  and therefore  $z'+z = 0$  by injectivity of the addition, which is absurd.

- (2) If  $x < y$  and  $y \leq z$ , then  $x+a = y$  for  $a \neq 0$  and  $y+b = z$  for  $b$ . So  $x+(a+b) = (x+a)+b = z$  by associativity of the addition. So  $x < z$ .
- (3) If  $x_1 < y_1$  and  $x_2 \leq y_2$ , then  $x_1 + z_1 = y_1$  for  $z_1 \neq 0$  and  $x_2 + z_2 = y_2$ . So  $(x_1+x_2) + (z_1+z_2) = (x_1+z_1) + (x_2+z_2) = y_1+y_2$  by associativity and commutativity of addition. So  $x_1 + x_2 < y_1 + y_2$ .
- (4) Let us show  $x < x \times y$  for all  $x \neq 0$  and all  $y \neq 0, 1$ . For  $y = 2$ , we have  $x < x+x$  by (8) of  $\mathbf{Q}$ . Now suppose  $x < x \times y$ . Then,  $x < x \times y < x \times y + x = x \times (y+1)$ . By  $\mathbf{l}_{\text{open}}$ , we have  $x < x \times y$  for all  $x \neq 0$  and all  $y \neq 0, 1$ .
- (5) Let us show the contrapositive of  $(x \times y = x \times z \wedge x \neq 0) \rightarrow y = z$ . Suppose  $y \neq z$ . By (1) of this lemma,  $y < z$  or  $z < y$ . Suppose  $y < z$ , the other case being symmetrical. Let us show by induction that, for all  $x \neq 0$ ,  $x \times y < x \times z$ . For  $x = 1$ , we have  $x \times y = y < z = x \times z$ . Let  $x \neq 0$  be such that  $y < z$  implies  $x \times y < x \times z$  for all  $y, z$ . Then,  $(x+1) \times y = x \times y + y$ . But  $x \times y < x \times z$  by the induction hypothesis and  $y < z$ . So, by (3) of this lemma,  $(x+1) \times y = x \times y + y < x \times z + z = (x+1) \times z$ . By  $\mathbf{l}_{\text{open}}$ , we have  $y < z \rightarrow x \times y < x \times z$  for all  $x \neq 0$  and all  $y, z$ . By (1) of this lemma, we have  $x \times y < x \times z$  implies  $x \times y \neq x \times z$ . ■

**Exercise 2.6.** Show the multiplication of inequalities in  $\mathbf{Q} + \mathbf{l}_{\text{open}}$ :  $(x < y \wedge z \neq 0) \rightarrow x \times z < y \times z$ . ◇

### 2.3. The $\Delta_0^0$ induction scheme

Apart from basic facts about integers, the system  $\mathsf{l}_{\text{open}}$  remains relatively weak. Defining some simple functions will require the use of bounded quantifiers. We are introducing a slightly more powerful system for this.

#### Notation

We denote by  $\mathsf{I}\Delta_0^0$  the induction scheme  $(F(0) \wedge \forall x (F(x) \rightarrow F(x+1))) \rightarrow \forall x F(x)$  restricted to  $\Delta_0^0$  formulas.

#### Notation

We write  $x|y$  to signify that  $x$  divides  $y$ , which is formalized by the following  $\Delta_0$  statement:  $\exists m \leq y \ x \times m = y$ .

**Exercise 2.7. (★)** Show the Euclidean division in  $\mathsf{Q} + \mathsf{I}\Delta_0^0$ : for all  $a, b$  with  $b \neq 0$ , there exists a unique pair  $q, r$  with  $0 \leq r < b$ , such that  $a = qb + r$ . ◇

**Lemma 2.8.** The following statements are provable in  $\mathsf{Q} + \mathsf{I}\Delta_0^0$ :

- (1)  $z|x \wedge z|y \rightarrow z|(x+y)$
- (2)  $2|x \vee 2|(x+1)$  ★

PROOF. (1) If  $z \times m_1 = x$  and  $z \times m_2 = y$ , then  $z \times (m_1 + m_2) = x + y$ .

- (2)  $2 \times 0 = 0$ , so  $2|0$ . Suppose  $2|x \vee 2|(x+1)$ . If  $2|(x+1)$ , then  $2|(x+1) \vee 2|(x+2)$ . If  $2|x$ , as also  $2|2$ , then  $2|(x+2)$ . By  $\mathsf{I}\Delta_0^0$ , we conclude  $2|x \vee 2|(x+1)$  for all  $x$ . ■

**Lemma 2.9.** The function  $\div : \omega^2 \rightarrow \omega$  defined by  $a \div b$  equal to  $z$  such that  $b + z = a$  if  $b < a$ , and by  $a \div b = 0$  if not, is provably total in  $\mathsf{Q} + \mathsf{I}\Delta_0^0$  via a  $\Delta_0$  formula. ★

PROOF. The  $\Delta_0$  formula which defines  $a \div b$  is  $F(a, b, r) \equiv (b < a \wedge b + r = a) \vee r = 0$ . Let's show existence. For any  $a, b$ , if  $b < a$ , then by (8) of  $\mathsf{Q}$ , there exists  $r \neq 0$  such that  $b + r = a$ . If  $b \geq a$ , then the formula holds for  $r = 0$ . Let's show the uniqueness. Suppose  $F(a, b, r_1)$  and  $F(a, b, r_2)$  verified. If  $\neg(b < a)$ , then  $r_1 = r_2 = 0$ . If  $b < a$ , then  $b + r_1 = a$  and  $b + r_2 = a$ , therefore  $b + r_1 = b + r_2$  and therefore  $r_1 = r_2$ , by injectivity of the addition. ■

**Lemma 2.10.** The following statements are provable in  $\mathbf{Q} + \mathbf{I}\Delta_0^0$ :

- (1)  $x \times (a \dot{+} b) = x \times a \dot{+} x \times b$
- (2) If  $z|x$  and  $z|y$ , then  $z|(x \dot{+} y)$  ★

PROOF. (1) Suppose  $a \leq b$ . Then,  $x \times a \leq x \times b$  by Exercise 2.6. In particular,  $x \times (a \dot{+} b) = 0 = x \times a \dot{+} x \times b$ . Now, suppose  $b < a$ . Then,  $x \times b \leq x \times a$  by Exercise 2.6. Let  $r_0$  be such that  $r_0 + b = a$ . Let  $r_1$  be such that  $r_1 + x \times b = x \times a$ . Then,  $x \times r_0 + x \times b = x \times a$ , which gives us  $x \times r_0 = r_1$ , which gives well  $x \times (a \dot{+} b) = x \times a \dot{+} x \times b$ .

(2) Let  $m_1, m_2$  be such that  $z \times m_1 = x$  and  $z \times m_2 = y$ . By (1),  $z \times m_1 \dot{+} z \times m_2 = x \dot{+} y$ . So  $z \times (m_1 \dot{+} m_2) = x \dot{+} y$ . So  $z|(x \dot{+} y)$ . ■

**Lemma 2.11.** Cantor's bijection  $\langle \rangle : \omega^2 \rightarrow \omega$  and its projections are provably total in  $\mathbf{Q} + \mathbf{I}\Delta_0^0$  via a  $\Delta_0$  formula. ★

PROOF. The pairing function is defined by the formula  $F(x, y, r) \equiv r \times 2 = (x + y)(x + y + 1) + 2y$ . Let's show existence. By Lemma 2.8, either  $2|(x + y)$  or  $2|(x + y + 1)$ . Then, still by Lemma 2.8,  $2|(x + y)(x + y + 1) + 2y$ . So  $\exists r F(x, y, r)$ . The uniqueness is clear by the injectivity of the multiplication.

We showed in Exercise 2-3.7 that the pairing function thus defined was bijective and increasing on each of its parameters. We leave it to the reader to check that this proof is formalized in  $\mathbf{I}\Delta_0^0$ . The projections are defined by the following formulas:  $P_1(z, r) \equiv \exists x \leq z \langle x, r \rangle = z$ .  $P_2(z, r) \equiv \exists y \leq z \langle r, y \rangle = z$ . The existence and uniqueness derives from the bijective nature of the pairing function. ■

### 3. Induction hierarchies

In order to continue our proof that primitive recursive functions are provably computable in  $\mathbf{RCA}_0$ , we need to examine some consequences of induction, commonly used in mathematics without thinking about it:

1. The principle of bounded collection: every finite set of integers has an upper bound.
2. The principle of minimum: any non-empty set of integers admits a smallest element.

In the presence of restricted induction, each of the above two principles is also restricted. We will understand how the restriction of induction impacts these principles, and study their different links.

### Notation

Let  $n > 0$ . We denote by  $I\Sigma_n^0$  (resp.  $II\Pi_n^0$ ) the induction scheme  $(F(0) \wedge \forall x (F(x) \rightarrow F(x+1))) \rightarrow \forall x F(x)$  restricted to  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formulas.

### 3.1. The bounded collection scheme

The principle of bounded collection — every finite set of integers has an upper bound — is obvious when we define by “finite set” the sets of elements which are precisely bounded: it is then a question of a tautology. The principle becomes less obvious when we define the finite sets as being those that we can put in bijection with the set  $\{0, 1, \dots, n\}$  for a certain  $n$ . We call *bounded collection scheme* for a formula  $F(x, y)$  the statement:

$$\forall n ((\forall x < n \exists y F(x, y)) \rightarrow \exists b \forall x < n \exists y < b F(x, y)) \quad (13)$$

### Notation

Let  $n > 0$ . We denote by  $B\Sigma_n^0$  (resp.  $B\Pi_n^0$ ) the bounded collection scheme restricted to  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formulas.

**In non-standard models.** Given a non-standard model  $M$  and a non-standard integer  $a \in M$ , we can define a function  $f$  from  $\{b : b < a\}$  to  $M$  which is unbounded in  $M$  (we can even define a bijection,  $M$  being countable). Of course if  $M$  is a model of the bounded collection scheme, such a function cannot be defined by a formula inside  $M$ . Conversely, if  $M$  is not a model of the bounded collection scheme for  $\Sigma_n^0$  formulas, this means that there is an element  $a$  and a  $\Sigma_n^0$  formula  $F(x, y)$  such that for all  $x < a$  there exists at least one element  $y$  for which  $F(x, y)$  is true, and such that these elements  $y$  are unbounded in  $M$ .

**Closure of  $\Sigma_n^0$  and  $\Pi_n^0$  formulas by bounded quantifications.** Lots of proofs use the fact that the different levels of the arithmetic hierarchy are closed under bounded quantification. Thus, for a  $\Sigma_1^0$  formula of the form  $\exists y F(x, y)$  with  $F(x, y) \Delta_0^0$ , the formula  $\exists a \forall y < a \exists y F(x, y)$  is also considered to be  $\Sigma_1^0$ . This has been proven via Proposition 9-3.3, and comes from the fact that we can rewrite it in the form  $\exists a \exists b \forall y < a \exists y < b F(x, y)$ . The equivalence between the two formulas however requires the bounded collection scheme, and is therefore not guaranteed in models which do not verify it for the formula  $F$ . We show here that this scheme is sufficient to guarantee Proposition 9-3.3.

**Proposition 3.1 (Parsons [170]).** Let  $F_1(\bar{a}, x), F_2(\bar{a}, x)$  and  $F(\bar{a}, x)$  be  $\Sigma_n^0$  (resp  $\Pi_n^0$ ) formulas for  $n > 0$ . Then, each of the following formulas is provably equivalent in  $\mathbf{Q} + \mathbf{I}\Delta_0^0 + \mathbf{B}\Sigma_n^0$ , as well as in  $\mathbf{Q} + \mathbf{I}\Delta_0^0 + \mathbf{B}\Pi_n^0$ , to a  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formula:

- (i)  $F_1(\bar{a}, x) \wedge F_2(\bar{a}, x), F_1(\bar{a}, x) \vee F_2(\bar{a}, x)$
- (ii)  $\exists x < b F(\bar{a}, x), \forall x < b F(\bar{a}, x),$
- (iii)  $\exists x F(\bar{a}, x)$  (resp.  $\forall x F(\bar{a}, x)$ ) ★

PROOF. Let  $F(\bar{a}, x) \equiv \exists y G(\bar{a}, x, y)$ ,  $F_1(\bar{a}, x) \equiv \exists y G_1(\bar{a}, x, y)$  and  $F_2(\bar{a}, x) \equiv \exists y G_2(\bar{a}, x, y)$ . Consider the following statements:

$$\begin{aligned}
 F_1(\bar{a}, x) \wedge F_2(\bar{a}, x) &\leftrightarrow \exists y \exists y_1, y_2 < y (G_1(\bar{a}, x, y_1) \wedge G_2(\bar{a}, x, y_2)) & (a) \\
 F_1(\bar{a}, x) \vee F_2(\bar{a}, x) &\leftrightarrow \exists y (G_1(\bar{a}, x, y) \vee G_2(\bar{a}, x, y)) & (b) \\
 \exists x < b F(\bar{a}, x) &\leftrightarrow \exists y \exists x < b G(\bar{a}, x, y) & (c) \\
 \forall x < b F(\bar{a}, x) &\leftrightarrow \exists z \forall x < b \exists y < z G(\bar{a}, x, y) & (d) \\
 \exists x F(\bar{a}, x) &\leftrightarrow \exists z \exists x < z \exists y < z G(\bar{a}, x, y) & (e)
 \end{aligned}$$

Now, if  $F, F_1, F_2$  are  $\Sigma_1^0$  with  $G, G_1, G_2 \Delta_0^0$ , the equivalences (a) (b) (c) (e) are easily provable in  $\mathbf{I}\Delta_0^0$ . Equivalence (d) follows directly from the bounded collection scheme for  $G$  which is  $\Delta_0^0$ . So the formulas of (i), (ii), (iii) are provably equivalent to  $\Sigma_1^0$  formulas in  $\mathbf{B}\Sigma_1^0$  and in  $\mathbf{B}\Pi_1^0$ . By passing to the negation, the formulas of (i), (ii), (iii) are provably equivalent to  $\Pi_1^0$  formulas in  $\mathbf{B}\Sigma_1^0$  and in  $\mathbf{B}\Pi_1^0$ , for the case where  $F, F_1, F_2$  are  $\Pi_1^0$ .

Suppose the proposition is true for the  $\Sigma_n^0$  and  $\Pi_n^0$  cases. Then, if  $F, F_1, F_2$  are  $\Sigma_{n+1}^0$  formulas where  $G, G_1, G_2$  are  $\Pi_n^0$ , in the same way  $\mathbf{B}\Sigma_{n+1}^0$  or  $\mathbf{B}\Pi_{n+1}^0$  shows the equivalences (a) (b) (c) (e). As  $\mathbf{B}\Sigma_{n+1}^0$  or  $\mathbf{B}\Pi_{n+1}^0$  prove  $\mathbf{B}\Sigma_n^0$ , and by induction hypothesis on  $G, G_1, G_2$ , the formulas of (i), (ii), (iii) are provably equivalent to  $\Sigma_{n+1}^0$  formulas in  $\mathbf{B}\Sigma_{n+1}^0$  or in  $\mathbf{B}\Pi_{n+1}^0$ . By passing to the negation, the formulas of (i), (ii), (iii) are provably equivalent to  $\Pi_{n+1}^0$  formulas in  $\mathbf{B}\Sigma_{n+1}^0$  or in  $\mathbf{B}\Pi_{n+1}^0$ , in the case where  $F, F_1, F_2$  are  $\Pi_{n+1}^0$ . ■

**Induction implies the collection scheme.** The bounded collection scheme is not one of the usual axioms of arithmetic. Here, we see how it arises from induction. We first start by showing that the collection scheme for  $\Pi_n^0$  formulas allows us to show it for  $\Sigma_{n+1}^0$  formulas.

**Proposition 3.2 (Paris and Kirby [169]).** Let  $n > 0$ . Then,

$$\mathbf{Q} + \mathbf{I}\Delta_0^0 \vdash \mathbf{B}\Pi_n^0 \leftrightarrow \mathbf{B}\Sigma_{n+1}^0 \quad \star$$

PROOF. The implication  $\mathbf{B}\Sigma_{n+1}^0 \rightarrow \mathbf{B}\Pi_n^0$  is immediate. Suppose  $\mathbf{B}\Pi_n^0$ . Let  $\exists z F(x, y, z)$  be a  $\Sigma_{n+1}^0$  formula where  $F$  is  $\Pi_n^0$ , and let  $a \in \mathbb{N}$  be such that  $\forall x < a \exists y (\exists z F(x, y, z))$ . We then have  $\forall x < a \exists v \exists y, z < v F(x, y, z)$ . Using  $\mathbf{B}\Pi_n^0$ , by Proposition 3.1, let  $G(x, v)$  be a  $\Pi_n^0$  formula equivalent to  $\exists y, z < v F(x, y, z)$ . Then, by  $\mathbf{B}\Pi_n^0$ , we have  $\exists b \forall x < a \exists v < b G(x, v)$ , which is then equivalent to  $\exists b \forall x < a \exists v < b \exists y, z < v F(x, y, z)$ , which implies  $\exists b \forall x < a \exists y < b (\exists z F(x, y, z))$ . ■

Now let's see how to show the collection scheme using the induction scheme.

**Theorem 3.3 (Paris and Kirby [169])**

Let  $n > 0$ . Then,  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \rightarrow \mathbf{B}\Sigma_n^0$ .

PROOF. By induction on  $n$ . As  $\mathbf{Q} + \mathbf{I}\Delta_0^0 \vdash \mathbf{B}\Sigma_{n+1}^0 \leftrightarrow \mathbf{B}\Pi_n^0$ , let us show  $\mathbf{Q} \vdash \mathbf{I}\Sigma_{n+1}^0 \rightarrow \mathbf{B}\Pi_n^0$ . Let  $a \in \mathbb{N}$  and  $F(x, y)$  be a formula. Consider the formulas  $G \equiv \forall x < a \exists y F(x, y)$  and  $H(a') \equiv \exists b \forall x < a' \exists y < b (a' \leq a \rightarrow F(x, y))$ . Suppose  $G$ .

In the case where  $F(x, y)$  is  $\Delta_0^0$ , then  $H(a')$  is  $\Sigma_1^0$ .  $H(0)$  is trivially true and by  $G$ , we deduce  $H(a') \rightarrow H(a' + 1)$  (there is here an argument which we leave to the reader). By  $\mathbf{I}\Sigma_1^0$ , we therefore have  $\forall a' H(a')$  and therefore  $H(a)$ , which implies  $\exists b \forall x < a \exists y < b F(x, y)$ .

Suppose now  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \rightarrow \mathbf{B}\Sigma_n^0$ , with  $F(x, y)$  a  $\Pi_n^0$  formula. Then,  $H$  is equivalent to a  $\Sigma_{n+1}^0$  formula by  $\mathbf{B}\Sigma_n^0$ . We then proceed as in the previous paragraph. So for all  $n \in \omega$ , we have  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \rightarrow \mathbf{B}\Sigma_n^0$ . ■

### 3.2. The minimum scheme

The minimum scheme informally tells us that every non-empty set of integers has a smallest element. Formally, we call *minimum scheme* for a formula  $F(x)$  the statement:

$$\exists x F(x) \rightarrow \exists x (F(x) \wedge \forall y < x \neg F(y)) \quad (14)$$

— Notation —

Let  $n > 0$ . We denote by  $\mathbf{L}\Sigma_n^0$  (resp.  $\mathbf{L}\Pi_n^0$ ) the minimum scheme restricted to the  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formulas.

**In non-standard models.** In a non-standard model  $M$ , a *cut* is a strict initial segment of  $M$ , non-empty, and closed under successor. For example the standard part  $\omega$  of a non-standard model is always a cut. If an element  $a$

is in the complement of a cut, it is also necessarily the case for  $a - 1$  because a cut is closed under successor. We deduce that if a cut is definable by a formula  $F$ , that directly contradicts the minimum scheme for the formula  $\neg F$ . This is in fact a necessary and sufficient condition. Indeed if  $A = \{x : F(x)\}$  is a non-empty set which has no smallest element, then the set of elements strictly smaller than any element of  $A$  is necessarily a cut. We deduce the following principle.

**Overspill principle**

The set of standard integers can never be defined in a model which respects the minimum scheme. In particular, if a formula is verified by all standard integers, it will necessarily also be true for a non-standard integer.

**Induction is equivalent to the minimum scheme.** We now see that the minimum scheme is, more or less, a simple reformulation of induction: if the set  $\{x : F(x)\}$  is non-empty and has no smallest element, it means that induction fails for the formula which defines the cut which one can create starting from  $F(x)$ .

**Proposition 3.4.** Let  $n > 0$ . Then,  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \leftrightarrow \mathbf{L}\Pi_n^0$  and  $\mathbf{Q} \vdash \mathbf{I}\Pi_n^0 \leftrightarrow \mathbf{L}\Sigma_n^0$ . ★

PROOF. We show the equivalence  $\mathbf{I}\Sigma_n^0 \leftrightarrow \mathbf{L}\Pi_n^0$ , the other being shown identically.

Suppose  $\neg \mathbf{I}\Sigma_n^0$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula such that  $F(0), F(x) \rightarrow F(x+1)$ , but such that the set  $A = \{x : \neg F(x)\}$  is non-empty. By contraposition,  $\neg F(x) \rightarrow \neg F(x-1)$ . So  $A$  has no smallest element. We therefore have  $\neg \mathbf{L}\Pi_n^0$ .

Now suppose  $\neg \mathbf{L}\Pi_n^0$  and suppose  $\mathbf{I}\Sigma_n^0$  by contradiction. Let  $F(x)$  be a  $\Pi_n^0$  formula such that the set  $A = \{x : F(x)\}$  is non-empty and has no smallest element. Let  $G(x) \equiv \forall y \leq x \neg F(y)$ . By  $\mathbf{I}\Sigma_n^0$ , which implies  $\mathbf{B}\Sigma_n^0$ ,  $G(x)$  is provably equivalent to a  $\Sigma_n^0$  formula. Note that  $\neg F(0)$  because otherwise,  $A$  would have a smallest element. So  $G(0)$ . Suppose  $G(x)$ . If we had  $F(x+1)$ , then  $A$  would have a smallest element. So  $\neg F(x+1)$  as well as  $G(x+1)$ . We therefore have  $G(0) \wedge (G(x) \rightarrow G(x+1))$ , but  $G(x)$  is not verified for any element of  $A$ , which is non-empty. So  $\neg \mathbf{I}\Sigma_n^0$ . ■

The following theorem is particularly interesting: by using the properties of integers, we show that the induction for the  $\Sigma_n^0$  formulas is equivalent to the induction for the  $\Pi_n^0$  formulas.

**Theorem 3.5 (Paris and Kirby [169])**

Let  $n > 0$ . Then,  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \leftrightarrow \mathbf{I}\Pi_n^0$ .

PROOF. Let's show  $\mathbf{Q} \vdash \mathbf{I}\Sigma_n^0 \rightarrow \mathbf{I}\Pi_n^0$ . Suppose that  $\mathbf{I}\Pi_n^0$  fails but not  $\mathbf{I}\Delta_0^0$  (in order to be able to use the basic facts about integers). Let  $F(x)$  be a  $\Pi_n^0$  formula such that  $F(0)$  and  $\forall x(F(x) \rightarrow F(x+1))$ , but  $\neg F(a)$  for an integer  $a > 0$ . Let  $G(y)$  be the formula  $\exists x (a = x + y \wedge \neg F(x))$ . Note that the formula  $G(y)$  is equivalent to a  $\Sigma_n^0$  formula by shifting " $a = x + y$ " towards its  $\Delta_0^0$  part. Also  $G(0)$  is true and  $G(a)$  is false. Let  $y$  be such that  $G(y)$  is true. In particular, there is an  $x$  such that  $a = x + y$  and  $\neg F(x)$ . Necessarily,  $y < a$ , therefore  $x > 0$ , or  $a = (x - 1) + (y + 1)$  and by hypothesis,  $\neg F(x) \rightarrow \neg F(x - 1)$ , therefore  $G(y + 1)$  is true. As  $G(0)$  and  $\forall y (G(y) \rightarrow G(y + 1))$  and  $\neg G(a)$ , then  $\mathbf{I}\Sigma_n^0$  fails.

Let's show  $\mathbf{Q} \vdash \mathbf{I}\Pi_n^0 \rightarrow \mathbf{I}\Sigma_n^0$ . Suppose  $\mathbf{I}\Sigma_n^0$  fails but not  $\mathbf{I}\Delta_0^0$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula such that  $F(0)$  and  $\forall x(F(x) \rightarrow F(x + 1))$ , but  $\neg F(a)$  for an integer  $a > 0$ . Let  $H(y)$  be the formula  $\forall x (a = x + y \rightarrow \neg F(x))$ . As before,  $H(y)$  is equivalent to a  $\Pi_n^0$  formula. Additionally  $H(0)$  is true and  $H(a)$  is false. We also show  $H(y) \rightarrow H(y + 1)$ . Then,  $H(0)$  and  $\forall y (H(y) \rightarrow H(y + 1))$  and  $\neg H(a)$ , so  $\mathbf{I}\Pi_n^0$  fails. ■

**Exercise 3.6 (Hájek and Pudlák [80]). (★)**

We call *ordered induction scheme* for any formula  $F(x)$  the statement

$$\forall x((\forall y < x F(y)) \rightarrow F(x)) \rightarrow \forall x F(x)$$

Show that for all  $n \in \omega$ ,  $\mathbf{Q}$  proves the equivalence between  $\mathbf{I}\Sigma_n^0$  (resp.  $\mathbf{I}\Pi_n^0$ ) and the ordered induction scheme for the  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formulas. ◇

**3.3. Collection and  $\Delta_n^0$  induction**

We have seen that  $\mathbf{B}\Sigma_n^0$  can be deduced from  $\mathbf{I}\Sigma_n^0$ . We now see that, subject to having a minimum of induction ( $\mathbf{I}\Delta_0^0$ ), the collection scheme for all the formulas implies the induction scheme for all the formulas. More precisely:

**Theorem 3.7 (Paris and Kirby [169])**

Let  $n > 0$ . Then,  $\mathbf{Q} + \mathbf{I}\Delta_0^0 \vdash \mathbf{B}\Sigma_{n+1}^0 \rightarrow \mathbf{I}\Sigma_n^0$ .

PROOF. By induction on  $n$ . Suppose  $\mathbf{B}\Sigma_{n+1}^0$ . Let  $F(x) \equiv \exists y G(x, y)$  for  $G$  be a  $\Pi_{n-1}^0$  formula (or  $\Delta_0^0$  if  $n = 1$ ). Suppose  $F(0)$  and  $\forall x(F(x) \rightarrow F(x +$

1)). Let  $a \in \mathbb{N}$ . Let's show  $F(a)$ . We have

$$\forall x \leq a \ (\exists y G(x, y) \rightarrow \exists u G(x + 1, u))$$

In other words

$$\forall x \leq a \ \exists u \ (\exists y G(x, y) \rightarrow G(x + 1, u))$$

By  $\text{B}\Pi_n^0$  (equivalent to  $\text{B}\Sigma_{n+1}^0$ ), there exists  $b \in \mathbb{N}$  such that

$$\forall x \leq a \ \exists u \leq b \ (\exists y G(x, y) \rightarrow G(x + 1, u))$$

which implies

$$\forall x \leq a \ \exists u \leq b \ (\exists y \leq b \ G(x, y) \rightarrow G(x + 1, u))$$

and so

$$\forall x \leq a \ (\exists y \leq b \ G(x, y) \rightarrow \exists u \leq b \ G(x + 1, u))$$

Let  $H(x) \equiv x \leq a \rightarrow (\exists u \leq b \ G(x, u))$ . If  $n = 1$ , the formula  $G(x, u)$  is  $\Delta_0^0$ , and therefore also the formula  $H(x)$ . By  $\text{I}\Delta_0^0$ , we therefore have  $H(x)$  for all  $x$ . If  $n > 1$ , the formula  $G(x, u)$  is  $\Pi_{n-1}^0$ . By  $\text{B}\Sigma_{n+1}^0$ , the formula  $H(x)$  is therefore equivalent to a  $\Pi_{n-1}^0$  formula. By induction hypothesis, we have  $\text{I}\Sigma_{n-1}^0$  (which implies  $\text{I}\Pi_{n-1}^0$  by Theorem 3.5) and we therefore have  $H(x)$  for all  $x$ . In all cases, we have  $H(x)$  for all  $x$ . In particular,  $H(a)$  is true, so  $\exists u \leq b \ G(a, u)$  and therefore  $F(a)$ . ■

It is possible to refine the second implication, and show that  $\text{B}\Sigma_n^0$  implies induction for the provably  $\Delta_n^0$  formulas.

### Notation

Let  $n > 0$ . We denote by  $\text{I}\Delta_n^0$  the scheme

$$\forall x (F(x) \leftrightarrow G(x)) \rightarrow ((F(0) \wedge F(n) \rightarrow F(n + 1)) \rightarrow \forall n F(n))$$

restricted to formulas  $F, G$  with  $F \Sigma_n^0$  and  $G \Pi_n^0$ .

### Theorem 3.8 (Hájek and Pudlák [80])

Let  $n > 0$ . Then,  $\mathbf{Q} + \text{I}\Delta_0^0 \vdash \text{B}\Sigma_n^0 \rightarrow \text{I}\Delta_n^0$ .

PROOF. Suppose  $\text{B}\Sigma_n^0$ . Let  $\exists y F(x, y)$  and  $\forall x G(x, y)$  be formulas respectively  $\Sigma_n^0$  and  $\Pi_n^0$ , such that  $\forall x (\exists y F(x, y) \leftrightarrow \forall y G(x, y))$ . Suppose  $\exists y F(0, y)$  and  $\forall x (\exists y F(x, y) \rightarrow \exists y F(x + 1, y))$

Let  $a \in \mathbb{N}$ . Let's show  $\exists y F(a, y)$ . We have

$$\forall x \exists y (F(x, y) \vee \neg G(x, y))$$

So by  $\text{B}\Sigma_n^0$ , there exists  $b \in \mathbb{N}$  such that

$$\forall x \leq a \exists y \leq b (F(x, y) \vee \neg G(x, y))$$

Let  $H(x) \equiv (x \leq a \rightarrow (\exists y \leq b F(x, y)))$ . By hypothesis on  $F$  and  $G$ , we have  $H(0)$  and  $H(x) \rightarrow H(x+1)$ . By  $\text{B}\Sigma_n^0$ , the formula  $H(x)$  is equivalent to a  $\Pi_{n-1}^0$  formula. As  $\text{B}\Sigma_n^0 \rightarrow \text{I}\Sigma_{n-1}^0$  and  $\text{I}\Sigma_{n-1}^0 \leftrightarrow \text{I}\Pi_{n-1}^0$ , then we have  $H(x)$  for all  $x$ . In particular  $\exists y F(a, y)$ . ■

Thus,  $\text{B}\Sigma_n^0$  is between  $\text{I}\Sigma_n^0$  and  $\text{I}\Delta_n^0$ . The question of whether  $\text{I}\Delta_n^0$  in turn implies  $\text{B}\Sigma_n^0$  remained open for over a decade, before being solved by Slaman in 2002, however using the theory  $\text{Q} + \text{I}\Sigma_1^0$  as the base theory.

**Theorem 3.9 (Slaman [203])**

*Let  $n > 0$ . Then,  $\text{Q} + \text{I}\Sigma_1^0 \vdash \text{B}\Sigma_n^0 \leftrightarrow \text{I}\Delta_n^0$ .*

The proof of the implication  $\text{I}\Delta_n^0 \rightarrow \text{B}\Sigma_n^0$  uses advanced coding techniques in non-standard models of arithmetic. To date, there is no syntactic proof of this implication.

### 3.4. Summary diagram

We therefore have for the moment in  $\text{Q} + \text{I}\Delta_0^0$  (and therefore in  $\text{RCA}_0$ ) the following induction hierarchy.

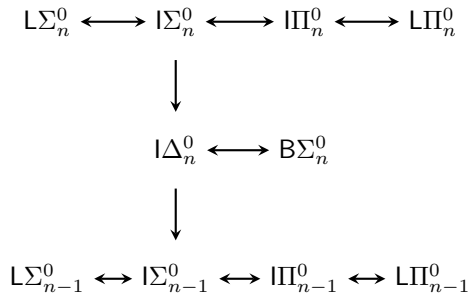


Figure 3.10: Induction hierarchy. The arrows indicate implications over  $\text{Q} + \text{I}\Sigma_1^0$ .

It is possible to show that the reciprocal implications of Figure 3.10 do not hold. This is done via the construction of non-standard models, within which for example  $\text{B}\Sigma_n^0$  is valid, but not  $\text{I}\Sigma_n^0$ . The reader who wishes to learn more can consult [108] or [80].

## 4. Primitive recursive functions and $\text{RCA}_0$

We now have the elements necessary to prove that the primitive recursive functions are provably computable in  $\text{RCA}_0$ . The proof goes through an encoding of finite sequences by integers, via a technique similar to that used for the  $\beta$  function of Gödel (see Lemma 9-3.6), but not based on the Chinese Remainder Theorem. The uniqueness of the decomposition of numbers into prime factors will be a key element of our coding, more particularly Gauss lemma. We recall that two numbers  $a, b$  are *coprime* if 1 is their only common divisor. This is a  $\Delta_0$  formula.

**Lemma 4.1 (Gauss lemma).**  $\mathbf{Q} + \text{I}\Delta_0^0$  shows that for all numbers  $p, a, b$  such that  $p$  and  $b$  are coprime, if  $p|ab$  then  $p|a$ . ★

PROOF. Let  $p, b$  be coprime. Suppose by contradiction that Gauss lemma is false, that is to say that the set  $\{a \neq 0 : p|ab \text{ and } \neg p|a\}$  is non-empty. By minimum scheme for the formulas  $\Delta_0$ , let  $a$  be the smallest element of this set. Let us show  $a < p$ . Suppose by contradiction  $a \geq p$ . Then, by Euclidean division (see Exercize 2.7), there exist  $q, r$  with  $r < p$  such that  $a = pq + r$ . As  $\neg p|a$ , then  $r \neq 0$ . As  $p|(pq + r)b$ , then  $p|pqb + rb$ . Now,  $p|pqb$ , so  $p|rb$ . However, as  $r < p$  we have  $\neg p|r$ , which contradicts the minimality of  $a$ . So  $a < p$ . Now let  $p = aq + r$  be  $0 \leq r < a$ . Let us show  $r \neq 0$ . Suppose  $p = aq$ . As  $\neg p|a$ , then  $q > 1$ . As  $p|ab$ , then  $aq|ab$  and therefore  $q|b$ . So  $q$  is a common divisor of  $b$  and  $p$  different from 1, which contradicts our hypothesis. So  $r \neq 0$ . We then have  $bp = baq + br$ , so  $br = bp \div baq$ . As  $p|bp$  and  $p|baq$ , then  $p|br$ . Now,  $0 < r < a < p$  therefore  $\neg p|r$ , which contradicts the minimality of  $a$ . ■

Our coding of lists relies on the ability to generate sequences of numbers pairwise coprime in the following manner.

**Lemma 4.2.** Let  $x$  be fixed, and let  $a$  be an integer, multiple of all integers  $i$  for  $1 \leq i \leq x$ .  $\mathbf{Q} + \text{I}\Delta_0^0$  shows that for  $i, j \leq x$  with  $i \neq j$  the numbers  $a(i+1)+1$  and  $a(j+1)+1$  are coprime. ★

PROOF. Suppose by contradiction that a prime number  $p$  divides  $a(i+1)+1$  and  $a(j+1)+1$  with  $i < j$ . So  $p$  divides  $a(j+1)+1 - a(i+1)+1 = a(j \div i)$  as well. Since  $p$  is prime, by Gauss lemma,  $p$  divides  $a$  or  $p$  divides  $j \div i$ . By assumption on  $a$ , in all cases  $p$  divides  $a$ . So  $p$  divides  $a(i+1)$ . Since  $p$  also divides  $a(i+1)+1$ , then  $p$  divides  $a(i+1)+1 \div a(i+1) = 1$ , which is a contradiction. So the numbers  $a(i+1)+1$  and  $a(j+1)+1$  are coprime. ■

The idea of coding our lists is as follows. Given  $x_0, x_1, \dots, x_{n-1}$ , we start by choosing  $a$  multiple of all integers  $z$  for  $1 \leq z \leq \max\{\langle i, x_i \rangle : i < n\}$ . Our

list is then coded by the ordered pair  $(a, b)$  such that  $b = \prod_{i \in A} (a(\langle i, x_i \rangle + 1) + 1)$ . Given  $(a, b)$ , the  $i$ th element of our list is given by the unique  $z \leq a$  such that  $\langle i, z \rangle \leq a$  and such that  $(a(\langle i, x_i \rangle + 1) + 1) | b$ .

### Notation

For an integer  $a$  we denote by  $p_i^a$  the integer  $a(i + 1) + 1$ .

**Definition 4.3.** An ordered pair  $(a, b)$  codes for a sequence of size  $n$  if  $\forall i < n \exists! x \leq a \langle i, x \rangle \leq a \wedge p_{\langle i, x \rangle}^a | b$ .  $\diamond$

### Notation

For a pair  $(a, b)$  which codes for a sequence of size  $n$ , we denote by  $(a, b)_i$  the unique integer  $x \leq a$  such that  $\langle i, x \rangle \leq a \wedge p_{\langle i, x \rangle}^a | b$ .

Our first use of the  $\Sigma_1^0$  induction scheme will be to show the existence of an integer  $a$  multiple of enough elements to encode our sequences.

**Lemma 4.4.**  $\text{RCA}_0$  shows that for all  $x \in \mathbb{N}$ , there exists a smallest integer  $y \in \mathbb{N}$  which is a multiple of all integers  $i$  for  $1 \leq i \leq x$ .  $\star$

PROOF. Let us show the existence. This is clear to  $x = 0$ . Suppose the existence for  $x$ . Let  $y$  be a multiple of all integers smaller than  $x$ . We show in  $\mathbf{Q} + \text{ID}_0^0$  that the divisors of  $y$  are also divisors of  $y \times (x + 1)$ , and that  $x + 1$  is a divisor of  $y \times (x + 1)$ . By  $\text{IS}_1^0$ , existence is assured for all  $x$ . In particular, for  $x$  fixed, the set  $\{y : \forall 1 \leq i \leq x \ i | y\}$  is not empty. Finally, we apply the minimum scheme for  $\Delta_0$  formulas to obtain the smallest element.  $\blacksquare$

We can now move on to the coding of finite sequences.

**Lemma 4.5.** Suppose that  $(a, b)$  codes for a sequence of size  $n$ . Then,  $\text{RCA}_0$  shows that for all  $r$ , there exists a pair  $(a', b')$  which codes for a sequence of size  $n + 1$ , such that  $(a', b')_n = r$  and such that  $\forall i < n \ (a', b')_i = (a, b)_i$ .  $\star$

PROOF. Let us first show the following statement: if  $(a, b)$  codes for a sequence of size  $n$ , then for all  $a'$  such that the divisors of  $a$  are all divisors of  $a'$ , there exists  $b'$  such that  $(a', b')_i = (a, b)_i$  for all  $i < n$ .

Suppose this is the case for  $n$ . Let  $(a, b)$  coding for a sequence of size  $n + 1$  and  $a'$  such that the divisors of  $a$  are all divisors of  $a'$ . Then,  $(a, b)$  also codes for a sequence of size  $n$ . So by induction hypothesis, there exists  $b'$  such that  $(a, b)$  and  $(a', b')$  code for the same sequence of size  $n$ . Let  $b'$  be the smallest integer of the sort  $(\Delta_0$  minimization scheme). Then,  $(a', b')$  does not code for a sequence of size  $n + 1$ , because if  $m \times p_{\langle n, x \rangle}^{a'} = b'$ , then  $(a', m)$

codes for the same sequence of size  $n$  as  $(a, b)$  — by application of Gauss lemma and the fact that  $p_{\langle n, x \rangle}^{a'}$  and  $p_{\langle i, y \rangle}^{a'}$  are coprime for all  $i < n$  — which contradicts the minimality of  $b'$ . Now, let  $r = (a, b)_n$  and  $b'' = b' p_{\langle n, r \rangle}^{a'}$ . Still using Gauss lemma, we show that  $(a, b)$  and  $(a', b'')$  code for the same sequence of size  $n + 1$ .

Let us now show the lemma. Let  $(a, b)$  coding for a sequence of size  $n$  and  $r \in \mathbb{N}$ . According to Proposition 4.4, let  $a'$  be such that any  $i \leq \max(a, \langle n, r \rangle)$  divides  $a'$ . According to the previous paragraph, there exists  $b'$  such that  $(a', b')$  codes for the same sequence of size  $n$  as  $(a, b)$ . Let  $b'' = b' p_{\langle n, r \rangle}^{a'}$ . It is clear that  $(a', b'')_n = r$ . Still using Gauss lemma we show that  $(a', b'')_i = (a', b')_i = (a, b)_i$  for  $i < n$ . ■

Let us pass finally to the proof that the primitive recursive functions are provably computable.

**Theorem 4.6**

*The primitive recursive functions are provably computable in  $\text{RCA}_0$ .*

PROOF. We leave it to the reader to show that the theorem holds for the basic functions. The composition scheme does not present any particular difficulty: Let  $f(\bar{x}) = g(h_1(\bar{x}), \dots, h_k(\bar{x}))$ , for functions  $g, h_1, \dots, h_k$  provably computable via formulas  $G, H_1, \dots, H_k$ . Then,  $f$  is provably computable by the formula

$$F(\bar{x}, r) \equiv \exists y_1, \dots, y_k \ H_1(\bar{x}, y_1) \wedge \dots \wedge H_k(\bar{x}, y_k) \wedge G(y_1, \dots, y_k, r).$$

We now move on to the primitive recursion scheme. Let

$$\begin{aligned} f(\bar{x}, 0) &= g(\bar{x}) \\ f(\bar{x}, n+1) &= h(\bar{x}, n, f(\bar{x}, n)) \end{aligned}$$

for functions  $g, h$  provably computable by formulas  $G, H$ . We are going to create a formula  $F(\bar{x}, n, r)$  such that:

$$\begin{aligned} \text{RCA}_0 &\vdash \forall \bar{x}, n \ \exists! r \ F(\bar{x}, n, r) \\ \text{RCA}_0 &\vdash \forall \bar{x}, \exists r \ F(\bar{x}, 0, r) \wedge G(\bar{x}, r) \\ \text{RCA}_0 &\vdash \forall \bar{x}, n \ \exists r_1, r_2 \ F(\bar{x}, n, r_1) \wedge F(\bar{x}, n+1, r_2) \wedge H(\bar{x}, n, r_1, r_2) \end{aligned}$$

The formula in question is given by

$$F(\bar{x}, n, r) \equiv \begin{aligned} &\exists a, b \ (a, b) \text{ codes for a sequence of size } n+1 \text{ and} \\ &r = (a, b)_n \text{ and } G(\bar{x}, (a, b)_0) \text{ and } \forall i < n \ H(\bar{x}, i, (a, b)_i, (a, b)_{i+1}) \end{aligned}$$

Note that by  $\text{BS}_1^0$ , which is provable in  $\text{IS}_1^0$ , the above formula is equivalent to a  $\Sigma_1^0$  formula.

Let us prove the existence by induction on  $n$ . It is clear that we have  $\exists r F(\bar{x}, 0, r)$ . Suppose  $\exists r F(\bar{x}, n, r)$  via a pair  $(a, b)$  coding for a sequence of size  $n + 1$ . Let  $r$  be the unique integer such that  $H(\bar{x}, n, (a, b)_n, r)$ .

According to Lemma 4.5, there exists  $(a', b')$  which codes for a sequence of size  $n + 2$ , such that  $(a', b')_{n+1} = r$  and such that  $(a', b')_i = (a, b)_i$  for  $i \leq n$ . It is then clear that we have  $G(\bar{x}, (a', b')_0) \wedge \forall i < n H(\bar{x}, i, (a', b')_i, (a', b')_{i+1})$ . In addition,  $H(\bar{x}, n, (a', b')_n, (a', b')_{n+1})$ . So  $F(\bar{x}, n + 1, (a', b')_{n+1})$ . By  $IS_1^0$ , existence is assured for all  $n$ .

Now let's show the uniqueness. Again, it is clear for  $n = 0$ . Suppose the uniqueness for  $n$ , and suppose that  $F(\bar{x}, n + 1, r_1)$  is satisfied via  $(a_1, b_1)$  and that  $F(\bar{x}, n + 1, r_2)$  is satisfied via  $(a_2, b_2)$ . Let  $v_1 = (a_1, b_1)_n$  and  $v_2 = (a_2, b_2)_n$ . Then,  $F(\bar{x}, n, v_1)$  and  $F(\bar{x}, n, v_2)$ . By induction hypothesis, we have  $v_1 = v_2$ . So  $H(\bar{x}, n, v_1, (a, b)_{n+1})$  and  $H(\bar{x}, n, v_1, (a', b')_{n+1})$ . So  $(a, b)_{n+1} = (a', b')_{n+1}$  and therefore  $r_1 = r_2$ . ■

## 5. Bounded comprehension scheme

We now see another axiom scheme on which it is worth stopping, and which one could summarize as follows: “finite sets exist”. We will see that, when the induction is restricted, the precise formulation of this statement, as well as its validity, become non-trivial.

### 5.1. Coding of finite sets

Now that we can use the primitive recursive functions in  $RCA_0$ , it is common to use the function  $x \mapsto 2^x$  to encode finite sets, coding the set  $x_0 < x_1 < \dots < x_n$  by the integer  $n = 2^{x_0} + 2^{x_1} + \dots + 2^{x_n}$ . Via this encoding, we easily recover from an integer  $n$  the set that it codes via the following lemma.

**Lemma 5.1 (Hájek and Pudlák [80]).**  $RCA_0$  proves that for any  $n, x \in \mathbb{N}$ , there exists a unique triplet  $(u, v, w)$  with  $u \leq y$ ,  $v < 1$  and  $w < 2^x$  such that:

$$y = 2^{x+1}u + 2^xv + w$$

Lemma 5.1 is proved using two applications of Euclidean division.

**Definition 5.2.** We call  $x$ -th bit of  $y$  the unique integer  $v \in \{0, 1\}$  of the preceding decomposition. We will write  $x \in y$  if  $v = 1$  and  $x \notin y$

otherwise. An integer  $y$  is the *canonical code* of the set  $A$  if for all  $x \in \mathbb{N}$ ,  $x \in A$  iff the  $x \in y$ . A set is *coded* if it has a canonical code.  $\diamond$

Note that the relation “ $x \in y$ ” is  $\Delta_0^0$  because it is written  $\exists u \leq y \exists w < 2^x (y = 2^{x+1}i + 2^x + w)$ . The canonical code of a set, if it exists, is unique by Lemma 5.1. It is easy to show in  $\text{RCA}_0$  that a set  $X$  is finite iff it has a canonical code (see Exercise 5.4 to come). Be careful however, by “finite”, we must understand finite in the model.

It is possible to extract in  $\text{RCA}_0$  all the standard information of canonical codes, such as their size, maximum or minimum element. We leave it to the reader to show that the corresponding functions are indeed primitive recursive.

## 5.2. Notions of infinity and $\text{RCA}_0$

What is an infinite set? The most intuitive definition, and which we will adopt by default, is that of the absence of upper bound:

**Definition 5.3.** A set  $A$  is *infinite* if for all  $x$  there exists  $y > x$  such that  $y \in A$ . A set is *finite* if it is not infinite.  $\diamond$

**In non-standard models.** Beware, the notion of finiteness and infinity are always relative to the model. If  $M$  is a non-standard model and  $a \in M$  is a non-standard integer, the set  $\{x \in M : x < a\}$  is infinite outside the model, but finite in the model. Also, be careful, from outside the model, we can build sets that are bounded in the model (and therefore finite) but which do not belong to it. This is the case for example of the initial segment made up of standard integers.

**Exercise 5.4.** Show in  $\text{RCA}_0$  that a set is finite iff it is canonically coded:

- (i)  $\text{RCA}_0 \vdash \forall y \exists X (y \text{ is the canonical code of } X)$
- (ii)  $\text{RCA}_0 \vdash \forall X \forall b (\forall x (x \in X \rightarrow x < b) \rightarrow X \text{ is canonically coded}). \quad \diamond$

**Equivalent definitions.** We could consider that a set  $A \subseteq \mathbb{N}$  is infinite if it is in bijection with  $\mathbb{N}$ . We could also consider that a set is infinite if it contains “blocks” of arbitrary size.  $\text{RCA}_0$  is sufficiently powerful to show the equivalence of these definitions.

**Proposition 5.5.**  $\text{RCA}_0$  proves the following equivalence for all  $A$ :

- (1)  $A$  is infinite
- (2) There exists an increasing one-to-one function  $f : \mathbb{N} \rightarrow A$

- (3) For any  $z$ , the initial segment of  $A$  of cardinality exactly  $z$  has a canonical code. ★

PROOF. (1)  $\rightarrow$  (2): Let  $g(x)$  be the smallest  $y > x$  such that  $y \in X$ . The function is  $\Sigma_1^0$ , and total because  $A$  is infinite, so is  $\Delta_1^0$ . The function  $g$  exists by  $\Delta_1^0$  comprehension. Let  $a$  be the smallest element of  $A$  (which exists by Proposition 3.4). Then, the function  $f : \mathbb{N} \rightarrow A$  defined by  $f(0) = a$  and  $f(n+1) = g(f(n))$  exists by primitive recursion and satisfies (2).

(2)  $\rightarrow$  (3): Let  $f : \mathbb{N} \rightarrow A$  be an increasing one-to-one function, and let  $z \in \mathbb{N}$ . By  $\Delta_1^0$  comprehension, the set  $A_z = \{f(x) : x < z\}$  exists, and by Exercise 5.4, the set  $A_z$  has a canonical code.

(3)  $\rightarrow$  (1): Let  $x \in \mathbb{N}$ . By (3), the canonical code  $q$  of the initial segment of  $A$  of cardinality  $x+1$  exists. The maximum element of  $q$  is an element of  $A$  greater than  $x$ . So  $A$  is infinite. ■

### 5.3. Bounded comprehension scheme

We have seen that in  $\text{RCA}_0$ , a set is finite iff it has a canonical code. Be careful, however, these are the sets whose existence can be shown in the model, via the comprehension scheme. On the other hand, given an arbitrary formula  $F(x)$  and a bound  $b$ , the set  $\{x < b : F(x)\}$  does not necessarily exist if  $F$  is too complex. The set will of course exist when  $F$  is provably  $\Delta_1^0$ . It is in fact possible to show, thanks to the  $\Sigma_1^0$  induction scheme, that this will also be the case if  $F$  is  $\Sigma_1^0$ . This is in fact an equivalent condition:  $\text{Q} + \text{I}\Delta_0^0$  shows that  $\text{I}\Sigma_1^0$  is equivalent to the fact that the set  $\{x < b : F(x)\}$  is canonically coded for any  $\Sigma_1^0$  formula  $F$ . This equivalence is propagated to all levels of the hierarchy.

Remember the notation  $y \in n$  of Definition 5.2. We call *bounded comprehension scheme* for a formula  $F(x)$  the statement

$$\forall t \exists n \forall y (y \in n \leftrightarrow (y < t \wedge F(y))) \quad (15)$$

As in the case of the induction scheme, the bounded comprehension hierarchy extends to  $\Delta_n^0$  predicates. For formulas  $F(x), G(x)$  with  $F(x) \Sigma_n^0$  and  $G(x) \Pi_n^0$ , the bounded comprehension scheme is the statement

$$\forall x (F(x) \leftrightarrow G(x)) \rightarrow (\forall t \exists n \forall y (y \in n \leftrightarrow (y < t \wedge F(y)))) \quad (16)$$

#### Notation

Let  $n > 0$ . We denote by  $\text{BC}\Sigma_n^0$  (resp.  $\text{BC}\Pi_n^0$ ) the bounded comprehension scheme (15) restricted to the  $\Sigma_n^0$  (resp.  $\Pi_n^0$ ) formulas and  $\text{BC}\Delta_n^0$  the bounded comprehension scheme (16) restricted to  $\Delta_n^0$  formulas.

In  $\text{RCA}_0$ , if a set exists, its complement also. Thus, for all  $n > 0$ ,  $\text{RCA}_0 \vdash \text{BC}\Sigma_n^0 \leftrightarrow \text{BC}\Pi_n^0$ .

**Theorem 5.6 (Hájek and Pudlák [80])**

Or  $n > 0$ .  $\text{RCA}_0 \vdash \text{BC}\Sigma_n^0 \leftrightarrow \text{I}\Sigma_n^0$ .

PROOF. Suppose  $\text{BC}\Sigma_n^0$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula such that  $F(0)$  is true, and such that  $F(n) \rightarrow F(n+1)$  for all  $n \in \mathbb{N}$ . Let  $a \in \mathbb{N}$ . Let us show that  $F(a)$  is true. By  $\text{BC}\Sigma_n^0$ , the set  $X = \{x \leq a : F(x)\}$  is coded by an integer  $z$ . Let  $G(n) \equiv (n \leq a \wedge n \in z) \vee n > a$ . We have  $G(0)$  and  $G(n) \rightarrow G(n+1)$ . By  $\text{I}\Delta_0^0$ , we have  $G(x)$  for all  $x$ , and therefore  $F(a)$ . Suppose  $\text{I}\Sigma_n^0$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula, and let  $z$  be fixed. Let  $G(q)$  be the  $\Pi_n^0$  formula  $\forall x < z (F(x) \rightarrow x \in q)$ . The integer  $2^z - 1$  codes for the set  $\{x \in \mathbb{N} : x < z\}$ . In particular,  $G(2^z - 1)$  is true. So by  $\text{L}\Pi_n^0$ , which is equivalent to  $\text{I}\Sigma_n^0$ , there is a smallest element  $q$  which satisfies  $G$ . Suppose  $x \in q \wedge \neg F(x)$ . Then,  $q - 2^x < q$  also satisfies  $G$ , which contradicts the minimality of  $q$ . So  $\forall x < z (F(x) \leftrightarrow x \in q)$ . ■

**Exercise 5.7. (★★)** Show that for all  $n \in \omega$ ,  $\text{RCA}_0 \vdash \text{I}\Delta_n^0 \leftrightarrow \text{BC}\Delta_n^0$ . ◇

Figure 5.8 summarizes the relationships between the different schemes encountered so far.

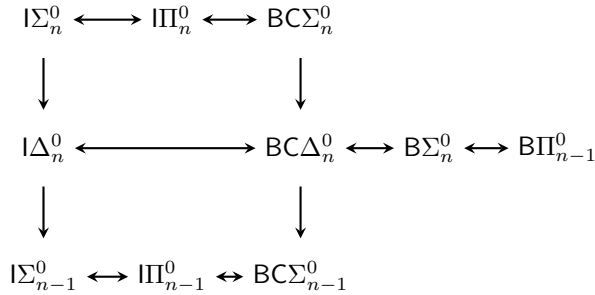


Figure 5.8: Induction, collection and bounded comprehension hierarchies. Arrows denote implications over  $\text{RCA}_0$ .

## 6. Conservation theorems

When we extend a theory  $S$  into a theory  $T$ , it is natural to ask to what extent the new theory  $T$  is stronger than  $S$ : what unprovable statements

in  $S$  become theorems of  $T$ ? We have so far studied individual statements, such as the Bolzano-Weierstrass theorem, which is provable in  $\text{ACA}_0$  but not in  $\text{WKL}_0$ . In this chapter, we will take a more systematic approach, and study the relative provability for collections of formulas. We will focus on three main types of questions:

- (1) *To what extent does second-order arithmetic prove new statements of first-order arithmetic?* To answer it more completely, we will now try to determine the *first-order part* of second-order arithmetic subsystems such as  $\text{RCA}_0$ ,  $\text{WKL}_0$  or  $\text{ACA}_0$ . In other words, we will find, for each of these systems  $T$ , a first-order theory  $S$  such that the first-order statements provable in  $T$  are exactly those provable in  $S$ .
- (2) *What is the relative power of systems of second-order arithmetic?* More precisely, we will show that for certain classes  $\Gamma$  of formulas and for subsystems  $S \subseteq T$  of second-order arithmetic, if  $T$  proves a statement in  $\Gamma$ , then this statement is already provable in  $S$ . For example, we will see that any provable arithmetic statement in  $\text{WKL}_0$  is already provable in  $\text{RCA}_0$ .
- (3) *To what extent does infinite mathematics prove statements which are not provable by finitary mathematics?* This question, at the heart of the crisis in the foundations of mathematics, is at the origin of Reverse Mathematics. We will see how the latter provide a partial response to Hilbert's program.

We will provide answers of the same nature to these three questions, by proving *conservation theorems*.

**Definition 6.1.** Let  $T$  and  $S$  be countable theories in languages  $\mathcal{L}_T \supseteq \mathcal{L}_S$  and let  $\Gamma$  be a set of closed formulas of  $\mathcal{L}_S$ .  $T$  is a *conservative extension* of  $S$  for the formulas of  $\Gamma$  if for any formula  $F \in \Gamma$ ,  $T \vdash F$  iff  $S \vdash F$ . If  $\Gamma$  is the set of formulas of  $\mathcal{L}_S$ , then we say that  $T$  is a *conservative extension* of  $S$ . ◇

Question (1) will ask to prove conservation theorems in the case where  $S$  is a theory in  $\mathcal{L}_{\text{PA}}$  and  $T$  in  $\mathcal{L}_{Z_2}$ , while question (2) will call for conservation theorems where  $S$  and  $T$  are both theories in  $\mathcal{L}_{Z_2}$ .

The theories that we are going to study are all subject to the Completeness Theorem, which says that a statement  $A$  is provable in a theory  $T$  iff any model of  $T$  is a model of  $A$ . In this case, to prove that a theory  $T$  is a conservative extension of  $S$  for a collection of closed formulas  $\Gamma$ , it suffices to show that for any model  $\mathcal{M}$  of  $S$ , it has a model  $\mathcal{N}$  of  $T$ , such that for any formula  $F \in \Gamma$ , if  $\mathcal{N} \models F$  then  $\mathcal{M} \models F$ . The structure  $\mathcal{N}$  is generally

obtained from  $\mathcal{M}$  by adding elements to it to satisfy  $T$ , being careful not to prove new formulas of  $\Gamma$ .

In this chapter, we will restrict ourselves to theories in countable languages, such as  $\mathcal{L}_{\text{PA}}$  or  $\mathcal{L}_{\text{Z}_2}$ . The following proposition will allow us to restrict the technique described above to countable structures.

**Theorem 6.2 (Countable Löwenheim-Skolem)**

*Let  $T$  be a consistent theory in a countable language  $\mathcal{L}$ . Then, there exists a countable model of  $T$ .*

PROOF. The proof of the completeness theorem 9-2.22 of Gödel produces a countable model in the case where the language  $\mathcal{L}$  is countable. ■

The previous proposition is a special case of the more general so-called *Löwenheim-Skolem* theorem, and we will sometimes refer to it in this way. The following proposition gives a very general approach to prove conservation results.

**Proposition 6.3.** Let  $S$  and  $T \supseteq S$  be theories in the countable languages  $\mathcal{L}_T \supseteq \mathcal{L}_S$ , and let  $\Gamma$  be a set of closed formulas of  $\mathcal{L}_S$ . Suppose that for any countable model  $\mathcal{M}$  of  $S$ , there exists a model  $\mathcal{N}$  of  $T$  such that

(★) For any formula  $F \in \Gamma$ , if  $\mathcal{M} \not\models F$ , then  $\mathcal{N} \not\models F$

Then,  $T$  is a conservative extension of  $S$  for the formulas of  $\Gamma$ . ★

PROOF. Let  $F \in \Gamma$  be a formula such that  $S \not\vdash F$ . Then,  $S \cup \{\neg F\}$  is a consistent theory, so by Theorem 6.2 of Löwenheim-Skolem, there exists a countable model  $\mathcal{M}$  of  $S \cup \{\neg F\}$ . Thus, there is a model  $\mathcal{N}$  of  $T$  satisfying (★). In particular,  $\mathcal{N} \not\models F$ , but  $\mathcal{N} \models T$ , so  $T \not\vdash F$  (see Theorem 9-2.20). ■

### 6.1. First-order parts

We will focus our attention to the first-order part of subsystems of second-order arithmetic. Let us begin by formally defining what is meant by *first-order part* for a theory and for a structure.

**Definition 6.4.** The *first-order part* of a theory  $T$  of  $\mathcal{L}_{\text{Z}_2}$  is the set of statements of  $\mathcal{L}_{\text{PA}}$  provable in  $T$ . The *first-order part* of an  $\mathcal{L}_{\text{Z}_2}$ -structure  $(M, S^{\mathcal{M}}, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  is the  $\mathcal{L}_{\text{PA}}$ -structure

$$\mathcal{M} \upharpoonright_{\mathcal{L}_{\text{PA}}} = (M, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$$

◇

Note that if  $\mathcal{M} \models T$  for a  $T$  theory of second-order arithmetic, then  $\mathcal{M} \upharpoonright_{\mathcal{L}_{\text{PA}}} \models S$ , where  $S$  is the first-order part of  $T$ . Characterizing the first-order part

of a system  $T$  of second-order arithmetic consists of two steps: first, we need to identify a set  $S$  of axioms in  $\mathcal{L}_{\text{PA}}$  such that  $T \vdash S$ , then prove that  $T$  is a conservative extension of  $S$ . To do this, we will use a more specific version of Proposition 6.3 where  $\mathcal{N} \upharpoonright_{\mathcal{L}_{\text{PA}}} = \mathcal{M}$ .

**Definition 6.5.** An  $\mathcal{L}_{Z_2}$ -structure  $\mathcal{N}$  is an  $\omega$ -extension of an  $\mathcal{L}_{\text{PA}}$ -structure  $\mathcal{M}$  if  $\mathcal{N} \upharpoonright_{\mathcal{L}_{\text{PA}}} = \mathcal{M}$ . We then write  $\mathcal{M} \subseteq_{\omega} \mathcal{N}$  and we also say that  $\mathcal{M}$  is a  $\omega$ -substructure of  $\mathcal{N}$ .  $\diamond$

Beware, the notion of  $\omega$ -extension has nothing to do with that of  $\omega$ -structure of Definition 22-3.4. In particular,  $\mathcal{N}$  can be an  $\omega$ -extension of  $\mathcal{M}$  without being an  $\omega$ -structure.

**Proposition 6.6.** Let  $S$  be an  $\mathcal{L}_{\text{PA}}$ -theory and  $T$  an  $\mathcal{L}_{Z_2}$ -theory. If any countable model of  $S$  is an  $\omega$ -substructure of a model of  $T$ , then  $T$  is a conservative extension of  $S$ .  $\star$

PROOF. Let  $\mathcal{M}$  be a countable model of  $S$  and let  $\mathcal{N}$  be an  $\omega$ -extension of  $\mathcal{M}$  such that  $\mathcal{N} \models T$ . Let  $F$  be a closed formula of  $\mathcal{L}_{\text{PA}}$  such that  $\mathcal{M} \not\models F$ . Then, since the satisfaction of a first-order formula on  $\mathcal{N}$  involves only its first-order part, we have  $\mathcal{N} \not\models F$ . Thus, by Proposition 6.3,  $T$  is a conservative extension of  $S$  for closed formulas of  $\mathcal{L}_{\text{PA}}$ .  $\blacksquare$

#### First order vs arithmetic formula

Be careful to distinguish formulas in  $\mathcal{L}_{\text{PA}}$  from arithmetic formulas in  $\mathcal{L}_{Z_2}$ . Indeed, in the second case, the free variables can be sets of integers. We thus distinguish the principle  $\text{IS}_n^0$ , which is a second-order theory, where the formulas can have free set variables, from  $\text{IS}_n$ , which is a first-order theory.

Our first conservation result shows that the first-order part of  $\text{ACA}_0$  is Peano arithmetic. In other words, the only formulas of  $\mathcal{L}_{\text{PA}}$  provable using  $\text{ACA}_0$  are the theorems of PA. We will need the following definition.

**Definition 6.7.** Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  be an  $\mathcal{L}_{Z_2}$ -structure and  $\Gamma$  a collection of formulas from  $\mathcal{L}_{Z_2}$  with parameters in  $\mathcal{M}$ , without free set variable, and having only  $x$  for free integer variable. A set  $X \subseteq M$  is  $\Gamma$ -definable in  $\mathcal{M}$  if  $X = \{n \in M : \mathcal{M} \models F(n)\}$  for a formula  $F \in \Gamma$ . A set  $X$  is  $\Delta_n^0$ -definable in  $\mathcal{M}$  if it is both  $\Sigma_n^0$ -definable and  $\Pi_n^0$ -definable.  $\diamond$

The previous definition also applies to first-order structures by considering their trivial  $\omega$ -extension whose second-order part is empty.

**Theorem 6.8 (Friedman [64])**

$\text{ACA}_0$  is a conservative extension of PA.

PROOF. By Proposition 6.6, it suffices to prove that for any countable model  $\mathcal{M} \models \text{PA}$ , there is an  $\omega$ -extension  $\mathcal{N}$  of  $\mathcal{M}$  such that  $\mathcal{N} \models \text{ACA}_0$ . Let  $\mathcal{M} = (M, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  be a countable model of PA. Let  $S^{\mathcal{N}}$  be the class of sets  $X \subseteq M$  definable by a formula of  $\mathcal{L}_{\text{PA}}$  with parameters in  $\mathcal{M}$ . Note that the parameters are only first-order. Let  $\mathcal{N}$  be the  $\omega$ -extension of  $\mathcal{M}$  whose second-order part is  $S^{\mathcal{N}}$ . Let us show that  $\mathcal{N} \models \text{ACA}_0$ . By Exercise 22-2.2, it suffices to show that  $\mathcal{N}$  satisfies Robinson arithmetic (Q) augmented with the axiom of induction on sets (11) and the comprehension scheme for arithmetic formulas with parameters.

Since  $\mathcal{N}$  is an  $\omega$ -extension of  $\mathcal{M}$  and  $\mathcal{M} \models \text{PA}$ , then  $\mathcal{N} \models \text{PA}$ , so  $\mathcal{N} \models \text{Q}$ . Let  $X \in S^{\mathcal{N}}$  and let  $F$  be the formula of  $\mathcal{L}_{\text{PA}}$  defining  $X$  in  $\mathcal{M}$ . As  $\mathcal{N} \models \text{PA}$ , then the induction scheme is satisfied for  $F$ , so the induction axiom is satisfied for  $X$ .

Let us show that  $\mathcal{N}$  satisfies the comprehension scheme for arithmetic formulas with parameters in  $\mathcal{N}$ . Let  $G(x)$  be an arithmetic formula of  $\mathcal{L}_{Z_2}$  with parameters in  $\mathcal{N}$ . Let  $X_0, \dots, X_{k-1}$  be the second-order parameters of  $G$ , and let  $F_0, \dots, F_{k-1}$  be the formulas of  $\mathcal{L}_{\text{PA}}$  defining them. Let  $H(x)$  be the formula of  $\mathcal{L}_{Z_2}$  where we have replaced all occurrences of  $y \in X_s$  by  $F_s(y)$ , so that  $H$  only has first-order parameters. In particular,  $H$  is a formula of  $\mathcal{L}_{\text{PA}}$  with parameters in  $\mathcal{M}$ , so  $Y = \{n \in M : \mathcal{M} \models H(n)\} \in S^{\mathcal{N}}$ , but  $Y = \{n \in M : \mathcal{N} \models G(n)\}$ , so  $\mathcal{N} \models \exists X \forall n (n \in X \leftrightarrow G(n))$ . ■

The second conservation result concerns the first-order part of  $\text{RCA}_0$ . We denote by  $\Sigma_1\text{-PA}$  the first-order theory composed of Robinson arithmetic Q, augmented with the  $\Sigma_1$  induction scheme for the formulas of  $\mathcal{L}_{\text{PA}}$  with first-order parameters ( $I\Sigma_1$ ). We will need the following lemma left as an exercise:

**Exercise 6.9.** Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  be a structure such that  $\mathcal{M} \models \text{BS}_1^0$  and let  $X \subseteq M$  be a  $\Delta_1^0$ -definable set with parameters in  $\mathcal{M}$ . For any  $\Sigma_1^0$  formula  $F$  with parameters in  $\mathcal{M} \cup \{X\}$ , there is a  $\Sigma_1^0$  formula  $G$  with parameters in  $\mathcal{M}$ , having the same free variables as  $F$ , and such that  $\mathcal{M} \cup \{X\} \models \forall x (F(x) \leftrightarrow G(x))$ . ◇

We are now ready to prove our conservation result.

**Theorem 6.10 (Friedman [64])**

$\text{RCA}_0$  is a conservative extension of  $\Sigma_1\text{-PA}$ .

PROOF. By Proposition 6.6, it suffices to prove that for any countable model  $\mathcal{M} \models \Sigma_1\text{-PA}$ , there is an  $\omega$ -extension  $\mathcal{N}$  of  $\mathcal{M}$  such that  $\mathcal{N} \models \text{RCA}_0$ . Let  $\mathcal{M} = (M, +^{\mathcal{M}}, \times^{\mathcal{M}}, <^{\mathcal{M}}, 0^{\mathcal{M}}, 1^{\mathcal{M}})$  be a countable model of  $\Sigma_1\text{-PA}$ . Let  $S^{\mathcal{N}}$  be the class of sets  $X \subseteq M$  definable by a  $\Delta_1$  formula of  $\mathcal{L}_{\text{PA}}$  with parameters in  $\mathcal{M}$ . Note that the parameters are only first-order. Let  $\mathcal{N}$  be the  $\omega$ -extension of  $\mathcal{M}$  whose second-order part is  $S^{\mathcal{N}}$ .

By iterating Exercize 6.9 to remove at each iteration a second-order parameter, we can show that if  $F(x)$  is a  $\Sigma_1^0$  formula with parameters in  $\mathcal{N}$  and without free set variable, there is a  $\Sigma_1$  formula  $G(x)$  with parameters in  $\mathcal{M}$ , with the same free variables, and such that  $\mathcal{N} \models \forall x (F(x) \leftrightarrow G(x))$ . Indeed, if  $F(x)$  has as second-order parameters  $X_1, \dots, X_k$ , we can successively construct  $\Sigma_1^0$  formulas  $F_1(x), \dots, F_{k-1}(x)$  such that  $F_i(x)$  has as second-order parameters  $X_1, \dots, X_{k-i-1}$ , and such that  $\mathcal{N} \models \forall x (F(x) \leftrightarrow F_i(x))$ . The last formula  $F_{k-1}(x)$  is the desired formula  $G(x)$ .

Let us show that  $\mathcal{N} \models \text{RCA}_0$ . Since  $\mathcal{N}$  is an  $\omega$ -extension of  $\mathcal{M}$  and  $\mathcal{M} \models \Sigma_1\text{-PA}$ , then  $\mathcal{N} \models \Sigma_1\text{-PA}$ , so  $\mathcal{N} \models \text{Q}$ . Let us show that  $\mathcal{N} \models \text{IS}_1^0$ . Let  $F(x)$  be a  $\Sigma_1^0$  formula closed with parameters in  $\mathcal{N}$ . Let  $G(x)$  be a  $\Sigma_1$  formula with parameters in  $\mathcal{M}$ , such that  $\mathcal{N} \models \forall x (F(x) \leftrightarrow G(x))$ . As  $\mathcal{M} \models \text{IS}_1$ , then the induction scheme applies in  $\mathcal{M}$  for  $G$ , so applies in  $\mathcal{N}$  for  $G$ , so in  $\mathcal{N}$  for  $F$ . Thus,  $\mathcal{N} \models \text{IS}_1^0$ . Let us show that  $\mathcal{N}$  satisfies the comprehension scheme for the closed  $\Delta_1^0$  formulas with parameters in  $\mathcal{N}$ , and with no free variable other than  $x$ . Let  $F_{\exists}(x)$  be a  $\Sigma_1^0$  formula with parameters in  $\mathcal{N}$  and  $F_{\forall}$  a  $\Pi_1^0$  formula with parameters in  $\mathcal{N}$ , such that  $\mathcal{N} \models \forall x (F_{\exists}(x) \leftrightarrow F_{\forall}(x))$ . Let  $G_{\exists}$  and  $G_{\forall}$  be the corresponding  $\Sigma_1$ , and  $\Pi_1$  formulas with parameters in  $\mathcal{M}$  using Exercize 6.9. Then,  $\mathcal{M} \models \forall x (G_{\exists}(x) \leftrightarrow G_{\forall}(x))$ , therefore the set  $X = \{n \in M : G_{\exists}(n)\}$  is definable by a  $\Delta_1$  formula with parameters in  $\mathcal{M}$ , therefore  $X \in S^{\mathcal{N}}$ . So  $\mathcal{N} \models \exists X \forall z (z \in X \leftrightarrow F_{\exists}(z))$ , so  $\mathcal{N}$  satisfies the comprehension scheme for closed  $\Delta_1^0$  formulas with parameters in  $\mathcal{N}$ . Thus  $\mathcal{N} \models \text{RCA}_0$ . ■

## 6.2. $\Pi_1^1$ conservation

We have so far studied the first-order part of the theories  $\text{ACA}_0$  and  $\text{RCA}_0$ , showing that they are conservative extensions of  $\text{PA}$  and  $\Sigma_1\text{-PA}$ , respectively. Note that these are extensions of first-order theories into second-order theories.

In this section, we are going to change our strategy. Suppose we have two theories  $S \subseteq T$ , both in the language of second-order arithmetic. Suppose also that the first-order part of  $S$  is known. In order to know the first-order part of  $T$ , it suffices to show that the latter is a conservative extension of  $S$  for arithmetic formulas. It will thus have the same first-order part as  $S$ .

The technique that we will expose will in fact work directly for all the  $\Pi_1^1$  formulas, that is to say the formulas of the form  $\forall X F(X)$  where  $F$  is an arithmetic formula and  $X$  a second-order variable (we will study the formulas and  $\Pi_1^1$  classes in detail in Chapter 29).

We are going to prove in particular that  $\text{WKL}_0$  is a conservative extension of  $\text{RCA}_0$  for the  $\Pi_1^1$  formulas, and therefore that the first-order part of  $\text{WKL}_0$  is  $\Sigma_1$ -PA. Let us start by extending the notion of  $\omega$ -extension to second-order structures.

**Definition 6.11.** Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  and  $\mathcal{N} = (N, S^{\mathcal{N}})$  be two structures of  $\mathcal{L}_{Z_2}$ . We say that  $\mathcal{N}$  is a  $\omega$ -extension of  $\mathcal{M}$  if  $\mathcal{M} \upharpoonright_{\mathcal{L}_{\text{PA}}} = \mathcal{N} \upharpoonright_{\mathcal{L}_{\text{PA}}}$  and  $S^{\mathcal{M}} \subseteq S^{\mathcal{N}}$ . We will then also say that  $\mathcal{M}$  is a  $\omega$ -substructure of  $\mathcal{N}$ .  $\diamond$

We now see how to use the notion of  $\omega$ -extension to prove that two theories show the same  $\Pi_1^1$  formulas, and therefore the same arithmetic formulas.

**Proposition 6.12.** Let  $S$  and  $T \supseteq S$  be  $\mathcal{L}_{Z_2}$ -theories such that any countable model  $\mathcal{M}$  of  $S$  is an  $\omega$ -substructure of a countable model  $\mathcal{N}$  from  $T$ . Then,  $T$  is a conservative extension of  $S$  for closed  $\Pi_1^1$  formulas.  $\star$

PROOF. Let  $\mathcal{M}$  be a countable model of  $S$  and let  $\mathcal{N}$  be an  $\omega$ -extension of  $\mathcal{M}$  such that  $\mathcal{N} \models T$ . Let  $S^{\mathcal{M}}$  and  $S^{\mathcal{N}}$  be the second-order parts of  $\mathcal{M}$  and  $\mathcal{N}$ , respectively. Let  $F$  be a closed  $\Pi_1^1$  formula of the form  $\forall X G(X)$ , where  $G(X)$  is an arithmetic formula, such that  $\mathcal{M} \not\models F$ . Then,  $\mathcal{M} \models \exists X \neg G(X)$ , so there is an  $X \in S^{\mathcal{M}}$  such that  $\mathcal{M} \models G(X)$ . As  $S^{\mathcal{M}} \subseteq S^{\mathcal{N}}$ ,  $X \in S^{\mathcal{N}}$ , and as  $\mathcal{M} \upharpoonright_{\mathcal{L}_{\text{PA}}} = \mathcal{N} \upharpoonright_{\mathcal{L}_{\text{PA}}}$ , then  $\mathcal{N} \models G(X)$ , so  $\mathcal{N} \models \exists X \neg G(X)$ , and  $\mathcal{N} \models \neg F$ . Thus, by Proposition 6.3,  $T$  is a conservative extension of  $S$  for closed  $\Pi_1^1$  formulas.  $\blacksquare$

As expressed in Section 22-8.1, a certain number of statements are expressed in the form  $\forall X (F(X) \rightarrow \exists Y H(X, Y))$ , where  $F$  and  $H$  are arithmetic formulas possibly with free variables. For example, for weak König's lemma,  $F(X)$  is the predicate “ $X$  codes for an infinite binary tree” and  $H(X, Y)$  means “ $Y$  is a path of the tree coded by  $X$ .” We will develop general techniques to create models of statements of the above form, while guaranteeing the conservation of  $\Pi_1^1$  formulas.

**Definition 6.13.** Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  be a structure of second-order arithmetic and let  $G \subseteq M$ . We denote by  $\mathcal{M} \cup \{G\}$  the  $\omega$ -extension of  $\mathcal{M}$  whose second-order part is  $S^{\mathcal{M}} \cup \{G\}$ , and  $\mathcal{M}[G]$  the  $\omega$  extension of  $\mathcal{M}$  whose part of the second-order is the collection of sets which are definable by a  $\Delta_1^0$  predicate with parameters in  $\mathcal{M} \cup \{G\}$ .  $\diamond$

Let us take our example above of a statement of the form  $\forall X (F(X) \rightarrow \exists Y H(X, Y))$ . Intuitively, starting from a structure  $\mathcal{M} \models \text{RCA}_0$  and

a set  $X \in \mathcal{M}$  such that  $F(X)$ , we want to create an  $\omega$ -extension  $\mathcal{M}_1$  containing a set  $G \subseteq M$  such that  $\mathcal{M}_1 \models H(X, G)$ , so that the statement  $F(X) \rightarrow \exists Y H(X, Y)$  is true in  $\mathcal{M}_1$ . By iterating the process, we will obtain a sequence of  $\omega$ -extensions  $\mathcal{M} \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \dots$  such that  $\bigcup_n \mathcal{M}_n$  is a model of  $\text{RCA}_0$  and  $\forall X (F(X) \rightarrow \exists Y H(X, Y))$ .

To satisfy a step, we will therefore build a set  $G \subseteq M$  such that  $\mathcal{M} \cup \{G\} \models H(X, G)$  and add it to  $\mathcal{M}$ . The structure  $\mathcal{M} \cup \{G\}$  obtained does not satisfy the  $\Delta_1^0$  comprehension scheme in general. We must therefore close the structure  $\mathcal{M} \cup \{G\}$  under this scheme, which yields  $\mathcal{M}[G]$ . All the “work” will consist in ensuring that the model  $\mathcal{M}[G]$  thus obtained will not sabotage  $\text{RCA}_0$ , and in particular the  $\Sigma_1^0$  induction scheme. It is therefore necessary to ensure that  $\mathcal{M}[G] \models \text{IS}_1^0$ . The following proposition shows that it suffices to prove that  $\mathcal{M} \cup \{G\} \models \text{IS}_1^0$  to obtain  $\mathcal{M}[G] \models \text{IS}_1^0$  and therefore  $\mathcal{M}[G] \models \text{RCA}_0$ .

**Proposition 6.14.** Let  $T \equiv \forall X (F(X) \rightarrow \exists Y H(X, Y))$  where  $F$  and  $H$  are arithmetic formulas. Suppose that for any countable model  $\mathcal{M}$  of  $\text{RCA}_0$ , and any set  $X \in \mathcal{M}$  such that  $\mathcal{M} \models F(X)$ , there exists a set  $G \subseteq M$  such that  $\mathcal{M} \cup \{G\} \models \text{IS}_1^0$  and  $\mathcal{M} \cup \{G\} \models H(X, G)$ . Then,  $\text{RCA}_0 + T$  is a conservative extension of  $\text{RCA}_0$  for the  $\Pi_1^1$  formulas. ★

PROOF. Let  $\mathcal{M}$  be a countable model of  $\text{RCA}_0$ . We will define a sequence  $\mathcal{M} = \mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \dots$  of  $\omega$ -countable extensions such that for all  $n \in \mathbb{N}$ ,

- (1)  $\mathcal{M}_n \models \text{RCA}_0$
- (2) For any  $X \in \mathcal{M}_n$  such that  $\mathcal{M}_n \models F(X)$ , there is  $m \in \mathbb{N}$  and  $Y \in \mathcal{M}_m$  such that  $\mathcal{M}_m \models H(X, Y)$

Let us show that if this is the case, then  $\text{RCA}_0 + T$  is a conservative extension of  $\text{RCA}_0$  for the  $\Pi_1^1$  formulas. Let  $\mathcal{N} = \bigcup_n \mathcal{M}_n$ . By (1),  $\mathcal{N} \models \text{RCA}_0$ , and by (2), as  $F$  and  $H$  are arithmetic formulas and as  $\mathcal{N}$  is an  $\omega$ -extension of  $\mathcal{M}_n$  for all  $n \in \mathbb{N}$ ,  $\mathcal{N} \models \forall X (F(X) \rightarrow \exists Y H(X, Y))$ . By Proposition 6.12,  $\text{RCA}_0 + T$  is a conservative extension of  $\text{RCA}_0$  for  $\Pi_1^1$  formulas.

Suppose we have constructed  $\mathcal{M}_n$ , and let  $X \in \mathcal{M}_n$  be a set such that  $\mathcal{M}_n \models F(X)$ . Let  $G \subseteq M$  be such that  $\mathcal{M}_n \cup \{G\} \models \text{IS}_1^0$  and  $\mathcal{M}_n \cup \{G\} \models H(X, G)$ . Let us show that  $\mathcal{M}_{n+1} = \mathcal{M}_n[G]$  satisfies (1) and (2) above. By construction,  $\mathcal{M}_n[G] \models H(X, G)$  and  $\mathcal{M}_n[G]$  satisfies the  $\Delta_1^0$  comprehension scheme. Let us show that  $\mathcal{M}_n[G] \models \text{IS}_1^0$ . Let  $P(x)$  be a  $\Sigma_1^0$  formula with parameters in  $\mathcal{M}_n[G]$ , with no free variable other than  $x$ . By iterating Exercize 6.9 as in the proof of Theorem 6.10, there exists a  $\Sigma_1^0$  formula  $Q(x)$  with parameters in  $\mathcal{M}_n \cup \{G\}$  such that  $\mathcal{M}_n[G] \models \forall x (P(x) \leftrightarrow Q(x))$ . As  $\mathcal{M}_n \cup \{G\} \models \text{IS}_1^0$ , then the induction scheme is satisfied for  $Q$ , so for  $P$ . Thus,  $\mathcal{M}_n[G] \models \text{IS}_1^0$ . So  $\mathcal{M}_n[G] \models \text{RCA}_0$ . This completes the construction.

Note that we must pay attention to the choice of  $X$  so that (2) is satisfied. More precisely, if we fix an enumeration  $\mathcal{M}_n = \{X_0^n, X_1^n, \dots\}$  of all the sets of  $\mathcal{M}_n$ , in step  $n$ , we choose  $X_m^n$  for the smallest  $\langle n, m \rangle$  which has not yet been satisfied. ■

We now have the elements necessary to prove the announced theorem:  $\text{WKL}_0$  is a conservative extension of  $\text{RCA}_0$  for the  $\Pi_1^1$  formulas. In particular, we can deduce that the first-order part of  $\text{WKL}_0$  is the same as that of  $\text{RCA}_0$ :  $\Sigma_1$ -PA.

**Theorem 6.15 (Harrington (voir Simpson [202]))**

*$\text{WKL}_0$  is a conservative extension of  $\text{RCA}_0$  for  $\Pi_1^1$  formulas.*

Before proving Theorem 6.15, we need to prove a technical lemma on the  $\Sigma_1^0$  formulas. By Theorem 9-3.4, a set  $A \subseteq \mathbb{N}$  is  $X$ -c.e. if it is definable by a  $\Sigma_1^0(X)$  formula of  $\mathcal{L}_{Z_2}$ . In particular, for any  $\Sigma_1^0$  formula  $F(X, x)$ , there exists a Turing functional  $\Phi_e$  such that  $\{n : F(X, n)\} = \{n : \Phi_e^X(n) \downarrow\}$ . It follows from the use property that  $\{n : F(X, n)\} = \{n : \exists k \Phi_e^{X \upharpoonright k}(n)[k] \downarrow\}$ . In other words, if we define the  $\Delta_0^0$  formula  $H(\sigma, x) \equiv \Phi_e^\sigma(x)[|\sigma|]$ , we have

$$\{n : F(X, n)\} = \{n : \exists k H(G \upharpoonright_k, n)\}$$

The following lemma shows that this equivalence can be formalized in  $\text{BS}_1^0$ :

**Exercize 6.16** (Simpson [202]). (★) Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  be an  $\mathcal{L}_{Z_2}$ -structure such that  $\mathcal{M} \models \text{BS}_1^0$ . Let  $F(X)$  be a  $\Sigma_1^0$  formula with parameters in  $\mathcal{M}$ , and with no free set variable other than  $X$ . Then, there exists a  $\Delta_0^0$  formula  $H(\sigma)$  with the same parameters as  $F$ , without free set variable, and with the same free integer variables as  $F$  plus a free integer variable  $\sigma$ , such that for any set  $G \subseteq M$ ,

$$\mathcal{M} \cup \{G\} \models F(G) \leftrightarrow \exists k H(G \upharpoonright_k) \quad \diamond$$

We come to the proof of Theorem 6.15. Let  $\mathcal{M} = (M, S^{\mathcal{M}})$  be a countable model of  $\text{RCA}_0$  and let  $T \in S^{\mathcal{M}}$  be an  $\mathcal{M}$ -infinite binary tree (see Section 5.2). We are going to construct an  $\mathcal{M}$ -infinite path  $G \in [T]$  by forcing  $\mathbb{P}$  on the  $\mathcal{M}$ -infinite subtrees of  $T$  in  $S^{\mathcal{M}}$ . Let us start by showing that any filter  $\mathcal{F}$  sufficiently generic for  $\mathbb{P}$  induces a unique path  $G \in \bigcap_{T \in \mathcal{F}} [T]$ .

**Lemma 6.17.** For any  $n \in M$ , the set  $D_n$  of trees  $S \in \mathbb{P}$  containing only one string of length  $n$  is dense in  $\mathbb{P}$ . ★

PROOF. Let  $T \in \mathbb{P}$  be an  $\mathcal{M}$ -infinite binary tree. For all  $\sigma \in 2^n$ , let  $S_\sigma = \{\tau \in T : \sigma \preceq \tau \vee \sigma \succ \tau\}$ . Let us show that there exists  $\sigma \in 2^n$  such that  $S_\sigma$

is  $\mathcal{M}$ -infinite. Indeed, if all the sets  $S_\sigma$  are  $\mathcal{M}$ -finite, we have:

$$\forall \sigma \in 2^n \exists m \forall \tau \in 2^m (\sigma \prec \tau \rightarrow \tau \notin T)$$

We use  $\text{B}\Sigma_1^0$  (which is a consequence of  $\text{RCA}_0$ ) to obtain:

$$\exists m \forall \sigma \in 2^n \exists m' < m \forall \tau \in 2^{m'} (\sigma \prec \tau \rightarrow \tau \notin T)$$

And by downward-closure of  $T$ , we get  $\exists m \forall \tau \in 2^m \tau \notin T$ , so  $T$  is  $\mathcal{M}$ -finite, which contradicts our hypothesis on  $T \in \mathbb{P}$ . ■

Recall that a condition  $T$  forces a  $\Sigma_1^0$  or  $\Pi_1^0$  formula  $F(G)$  if  $\mathcal{M} \models F(P)$  for all  $P \in [T]$ . Our goal is to “reduce” the  $\Sigma_1^0$  induction from  $\mathcal{M} \cup \{G\}$  to that of  $\mathcal{M}$  using the forcing relation. Indeed, the forcing relation for  $\Sigma_1^0$  properties is  $\Sigma_1^0$  on  $\mathcal{M}$ . More precisely, we will use the principle  $\text{L}\Pi_1^0$ , equivalent to  $\text{I}\Sigma_1^0$ , which states that any non-empty  $\Pi_1^0$  subset of  $\mathcal{N}$  admits a minimum element.

To do this, suppose that  $F(x, G)$  is a  $\Sigma_1^0$  formula such that  $T \Vdash \neg F(b, G)$  for a condition  $T \in \mathbb{P}$  in our filter and a  $b \in \mathcal{M}$ . It follows that the set  $\{a \in \mathcal{M} : \neg F(a, G)\}$  will be non-empty, where  $G$  is the generic set constructed. For  $F$  to satisfy the  $\text{L}\Pi_1^0$  principle, it will be necessary to ensure that there is a minimum  $b_1$  element in  $\mathcal{M}$  such that  $\neg F(b_1, G)$  is true. One way to ensure this is to find an extension  $S \in \mathbb{P}$  such that for any  $a <^{\mathcal{M}} b$ ,  $S \Vdash F(a, G)$  or  $S \Vdash \neg F(a, G)$ . Indeed, the set  $E = \{a \leq^{\mathcal{M}} b : S \Vdash \neg F(a, G)\}$  is  $\Pi_1^0$  and non-empty in  $\mathcal{M}$ , and must therefore satisfy the principle  $\text{L}\Pi_1^0$  in the starting model  $\mathcal{M}$ . We therefore deduce that there is a minimum  $b_1$  in  $\mathcal{M}$  such that  $S \Vdash \neg F(b_1, G)$ , so that  $\text{L}\Pi_1^0$  is satisfied for  $F$ .

Given a condition  $T \in \mathbb{P}$  and a  $\Sigma_1^0$  formula  $F(G)$ , it is easy to find an extension  $S$  forcing either  $F(G)$  or  $\neg F(G)$ . The natural approach to preserving  $\text{L}\Pi_1^0$  would be to try to create a decreasing sequence of extensions  $T \supseteq T_1 \supseteq T_2 \supseteq T_3 \cdots \supseteq T_b$  such that  $T_{a+1} \Vdash F(a, G)$  or  $T_{a+1} \Vdash \neg F(a, G)$  for all  $a <^{\mathcal{M}} b$ . However, we are working in an arbitrary countable model of  $\text{RCA}_0$ , which can contain non-standard integers  $b$ , and in particular such that the set  $\{a \in \mathcal{M} : a <^{\mathcal{M}} b\}$  is infinite. The goal of the following lemma is to show that we manage to simultaneously force  $F(a, G)$  or  $\neg F(a, G)$  for all  $a <^{\mathcal{M}} b$  with the same forcing condition, even though the set  $\{a \in \mathcal{M} : a <^{\mathcal{M}} b\}$  would be infinite.

**Lemma 6.18.** For any  $b \in M$  and any  $\Sigma_1^0$  formula  $F(x, G)$ , the set  $D_{b,F}$  of trees  $S \in \mathbb{P}$  such that for all  $a <^{\mathcal{M}} b$ ,  $S \Vdash F(a, G)$  or  $S \Vdash \neg F(a, G)$  is dense in  $\mathbb{P}$ . ★

PROOF. By Exercize 6.16,  $F(x, G) \equiv \exists z H(x, z, G \upharpoonright_z)$  for a  $\Delta_0^0$  formula  $H$ . For all  $\sigma \in 2^{\leq b}$ , let us define  $S_\sigma \subseteq T$  inductively as follows:  $S_\epsilon = S$ .

Suppose  $S_\sigma$  is defined. Then,  $S_{\sigma 0} = \{\rho \in S_\sigma : \forall z < |\rho| \neg H(a, z, \rho \upharpoonright_z)\}$  and  $S_{\sigma 1} = S_\sigma$ . Let

$$U = \{\sigma \in 2^b : S_\sigma \text{ is } \mathcal{M}\text{-infinite}\}$$

Note that  $U$  is not empty because  $S_{111\dots 1} = S$  is  $\mathcal{M}$ -infinite. In addition,  $U$  is  $\Pi_1^0$  in  $\mathcal{M}$  because  $S_\sigma$  is closed under prefix for all  $\sigma$ :

$$U = \{\sigma \in 2^b : \forall n \in \mathcal{M} \exists \tau \in 2^n \tau \in S_\sigma\}$$

By  $\text{L}\Pi_1^0$ ,  $U$  contains a minimum element  $\sigma$  for the lexicographic order.

Let us show that  $S_\sigma \in D_{b,F}$ . Let  $a <^{\mathcal{M}} b$ .

- If  $\sigma(a) = 0$ , then  $\forall \rho \in S_\sigma \forall z < |\rho| \neg H(a, z, \rho \upharpoonright_z)$ . Then, for all  $G \in [S_\sigma]$  we have  $\forall z \neg H(a, z, G \upharpoonright_z)$ , in other words  $\neg F(a, G)$ . So  $S_\sigma \Vdash \neg F(a, G)$ .
- If  $\sigma(a) = 1$ , then by minimality of  $\sigma$  in  $U$ , the string  $\sigma \upharpoonright_a 0111\dots 1$  of length  $b$  is not in  $U$ , so  $S_{\sigma \upharpoonright_a 0}$  is  $\mathcal{M}$ -finite. It follows that there exists a  $k \in M$  such that for all  $\rho \in 2^k$ , if  $\rho \in S_{\sigma \upharpoonright_a}$  then  $\exists z < k H(a, z, \rho \upharpoonright_z)$ . Thus, for all  $G \in [S_{\sigma \upharpoonright_a}]$ , there exists  $z < k$  such that  $H(a, z, G \upharpoonright_z)$ , therefore  $S_{\sigma \upharpoonright_a} \Vdash F(a, G)$ , or  $S_\sigma \subseteq S_{\sigma \upharpoonright_a}$ , therefore  $S_\sigma \Vdash F(a, G)$ . ■

We are now ready to put the pieces of the puzzle together to prove Theorem 6.15.

**PROOF OF THEOREM 6.15.** By Proposition 6.14, it suffices to show that for any countable model  $\mathcal{M} = (M, S^{\mathcal{M}})$  of  $\text{RCA}_0$  and any binary tree  $\mathcal{M}$ -infinity  $T \in \mathcal{M}$ , there exists a set  $G \subseteq M$  such that  $G \in [T]$  and  $\mathcal{M} \cup \{G\} \models \text{L}\Pi_1^0$ . Let us fix  $\mathcal{M}$  and  $T \in \mathcal{M}$ . Let  $\mathbb{P}$  be the collection of  $\mathcal{M}$ -infinite subtrees of  $T$  in  $S^{\mathcal{M}}$ , ordered by inclusion, and let  $\mathcal{F}$  be a sufficiently generic filter for  $\mathbb{P}$ . By Lemma 6.17, there exists a unique set  $G \subseteq M$  such that  $G \in \bigcap_{S \in \mathcal{F}} [S]$ . In particular,  $T \in \mathcal{F}$ , so  $G \in [T]$ .

Let us show that  $\mathcal{M} \cup \{G\} \models \text{L}\Pi_1^0$ . Let  $F(x, G)$  be a  $\Sigma_1^0$  formula such that  $\mathcal{M} \cup \{G\} \models \neg F(b, G)$  for an element  $b \in M$ . By Lemma 6.18, there exists a condition  $S \in \mathcal{F}$  such that for any  $a \leq^{\mathcal{M}} b$ ,  $S \Vdash F(a, G)$  or  $S \Vdash \neg F(a, G)$ . So we have

$$\{a \leq^{\mathcal{M}} b : \mathcal{M} \cup \{G\} \models F(a, G)\} = \{a \leq^{\mathcal{M}} b : S \Vdash F(a, G)\}$$

and the second predicate is  $\Sigma_1^0$  in  $\mathcal{M}$ . As  $\neg F(b, G)$ , then  $S \Vdash \neg F(b, G)$ , so by  $\text{L}\Pi_1^0$  applied to the  $\Sigma_1^0$  predicate  $S \Vdash F(x, G)$  with a free variable  $x$ , there exists a smallest  $a \leq^{\mathcal{M}} b$  such that  $S \Vdash \neg F(a, G)$ . In particular,  $a$  is the smallest element such that  $\neg F(a, G)$  is satisfied. This completes the proof of Theorem 6.15. ■

**Corollary 6.19**

$\text{WKL}_0$  is a conservative extension of  $\Sigma_1\text{-PA}$ .

PROOF. Let  $F$  be a closed formula of  $\mathcal{L}_{\text{PA}}$  such that  $\Sigma_1\text{-PA} \not\models F$ . By Theorem 6.10,  $\text{RCA}_0 \not\models F$ , and by Theorem 6.15,  $\text{WKL}_0 \not\models F$ . ■

## 7. Hilbert program

Let us return to the original motivations of Reverse Mathematics. As we mentioned in Chapter 9, mathematics experienced a great foundational crisis at the beginning of the 20th century, with Cantor's development of a theory of infinity, marking a further step in mathematical abstraction and its distance from reality. The difficulty, if not the impossibility, of attaching infinity to a sensitive reality, as well as the paradoxes on which mathematicians have stumbled, naturally aroused mistrust. Thus the crisis of the foundations has encouraged the development of finitist thoughts. At that time, scientific advances tended to show the finitude of the real world, through the discovery of the atom (the infinitely small would therefore not exist) or of the finitude of the universe (the infinitely large neither.) Influential mathematicians like Poincaré, Brouwer or Kronecker then advocated the restriction to finitary mathematics.

### 7.1. Formal interpretation of Hilbert's program

David Hilbert proposed in 1900 a program intended to restore confidence in the foundations of mathematics, the objective of which was to found mathematics in all its generality, on purely finite bases. This program can be broken down into three stages:

- (1) Formalize infinite mathematics  $T_\infty$ .
- (2) Identify the finite fragment  $T_{fin}$  of mathematics.
- (3) Show that any statement, or at least any statement in a certain class  $\mathcal{C}$ , provable in  $T_\infty$  is provable in  $T_{fin}$ .

Hilbert having left points of uncertainty as to the formal content of his program, Steve Simpson [201] proposed the following interpretation.

**Theory  $T_\infty$ .** Simpson suggests the choice of  $\text{Z}_2$  as the reference theory for infinite mathematics. This choice is justified by the work of Hilbert and Bernays, *Grundlagen der Mathematik* (1934-1936), which show that

the great majority of usual mathematics can be formalized in second-order arithmetic.

**Theory  $T_{fin}$ .** A first idea would be to choose  $\text{RCA}_0$  as the finite reference fragment. However the induction scheme, even restricted to  $\Sigma_1^0$  formulas constitutes an infinite hypothesis. It is unfortunately necessary to formalize the notion of computable function within arithmetic. A more elementary system is however possible: that known as *primitive recursive arithmetic* (PRA) generally obtains consensus (see Tait [219]). The idea is as follows, we restrict the induction to its strict minimum, i.e., to formulas without quantifiers (the system  $\text{I}_{\text{open}}$  defined in Section 2.2). In order to keep some working tools in spite of everything, we add function symbols in our language making it possible to directly handle other usual functions than addition and multiplication, and first of all the exponential function. In fact, we allow ourselves all the primitive recursive functions, as defined in Section 6-1.

## 7.2. Some intuitions on the system PRA

Let us recall here the definition of the class of primitive recursive functions denoted by  $\mathcal{PR}$ . This is the smallest class of functions from  $\mathbb{N}^n$  to  $\mathbb{N}$  for any  $n \in \mathbb{N}$ , containing the following basic functions:

- (a) The successor function:  $\text{succ}(x) = x + 1$
- (b) The constant functions:  $c_m^n(x_1, \dots, x_n) = m$  for all  $n, m \geq 0$
- (c) The projections:  $p_i^n(x_1, \dots, x_n) = x_i$  for all  $n$  and all  $i \in 1, \dots, n$

and which is closed under the following operations:

- (i) Composition: if  $g_1, \dots, g_m \in \mathcal{PR}$  are functions of  $n$  variables and  $h \in \mathcal{PR}$  is a function of  $m$  variables, then the function

$$f(\bar{x}) = h(g_1(\bar{x}), \dots, g_m(\bar{x}))$$

is in  $\mathcal{PR}$ .

- (ii) Primitive recursion: if  $g, h \in \mathcal{PR}$  are functions of respectively  $n$  and  $n+2$  variables, the function  $f$  defined by  $f(\bar{x}, 0) = g(\bar{x})$  and  $f(\bar{x}, m+1) = h(\bar{x}, m, f(\bar{x}, m))$  is a function of  $n+1$  variables in  $\mathcal{PR}$ .

The reader can refer to Section 6-1 for an argument that primitive recursive functions are computable (and by definition total) and to Section 6-3 for a formal study of these functions and finally to Theorem 4.6 for a proof that these functions are all provably total in  $\text{RCA}_0$ . However, this is only a strict subclass of the class of all total computable functions, and it is necessary

to add the closure under the minimization scheme in order to obtain them all.

If the class  $\mathcal{PR}$  does not contain all the computable functions, it nevertheless contains a large number of usual functions, such as addition, multiplication, exponential function, but also encoding and decoding functions, such as the Cantor pairing function. It therefore forms a robust class of computable functions with good closure properties.

The system PRA is therefore placed in a language containing a symbol for each primitive recursive function. Its axioms include the induction scheme  $I_{\text{open}}$ , and in the same way that the axioms of  $\mathbf{Q}$  regulate the behavior of addition and multiplication, those of PRA will regulate each of the primitive recursive functions. So for example, if  $f, h, g_1, \dots, g_m$  are function symbols such that  $f$  must be interpreted by  $\bar{x} \mapsto h(g_1(\bar{x}), \dots, g_m(\bar{x}))$ , then  $\forall \bar{x} f(\bar{x}) = h(g_1(\bar{x}), \dots, g_m(\bar{x}))$  will be an axiom.

The following results, the proofs of which are beyond the scope of this work, show that the system PRA is sufficient to show all the  $\Pi_2$  statements provable in  $\text{WKL}_0$ .

**Theorem 7.1 (Friedman (voir Simpson [202]))**

*$\text{RCA}_0$  is a conservative extension of PRA for the  $\Pi_2$  statements of  $\mathcal{L}_{\text{PA}}$ .*

Note that if we are completely formal, the above theorem does not fall within the scope of conservation statements, for two reasons:

First of all, the language of second-order arithmetic is not an extension of that of primitive recursive arithmetic. Indeed, PRA includes function symbols for each primitive recursive function. However, as we have seen with Theorem 4.6, any primitive recursive function is definable by a formula of first-order arithmetic. We can therefore translate any formula of PRA into a formula of  $\mathcal{L}_{\text{PA}}$ , and a fortiori into a formula of second-order arithmetic.

Then, the  $\Pi_2$  statements of first-order arithmetic are not statements of primitive recursive arithmetic: indeed, PA has a function symbol for addition, one for multiplication, and a relationship symbol for the order. However, these two operations and the order relation are primitive recursive, and there is therefore a natural translation of the arithmetic statements into the language of PRA.

There is therefore a double translation of the statements from PA to PRA and vice versa, so that the formulas obtained by back and forth are provably equivalent to the starting formula. Theorem 7.1 gives us the following corollary.

**Corollary 7.2 (Friedman)**

*$\text{WKL}_0$  is a conservative extension of PRA for the  $\Pi_2$  statements of  $\mathcal{L}_{\text{PA}}$ .*

PROOF. Suppose that  $\text{WKL}_0 \vdash F$  for a  $\Pi_2$  statement  $F$ . In particular,  $F$  is  $\Pi_1^1$ , so  $\text{RCA}_0 \vdash F$  by Theorem 6.15. By Theorem 7.1,  $\text{PRA} \vdash F$ . ■

The connection between PRA and  $\text{RCA}_0$  goes further: the primitive recursive functions are exactly the total functions which are provably computable in  $\text{RCA}_0$ .

**Theorem (1.4 Parikh, see Hajek and Pudlak [80])**

*The  $\text{RCA}_0$ -provably computable functions are exactly the primitive recursive functions.*

This last result should be put in perspective with the assertion that  $\text{RCA}_0$  captures computable mathematics. From the point of view of Computability Theory, which is concerned with standard models, there exists a minimal  $\omega$ -model (for inclusion) of  $\text{RCA}_0$  which contains exactly the computable sets in the second-order. Thus, semantically,  $\text{RCA}_0$  captures computable mathematics. On the other hand, from the point of view of  $\text{RCA}_0$ , the only predicates that the theory can prove to be  $\Delta_1^0$  are the primitive recursive predicates, and thus  $\text{RCA}_0$  can only prove the existence of the primitive recursive sets. This difference comes from the fact that the notion of  $\Delta_1^0$  is relative to a theory or to a model, unlike the notion of  $\Sigma_1^0$  which is syntactic and therefore absolute.



# Chapter 24

## Computable reductions

As mentioned in Section 22-8.1, most of the theorems that we will study in this part are expressed in the form

$$\forall X (F(X) \rightarrow \exists Y G(X, Y))$$

where  $F$  and  $G$  are arithmetic formulas. We can then see the theorem as a mathematical problem, formulated in terms of instances and solutions. We will say that a set  $X$  is an *instance* of the problem if  $F(X)$  is true, and that  $Y$  is a *solution* to the instance  $X$  if  $G(X, Y)$  is true.

**Example 1.** König's lemma is the problem whose instances are infinite, finitely-branching trees, and for which the solutions of a tree are its infinite paths.

This kind of theorem concerns the existence of second-order objects, and the overwhelming majority of separations of the form  $\text{RCA}_0 + T_1 \not\vdash T_2$  where  $T_1, T_2$  are two such theorems, is done by constructing an  $\omega$ -model of  $\text{RCA}_0 + T_1$  which is not a model of  $T_2$ . This is what we will study in this chapter. We therefore focus on  $\omega$ -models in order to study the computational power of the theorems. Let's define what we mean by problem in all generality:

**Definition 2.** A *problem* is a relation  $P \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$ . An *instance* of  $P$  is an element of  $\text{dom } P = \{X \in 2^{\mathbb{N}} : \exists Y (X, Y) \in P\}$ . Given an instance  $X$  of  $P$ , we denote by  $P(X) = \{Y : (X, Y) \in P\}$  the class of *solutions* to  $X$ .  $\diamond$

For any true formula  $\forall X (F(X) \rightarrow \exists Y G(X, Y))$ , where  $F$  and  $G$  are arith-

metric formulas, we can associate the problem

$$P_{F,G} = \{(X, Y) \in 2^{\mathbb{N}} : F(X) \wedge G(X, Y)\}.$$

We then have  $\text{dom}(P_{F,G}) = \{X \in 2^{\mathbb{N}} : F(X)\}$  and for all  $X \in \text{dom}(P_{F,G})$ , the class  $P_{F,G}(X) = \{Y \in 2^{\mathbb{N}} : G(X, Y)\}$ . We call  $\Pi_2^1$  *problem* a problem of this form, and we will identify it to its corresponding formula.

**Example 3.** König’s lemma is a  $\Pi_2^1$  problem. Let’s see another example. Theorem ADS — acronym of “Ascending-Descending Sequence” — affirms for any infinite linear order  $R \subseteq \mathbb{N} \times \mathbb{N}$ , the existence of an infinite strictly increasing sequence, or that of an infinite strictly decreasing sequence. This theorem is therefore the  $\Pi_2^1$  problem  $\text{ADS} \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  whose instances are infinite linear orders over  $\mathbb{N}$ , and for each instance  $R \in \text{dom}(\text{ADS})$ , the solutions  $\text{ADS}(R)$  are strictly increasing or strictly decreasing infinite sequences of elements.

We will study different notions of reductions between problems. Informally, a reduction  $P \leq Q$  between two problems  $P$  and  $Q$  means that the problem  $Q$  is at least as difficult to solve as  $P$ , in the sense that if one had a blackbox allowing to compute solutions to instances of  $Q$ , it would be possible to compute solutions to instances of  $P$ . Let us immediately see our first reduction, which follows directly from the usual separation techniques of theorems over  $\text{RCA}_0$ .

## 1. $\omega$ -reduction

The implication over  $\text{RCA}_0$  induces a purely computational reduction by restricting oneself to  $\omega$ -structures. Consider a  $\Pi_2^1$  problem  $P$ , and its corresponding formula  $\forall X(F(X) \rightarrow \exists Y G(X, Y))$ . Let  $\mathcal{M} = (M, S, +, \times, <)$  be an  $\mathcal{L}_{Z_2}$ -structure. Then,  $\mathcal{M} \models P$  iff for all  $X \in S$  such that  $\mathcal{M} \models F(X)$ , there exists  $Y \in S$  such that  $\mathcal{M} \models G(X, Y)$ . Recall that an  $\omega$ -structure  $\mathcal{M} = (\omega, \mathcal{I}, +, \times, <)$  is a model of  $\text{RCA}_0$  iff  $\mathcal{I}$  is a Turing ideal. This leads us to the following definition.

**Definition 1.1.** A Turing ideal  $\mathcal{I}$  *satisfies* a problem  $P$  (denoted  $\mathcal{I} \models P$ ) if for any instance  $X$  of  $P$  in  $\mathcal{I}$ , there exists a solution to  $X$  in  $\mathcal{I}$ . ◇

Note that for a  $\Pi_2^1$  problem  $P$ , a Turing ideal  $\mathcal{I}$  satisfies  $P$  iff the  $\omega$ -structure  $\mathcal{M} = (\omega, \mathcal{I}, +, \times, <)$  satisfies its corresponding formula. The restriction to  $\omega$ -structures makes it possible to define a notion of satisfiability on arbitrary problems, beyond the  $\Pi_2^1$  problems.

**Definition 1.2.** A problem  $P$  is  $\omega$ -reducible to  $Q$  (denoted  $P \leq_\omega Q$ ) if any Turing ideal which satisfies  $Q$  also satisfies  $P$ .  $\diamond$

In particular, if  $P$  and  $Q$  are  $\Pi_2^1$  problems, then  $P \leq_\omega Q$  iff every  $\omega$ -model of  $\text{RCA}_0 + Q$  is an  $\omega$ -model of  $P$ .

**Example 1.3.** Consider the problems  $\text{KL}$  and  $\text{J}$ , where  $\text{KL}$  is the problem associated with König's lemma (distinct from  $\text{WKL}$ , standing for weak König's lemma), and  $\text{J}$  is the Turing jump problem, whose instances are the elements of  $2^\mathbb{N}$ , and whose unique solution to the instance  $X$  is  $X'$ . The Turing ideals satisfying  $\text{J}$  are by definition the jump ideals (see Section 22-6.1), and characterize the  $\omega$ -models of  $\text{ACA}_0$ .

Let us show that  $\text{KL} \leq_\omega \text{J}$ . Let  $\mathcal{I} \models \text{J}$  and let  $T \in \mathcal{I}$  be an instance of  $\text{KL}$ . Then,  $T$  is a computably bounded tree relative to  $T'$ . By Corollary 8-7.9 relativized to  $T'$ , any PA degree relative to  $T'$  computes a path of  $T$ . In particular, the Turing double jump of  $T$  computes a path of  $T$ . As  $\mathcal{I} \models \text{J}$  and  $T \in \mathcal{I}$ , then  $T' \in \mathcal{I}$ , and therefore  $T'' \in \mathcal{I}$ . By closure of  $\mathcal{I}$  under the Turing reduction,  $\mathcal{I}$  contains a path of  $T$ . Thus  $\mathcal{I} \models \text{KL}$ .

Let us show that  $\text{J} \leq_\omega \text{KL}$ . Let  $\mathcal{I} \models \text{KL}$ , and let  $X \in \mathcal{I}$  be an instance of  $\text{J}$ . By Proposition 8-7.4, there exists a finitely-branching  $X$ -computable  $T$  tree whose unique path is Turing equivalent to  $X'$ . Since  $\mathcal{I}$  is a Turing ideal,  $T \in \mathcal{I}$ . As  $\mathcal{I} \models \text{KL}$ , the unique path of  $T$  belongs to  $\mathcal{I}$ , and as  $\mathcal{I}$  is a Turing ideal,  $X' \in \mathcal{I}$ . So  $\mathcal{I} \models \text{J}$ .

Note that in the previous example, we used two instances of  $\text{J}$  for “solve” an instance of  $\text{KL}$ , while in the opposite direction, a single instance of  $\text{KL}$  was sufficient. We will come back to this point in the next section.

**Exercise 1.4.** Recall that  $\text{WKL}$  and  $\text{KL}$  denote König's lemma for infinite binary trees and infinite finitely-branching trees, respectively. Show that  $\text{WKL} \leq_\omega \text{KL}$  but  $\text{KL} \not\leq_\omega \text{WKL}$ .  $\diamond$

### 1.1. Separation proofs

Let  $P$  and  $Q$  be  $\Pi_2^1$  problems. As explained previously, if  $\text{RCA}_0 + Q \vdash P$ , then in particular  $P \leq_\omega Q$ <sup>1</sup>. By contraposition, the reduction  $\leq_\omega$  makes it possible to separate theorems: if we manage to show  $P \not\leq_\omega Q$ , then  $\text{RCA}_0 + Q \not\vdash P$ . The vast majority of separation proofs in Reverse Mathematics are indeed separation proofs for the  $\omega$ -reduction.

<sup>1</sup>Note that we do not necessarily have  $P \leq_\omega Q$  implies  $\text{RCA}_0 \vdash Q \rightarrow P$ .

**Example 1.5.** The separation of  $\text{WKL}_0$  and  $\text{ACA}_0$  from Proposition 22-7.3 has been done by showing  $J \not\leq_\omega \text{WKL}$ .

In order to prove a separation  $P \not\leq_\omega Q$ , it is necessary to create a Turing ideal  $\mathcal{I}$  such that  $\mathcal{I} \models Q$  but  $\mathcal{I} \not\models P$ . For this, we generally fix an instance  $X_P$  of  $P$  whose solutions are “computably complicated”, and in particular are not  $X_P$ -computable. The class  $\mathcal{I}_0 = \{Z \in 2^\mathbb{N} : Z \leq_T X_P\}$  is a Turing ideal containing  $X_P$ , but none of its solutions. Thus,  $\mathcal{I}_0 \not\models P$ . The next step is to extend  $\mathcal{I}_0$  into a Turing ideal  $\mathcal{I} \models Q$ , while ensuring that  $\mathcal{I}$  does not have any solution to  $X_P$ .

The approach consists in fixing an instance  $X_Q$  of  $Q$  in  $\mathcal{I}_0$ , in choosing a “computably weak” solution  $Y_0$  — enough so that  $X_P \oplus Y_0$  does not compute no solution to  $X_P$  — and define  $\mathcal{I}_1 = \{Z \in 2^\mathbb{N} : Z \leq_T X_P \oplus Y_0\}$ . The Turing ideal  $\mathcal{I}_1$  took a step forward to satisfy  $Q$ , adding a solution to  $X_Q$ , without adding one of  $X_P$ . However, this solution also added new instances of  $Q$  in  $\mathcal{I}_1$ , which will also need to be satisfied. We must then intelligently iterate the construction, to define an increasing sequence of Turing ideals  $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \dots$  so as to take into account all the instances of  $Q$  which appear in  $\mathcal{I}_n$  for a certain  $n$ , while making sure to avoid adding solutions of  $X_P$ . We obtain a Turing ideal  $\mathcal{I} = \bigcup_n \mathcal{I}_n$  satisfying  $Q$ , without satisfying  $P$ .

The notion of “computably weak” solution will correspond to a weakness property, as defined in Section 4-8 on the finite extension method:

**Definition 1.6.** A *weakness property* is a class  $\mathcal{W} \subseteq 2^\mathbb{N}$  downward-closed for the Turing reduction: if  $X \in \mathcal{W}$  and  $Y \leq_T X$ , then  $Y \in \mathcal{W}$ .  $\diamond$

In particular, any Turing ideal is a weakness property. Many computability-theoretic notions are weakness properties without necessarily forming Turing ideals.

**Example 1.7.** The following classes are weakness properties:

1. Sets of low degree.
2. Computably dominated sets.
3. For any non-computable  $A$ , the class  $\mathcal{W} = \{Z \in 2^\mathbb{N} : A \not\leq_T Z\}$ .
4. Arithmetic sets.

Let  $\mathcal{W}$  be a non-empty weakness property. Suppose that  $P$  has an instance  $X_P \in \mathcal{W}$  such that  $\mathcal{W}$  contains no solution to  $X_P$ . We will then try to build our ideal  $\mathcal{I} \subseteq \mathcal{W}$  with  $X_P \in \mathcal{I}$ , and such that  $\mathcal{I} \models Q$ . During

the construction of our ideal, we will find ourselves with instances of  $Q$  computable relative to solutions of instances previously considered. This leads us to the following definition:

**Definition 1.8.** Let  $\mathcal{W}$  be a weakness property. A problem  $Q$  *preserves*  $\mathcal{W}$  if for all  $Z \in \mathcal{W}$ , every instance  $X \leq_T Z$  of  $Q$  admits a solution  $Y$  such that  $Z \oplus Y \in \mathcal{W}$ .  $\diamond$

We have already seen a number of computability preservation proofs.

**Example 1.9.** WKL preserves the class of low sets: For any  $Z$  of low degree, for any infinite  $Z$ -computable tree  $T \subseteq 2^{<\mathbb{N}}$ , by Theorem 8-4.3 relativized to  $Z$ , it exists  $P \in [T]$  such that  $Z \oplus P$  is low relative to  $Z$ . Now if  $A$  and  $B$  is low relative to  $A$ , then  $B$  is low. Indeed,  $B' \leq_T A' \leq_T \emptyset'$ . It follows that  $Z \oplus P$  is low. Likewise, by Theorem 8-4.5, WKL preserves the class of computably dominated sets.

The notion of weakness property preservation is a very powerful tool to prove separations for the  $\omega$ -reduction.

**Lemma 1.10.** Suppose that  $Q$  preserves a weakness property  $\mathcal{W}$ . Then, for all  $X \in \mathcal{W}$ , there exists a Turing ideal  $\mathcal{I} \subseteq \mathcal{W}$  containing  $X$  such that  $\mathcal{I} \models Q$ .  $\star$

PROOF. Let us define an increasing sequence of sets  $Z_0 \leq_T Z_1 \leq_T \dots$  such that  $Z_0 = X$ , such that for all  $n \in \mathbb{N}$  we have  $Z_n \in \mathcal{W}$ , and such that for all  $n, e \in \mathbb{N}$ , if  $\Phi_e^{Z_n}$  is a  $Q$ -instance, then  $Z_{\langle n, e \rangle}$  computes a solution to  $\Phi_e^{Z_n}$ .

Suppose  $Z_0, \dots, Z_n$  is defined. Let  $n+1 = \langle p, e \rangle$ , with  $p, e \leq n$ . If  $\Phi_e^{Z_p}$  is not an instance of  $Q$ , then  $Z_{n+1} = Z_n$ . Otherwise, as  $Q$  preserves  $\mathcal{W}$  and  $Z_n \in \mathcal{W}$ , there exists a solution  $\tilde{Y}$  to  $\Phi_e^{Z_p}$  — because in particular  $\tilde{Y}$  is also computable in  $Z_n$  — such that  $Z_n \oplus \tilde{Y} \in \mathcal{W}$ . Let  $Z_{n+1} = Z_n \oplus \tilde{Y}$ .

By Exercise 22-5.11,  $\mathcal{I} = \{Y \in 2^{\mathbb{N}} : \exists n Y \leq_T Z_n\}$  is a Turing ideal. As  $Z_0 = X$  then  $X \in \mathcal{I}$ . Finally, let us show that  $\mathcal{I} \models Q$ . Let  $\tilde{X}$  be an instance of  $Q$  in  $\mathcal{I}$ . Let  $n \in \mathbb{N}$  be such that  $\tilde{X} \leq_T Z_n$ , and let  $e \in \mathbb{N}$  be such that  $\Phi_e^{Z_n} = \tilde{X}$ . Then,  $Z_{\langle n, e \rangle}$  computes a solution to  $\tilde{X}$ ,  $Z_{\langle n, e \rangle} \in \mathcal{I}$ , so  $\mathcal{I}$  contains a solution to  $\tilde{X}$ .  $\blacksquare$

The following theorem generalizes many constructions of Reverse Mathematics

**Theorem 1.11**

Let  $\mathcal{W}$  be a weakness property. Suppose that  $Q$  preserves  $\mathcal{W}$ , unlike  $P$ .

Then,  $P \not\leq_\omega Q$ .

PROOF. As  $P$  does not preserve  $\mathcal{W}$ , there is an element  $Z \in \mathcal{W}$  and a  $P$ -instance  $X \leq_T Z$  such that for any  $P$ -solution  $Y$ ,  $Z \oplus Y \notin \mathcal{W}$ . As  $Q$  preserves  $\mathcal{W}$ , by Lemma 1.10, there exists a Turing ideal  $\mathcal{I} \subseteq \mathcal{W}$  such that  $Z \in \mathcal{I}$  and  $\mathcal{I} \models Q$ . Let us show that  $\mathcal{I} \not\models P$ . Indeed, by closure of  $\mathcal{I}$  under the Turing reduction,  $X \in \mathcal{I}$ . Let  $Y$  be a  $P$ -solution to  $X$ . Then,  $Z \oplus Y \notin \mathcal{W}$ , so  $Z \oplus Y \notin \mathcal{I}$ , but  $Z \in \mathcal{I}$ , so by closure of  $\mathcal{I}$  under join,  $Y \notin \mathcal{I}$ . Thus  $\mathcal{I}$  is a Turing ideal satisfying  $Q$  but not  $P$ , so  $P \not\leq_\omega Q$ . ■

### Corollary 1.12

Let  $\mathcal{W}$  be a weakness property, and  $P, Q$  be  $\Pi_2^1$  problems. Suppose that  $Q$  preserves  $\mathcal{W}$ , unlike  $P$ . Then,  $\text{RCA}_0 + Q \not\models P$ .

PROOF. By Theorem 1.11,  $P \not\leq_\omega Q$ , so there is an  $\omega$ -model  $\mathcal{M}$  of  $\text{RCA}_0 + P$  which is not a model of  $Q$ , so  $\text{RCA}_0 + Q \not\models P$ . ■

For example,  $\text{WKL}$  preserves the class of low sets, while  $\text{KL}$  does not, so  $\text{KL} \not\leq_\omega \text{WKL}$ , and  $\text{RCA}_0 + \text{WKL} \not\models \text{KL}$ .

## 1.2. Cone avoidance

Among the weakness properties, there are several large recurring families in Reverse Mathematics. The most emblematic is that of *cone avoidance*, which expresses the inability of a problem to “code” a non-computable set in each of its solutions.

**Definition 1.13.** Let  $k \in \mathbb{N}$ . A problem  $P$  *avoids  $k$  cones* (resp.  $\omega$  cones) if for any  $Z \in 2^\mathbb{N}$ , any sequence  $(C_j)_{j < k}$  (resp.  $(C_j)_{j \in \mathbb{N}}$ ) of non  $Z$ -computable sets, and for any  $P$ -instance  $X \leq_T Z$ , there exists a solution  $Y$  to  $X$  such  $C_j$  is not  $Z \oplus X$ -computable for any  $j < k$  (resp. any  $j \in \mathbb{N}$ ). ♦

Formulated in the language of preservations, for any  $k \in \mathbb{N}$  and any sequence of sets  $\overline{C} = (C_j)_{j < k}$ , we associate the weakness property  $\mathcal{W}_{\overline{C}} = \{Z : \forall j < k \ C_j \not\leq_T Z\}$ . Note that  $\mathcal{W}_{\overline{C}} = \emptyset$  if one of  $C_j$  is computable, in which case any problem preserves  $\mathcal{W}_{\overline{C}}$ . By unwinding the definitions,  $P$  avoids  $k$  cones iff  $P$  preserves  $\mathcal{W}_{\overline{C}}$  for any sequence of sets  $\overline{C} = (C_j)_{j < k}$ . In particular, by Theorem 1.11, if  $Q$  avoids  $k$  cones, unlike  $P$ , then  $P \not\leq_\omega Q$ .

**Example 1.14.**  $\text{WKL}$  avoids  $\omega$  cones by Theorem 8-4.7. On the other hand,  $\text{KL}$  does not avoid 1 cone, because there exists a computable

finitely-branching tree of which all the paths compute the halting problem. It follows from Theorem 1.11 that  $\text{KL} \not\leq_c \text{WKL}$ .

The notion of cone avoidance is used in particular to separate theorems from  $\text{ACA}_0$ . We will see a non-trivial example of this with Ramsey's theorem for pairs. Besides its use for separation theorems, the notion was studied for itself, notably by Downey, Greenberg, Harrison-Trainer, Patey and Turetsky [46] who found several equivalent characterizations. Let us mention in particular the following nice theorem.

**Theorem 1.15 ([46])**

*A P problem avoids 1 cone iff it avoids  $\omega$  cones.*

## 2. Computational reduction

The implication over  $\text{RCA}_0$  and the  $\omega$ -reduction are computably coarse, because they do not take into account the number of applications of each problem. To prove  $\text{RCA}_0 \vdash Q \rightarrow P$ , we can use the premise  $Q$  as many times as we want in our proof. Similarly, to prove  $P \leq_\omega Q$ , one is allowed to use an arbitrary number of instances of  $Q$  to solve  $P$ .

As soon as we assume the existence of the Turing jump in  $\text{RCA}_0$ , the system obtained implies  $\text{ACA}_0$ , and implies the existence of all finite iterations of the Turing jump. Dzhafarov [51] introduced a finer notion of reduction, sensitive to the number of applications of each problem.

**Definition 2.1.** A problem  $P$  is *computably reducible* at  $Q$  (denoted  $P \leq_c Q$ ) if for any instance  $X$  of  $P$ , there exists an instance  $\tilde{X}$  of  $Q$  computable in  $X$ , such that for any  $Q$ -solution  $\tilde{Y}$  to  $\tilde{X}$ ,  $X \oplus \tilde{Y}$  computes a  $P$ -solution to  $X$ . ◇

The computational reduction is closer to the concepts of reduction studied in Complexity Theory. Note that the  $Q$ -solution  $\tilde{Y}$  to  $\tilde{X}$  has the right to call on the instance  $X$  of  $P$  to compute a  $P$ -solution to  $X$ . The computational reduction is a refinement of the  $\omega$ -reduction:

**Exercise 2.2.** Show that if  $P \leq_c Q$ , then  $P \leq_\omega Q$ . ◇

The reverse implication is not generally true. Intuitively, the computational reduction is an  $\omega$ -reduction which allows only one application of the right-hand problem. It allows, for example, to give a formal framework to say that a single application of  $J$  is not enough to prove  $\text{KL} \leq_\omega J$  in Example 1.3.

**Exercise 2.3.** Let  $\text{KL}$  and  $\text{J}$  be the problems defined in Example 1.3. Show that  $\text{KL} \not\leq_c \text{J}$ .  $\diamond$

The computational reduction is therefore strictly finer than the  $\omega$ -reduction.

**Remark**

The reduction  $\text{RCA}_0 \vdash Q \rightarrow P$  does not imply  $P \leq_c Q$  because several instances of  $Q$  may be needed in the proof. The reduction  $P \leq_c Q$  also does not imply  $\text{RCA}_0 \vdash Q \rightarrow P$  because the Turing reductions used in  $P \leq_c Q$  are not necessarily provably total using  $\Sigma_1^0$  induction alone. However, many proofs of  $\text{RCA}_0 \vdash Q \rightarrow P$  are actually proofs of  $P \leq_c Q$  which formalize in  $\text{RCA}_0$ .

As we have seen,  $P \leq_\omega Q$  does not generally imply  $P \leq_c Q$ . This is however the case when  $Q = \text{WKL}$ . Here,  $\text{WKL}$  is the problem whose instances are infinite binary trees, and the solutions are the paths of these trees. The reason for this implication stems from Scott's following result.

**Proposition 2.4 (Scott [193]).** Let  $P$  be a PA degree. There is a Scott ideal whose elements are  $P$ -computable.  $\star$

PROOF. Let  $T$ , be a tree functional such that for all  $X \in 2^\mathbb{N}$ ,  $T^X$  is a binary tree, all the paths of which are of PA degree relative to  $X$ . For example,  $T^X = \{\sigma \in 2^{<\mathbb{N}} : \forall e < |\sigma| \ \Phi_e^X(e)[|\sigma|] \neq \sigma(e)\}$ . Let  $\mathcal{C} = \{\bigoplus_i X_i \in 2^\mathbb{N} : \forall i \ X_{i+1} \in [T^{X_i}]\}$ . Note that  $\mathcal{C}$  is a non-empty  $\Pi_1^0$  class. In particular,  $P$  computes a member  $\bigoplus_i X_i$  of  $\mathcal{C}$ . Note that  $X_{i+1}$  is of PA degree relative to  $X_i$ , and in particular  $X_{i+1} \geq_T X_i$  (see Exercise 8-7.6).

Let  $\mathcal{I} = \{Z \in 2^\mathbb{N} : \exists i \ Z \leq_T X_i\}$ . By Exercise 22-5.11,  $\mathcal{I}$  is a Turing ideal. Moreover, for all  $Z \in \mathcal{I}$ , there exists  $i$  such that  $Z \leq_T X_i$ , but  $X_{i+1} \in \mathcal{I}$  is of PA degree relative to  $X_i$ , therefore relative to  $Z$  as well. So  $\mathcal{I}$  is a Scott ideal. Finally, all the elements of  $\mathcal{I}$  are  $P$ -computable.  $\blacksquare$

The following proposition is a direct consequence of Scott's result.

**Proposition 2.5 (Hirschfeldt and Jockusch [87]).** Let  $P$  be a problem. If  $P \leq_\omega \text{WKL}$ , then  $P \leq_c \text{WKL}$ .  $\star$

PROOF. Let  $X$  be an instance of  $P$ . Let  $T$  be an  $X$ -computable tree in which all the paths are of PA degree relative to  $X$ . The tree  $T$  is an  $X$ -computable instance of  $\text{WKL}$ . Let  $\tilde{Y}$  be a  $\text{WKL}$ -solution to  $T$ , in other words  $\tilde{Y} \in [T]$ . Let us show that  $\tilde{Y}$  computes a  $P$ -solution to  $X$ . By Proposition 2.4 relativized to  $X$ , there exists a Scott ideal  $\mathcal{I}$  containing  $X$  and whose elements are  $X \oplus \tilde{Y}$ -computable (therefore  $\tilde{Y}$ -computable because  $\tilde{Y} \geq_T X$ ). Since  $\mathcal{I}$  is a Scott ideal,  $\mathcal{I} \models \text{WKL}$ , but  $P \leq_\omega \text{WKL}$ ,

so  $\mathcal{I} \models P$ . As  $X \in \mathcal{I}$ , there is a  $P$ -solution  $Y$  to  $X$  in  $\mathcal{I}$ , so there is an  $\tilde{Y}$ -computable  $P$ -solution to  $X$ . ■

Note that in the previous proof,  $\tilde{Y}$  does not need the instance  $X$  as an oracle to compute a  $P$ -solution to  $X$ . We will see in Section 5 that this corresponds to a stronger notion than the computational reduction.

Let us end by introducing a last notion which formalizes the fact that certain problems are trivial from the point of view of computational reduction.

**Definition 2.6.** A  $P$  problem is *computably true* if for any  $X$  instance of  $P$ ,  $X$  computes a  $P$ -solution to  $X$ . ◇

We will see in Proposition 25-2.4 that this is for example the case with the pigeonhole principle.

**Exercise 2.7.** Show that  $P$  is computably true if and only if  $P \leq_c \text{Id}$ , where  $\text{Id}$  is the identity problem whose solutions are equal to the instances. ◇

### 3. Weihrauch reduction

Consider the proof in  $\text{RCA}_0$  of the Intermediate Value Theorem (see Proposition 22-5.13). Given a continuous function  $f : [0, 1] \rightarrow \mathbb{R}$  satisfying  $f(0) < 0 < f(1)$ , the proof proceeds by case analysis: either there exists a rational number  $q \in ]0, 1[$  such that  $f(q) = 0$ , in which case, any rational number being computable,  $f$  has a computable solution, or one obtains a real  $r$  such that  $f(r) = 0$  by a computable dichotomous search. Can we get rid of this case analysis? Just as the computational reduction refines the  $\omega$ -reduction by taking into account the number of applications, the Weihrauch reduction makes it possible to give a formal meaning to the previous question by taking into account the uniformity of the proofs.

**Definition 3.1.** A problem  $P$  is *Weihrauch reducible* to  $Q$  (denoted  $P \leq_W Q$ ) if there are two Turing functionals  $\Phi$  and  $\Psi$  such that for any instance  $X$  of  $P$ ,  $\Phi^X$  is an instance of  $Q$ , such that for any  $Q$ -solution  $\tilde{Y}$  to  $\Phi^X$ ,  $\Psi^{X \oplus \tilde{Y}}$  is a  $P$ -solution to  $X$ . We say that the functionals  $\Phi$  and  $\Psi$  *witness*  $P \leq_W Q$ . ◇

It follows directly from the definitions that if  $P \leq_W Q$ , then  $P \leq_c Q$ . The Weihrauch reduction was introduced by Klaus Weihrauch in the 80s as a computable analysis tool, before experiencing a revival with the work

of Gherardi and Marcone [72] in 2009, showing the usefulness of this reduction to analyze the uniform content of proofs in Reverse Mathematics. Since then, the Weihrauch degrees have grown and appear as a formalism concurrent with the implication over  $\text{RCA}_0$  for the analysis of theorems.

### 3.1. Intermediate Value Theorem

In order to formulate more precisely our question on the uniformity of the Intermediate Value Theorem, we introduce an analog for Weihrauch's reduction of the notion of computably true problem.

**Definition 3.2.** A problem  $P$  is *uniformly true* if there exists a Turing functional  $\Phi$  such that for any instance  $X$  of  $P$ ,  $\Phi^X$  is a  $P$ -solution to  $X$ .  $\diamond$

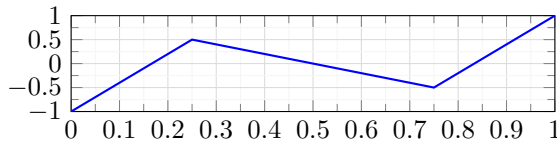
**Exercise 3.3.** Show that  $P$  is uniformly true if and only if  $P \leq_W \text{Id}$ , where  $\text{Id}$  is the identity problem whose solutions are equal to the instances.  $\diamond$

We now have the tools to answer the question posed at the beginning of the section, namely, is there a proof of the Intermediate Value Theorem with no case analysis?

**Proposition 3.4 (Weihrauch [232]).** The Intermediate Value Theorem is not uniformly true.  $\star$

**PROOF.** Let  $\Phi$  be a Turing functional such that, for any continuous  $f : [0, 1] \rightarrow \mathbb{R}$  function satisfying  $f(0) < 0 < f(1)$ ,  $\Phi(f)$  is a root of  $f$ . The concept of computable function is defined as follows for a function  $\Phi : \mathcal{C}([0, 1], \mathbb{R}) \rightarrow \mathbb{R}$ :  $\Phi$  starts its computation on an element of  $2^{\mathbb{N}}$  which represents a continuous function as explained in Section 22-4, and returns an element of  $2^{\mathbb{N}}$  which represents — still as explained in Section 22-4 — a real number. We require that on any representation of  $f$ , our process computes a representation of  $\Phi(f)$ .

Let  $f : [0, 1] \rightarrow \mathbb{R}$  be a strictly increasing continuous function over the interval  $[0, 0.25]$ , strictly decreasing over the interval  $[0.25, 0.75]$ , and strictly increasing over  $[0.75, 1]$ , such that  $f(0) = -1$ ,  $f(0.25) = 0.5$ ,  $f(0.5) = 0$ ,  $f(0.75) = -0.5$  and  $f(1) = 1$  (see Figure 3.5). Let  $g : [0, 1] \rightarrow \mathbb{R}$  be defined by  $g(a) = \Phi(x \mapsto f(x) + a)$ . The reader will be able to verify that  $g$  is a continuous function, in particular because  $\Phi$  is computable in the sense given above. Also for  $a \in ]0.5, 1[$ , the function  $x \mapsto f(x) + a$  has only one root (the one on the left). By continuity,  $g$  will continue to return the leftmost root on  $[0, 0.5]$ . Conversely, for  $a \in ]-1, -0.5[$ , the function  $x \mapsto f(x) + a$  has only the right root. By continuity,  $g$  will return the rightmost root on  $[-0.5, 0]$ . We therefore have two different values for  $g(0)$ , which is a contradiction.  $\blacksquare$

Figure 3.5: Graph of the function  $f$  from Proposition 3.4

The structure of the degrees induced by the Weihrauch reduction turns out to be extremely rich. The reader who would like to know more will find a detailed introduction in *Weihrauch Complexity in Computable Analysis* by Brattka, Gherardi and Pauly [24].

### 3.2. Choice problems and parallelization

The Weihrauch reduction provides a panel of very weak problems which make it possible to measure computational powers which cannot be analyzed in terms of Turing degrees. Among these problems are the principles of omniscience:

- Definition 3.6.** (1) The *Limited Principle of Omniscience* (LPO) is the problem whose instances are sequences  $X \in 2^{\mathbb{N}}$ , and which decides if there is  $n \in \mathbb{N}$  such that  $X(n) = 0$ .
- (2) The *Lesser Limited Principle of Omniscience* (LLPO) is the problem whose instances are sequences  $X \in 2^{\mathbb{N}}$  such that  $X(n) = 1$  for at most one  $n$ , and whose solutions any  $i < 2$  such that  $\forall n \ X(2n + i) = 0$ .  $\diamond$

Note that the co-domain of these problems is not in  $2^{\mathbb{N}}$  but in  $\mathbb{N}$ . It is however possible to represent any integer  $n$  as the following  $1^n 0^\infty$ . Each instance of the principle LPO has a unique solution, while some instances of LLPO have two solutions.

**Exercise 3.7.** ( $\star$ ) Show that  $\text{LLPO} \leq_W \text{LPO}$  but that  $\text{LPO} \not\leq_W \text{LLPO}$ .  $\diamond$

**Exercise 3.8.** ( $\star$ ) Show that LLPO is not uniformly true.  $\diamond$

The problems LPO and LLPO can be seen as atomic versions of well-known computational phenomena through the operation of parallelization.

**Definition 3.9.** The *parallelization* of a problem  $P$  is the problem  $\hat{P}$  whose instances are of the form  $\bigoplus_n X_n$  where  $X_n \in \text{dom } P$  for all  $n$ . A  $\hat{P}$ -solution to  $\bigoplus_n X_n$  is a set of the form  $\bigoplus_n Y_n$  where  $Y_n$  is a  $P$ -

| solution to  $X_n$  for all  $n$ . ◇

It is easy to show that the parallelization operation is stable under Weihrauch reduction, and therefore corresponds to an operation on Weihrauch degrees.

**Exercise 3.10.** Show that if  $P \leq_W Q$ , then  $\widehat{P} \leq_W \widehat{Q}$ . ◇

Recall that the Turing jump problem is the problem  $J$  whose instances are the sets  $X \in 2^{\mathbb{N}}$ , and whose unique solution to  $X$  is its Turing jump  $X'$ . The following proposition shows that the problem  $LPO$  can be seen as a bit of information from the Turing jump.

**Proposition 3.11.**  $J \equiv_W \widehat{LPO}$ . ★

PROOF. Let us show that  $J \leq_W \widehat{LPO}$ . For any  $e$ , let  $\Gamma_e$  be the Turing functional defined by  $\Gamma_e^X(s) = 0$  iff  $\Phi_e^X(e)[s] \downarrow$ . Note that for all  $X$ ,  $\Gamma_e^X$  seen as an instance of  $LPO$  has for solution 1 iff  $\Phi_e^X(e) \downarrow$ , in other words iff  $e \in X'$ . Let  $\Phi$  be the Turing functional defined by  $\Phi^X = \bigoplus_e \Gamma_e^X$ . Let  $\Psi$  be the functional defined by  $\Psi(X \oplus Y) = Y$ . The functionals  $\Phi$  and  $\Psi$  witness that  $J \leq_W \widehat{LPO}$ .

Let us show that  $\widehat{LPO} \leq_W J$ . Let  $\Psi$  be defined as follows:  $\Psi^{X \oplus Y}(n) = Y(e_n)$ , where  $e_n$  is the Turing index of the machine such that  $\Phi_{e_n}^X(e_n) \downarrow$  iff there is  $s \in \mathbb{N}$  such that  $\langle n, s \rangle \notin X$ . Let us show that if  $\bigoplus_r X_r$  is an instance of  $\widehat{LPO}$ ,  $\Psi((\bigoplus_r X_r) \oplus (\bigoplus_r X_r)')$  is a  $\widehat{LPO}$ -solution to  $\bigoplus_r X_r$ . Let  $n \in \mathbb{N}$ .  $\Psi((\bigoplus_r X_r) \oplus (\bigoplus_r X_r)', n) = (\bigoplus_r X_r)'(e_n)$ , where  $\Phi_{e_n}^{\bigoplus_r X_r}(e_n) \downarrow$  iff there is  $s \in \mathbb{N}$  such that  $s \notin X_n$ . Then,  $(\bigoplus_r X_r)'(e_n) = 1$  iff there is an  $s \in \mathbb{N}$  such that  $s \notin X_n$ , so for any  $n$ ,  $\Psi((\bigoplus_r X_r) \oplus (\bigoplus_r X_r)', n)$  is a  $LPO$ -solution to  $X_n$ . So  $\Psi((\bigoplus_r X_r) \oplus (\bigoplus_r X_r)')$  is a  $\widehat{LPO}$ -solution to  $\bigoplus_r X_r$ . The identity functional as well as  $\Psi$  therefore witness that  $\widehat{LPO} \leq_W J$ . ■

Likewise, the problem  $LLPO$  can be seen as a step of weak König's lemma.

**Exercise 3.12.** (★) Show that  $WKL \equiv_W \widehat{LLPO}$ . ◇

### Remark

The parallelization operator makes it possible to establish bridges between the Weihrauch reduction and the computational reduction. By Exercise 3.10, if  $P \leq_W Q$ , then  $\widehat{P} \leq_W \widehat{Q}$ , therefore in particular  $\widehat{P} \leq_c \widehat{Q}$ . The contrapositive allows us to prove a number of separation results in the Weihrauch degrees using Computability Theory. For example, by the low basis theorem for the  $\Pi_1^0$  classes,  $J \not\leq_c WKL$ . It follows from Proposition 3.11 and Exercise 3.12 that  $LPO \not\leq_W LLPO$ . Likewise,

if  $P$  is uniformly true, then  $\hat{P}$  is computably true. We therefore deduce that,  $WKL$  not being computably true,  $LLPO$  is not uniformly true.

## 4. Reduction games

It is often more intuitive to think of computational reductions as two-player games: the first player tries to convince us that the reduction is true, while the second player tests the reduction by trying to make the worst possible choice of solutions. Hirschfeldt and Jockusch [87] gave a characterization of the  $\omega$ -reduction in terms of games.

**Definition 4.1.** Let  $P$  and  $Q$  be problems. The *reduction game*

$$G(Q \rightarrow P)$$

is a two-player game that proceeds as follows:

- (1) If during the game one player gets stuck, the other player wins.
- (2) If a player wins, the game ends.
- (3) In the first turn, Player 1 plays an instance  $X_0$  of  $P$ . Then Player 2 plays either an  $X_0$ -computable solution to  $X_0$  and wins, or plays an  $X_0$ -computable  $Y_1$  instance of  $Q$ .
- (4) On the  $n$ th turn ( $n > 2$ ), Player 1 plays an  $X_{n-1}$  solution to the  $Q$ -instance  $Y_{n-1}$ . Then Player 2 plays either a  $(X_0 \oplus \cdots \oplus X_{n-1})$ -computable  $P$ -solution to  $X_0$  and wins, or plays a  $(X_0 \oplus \cdots \oplus X_{n-1})$ -computable  $Y_n$  instance of  $Q$ .
- (5) If the game does not end, Player 1 wins. ◇

Note that Player 1 can never get stuck, because by definition, any instance admits a solution. On the other hand, Player 2 can get stuck because  $Q$  may have no  $(X_0 \oplus \cdots \oplus X_{n-1})$ -computable instance. Most of the problems considered in Reverse Mathematics have computable instances, and the game never gets stuck.

**Proposition 4.2 (Hirschfeldt and Jockusch [87]).**  $P \leq_\omega Q$  iff Player 2 has a winning strategy for  $G(Q \rightarrow P)$ . ★

PROOF. Suppose  $P \leq_\omega Q$ . Let's define a winning strategy for Player 2. In turn  $n$ , when Player 2 can play,  $X_0, \dots, X_{n-1}$  and  $Y_1, \dots, Y_{n-1}$  have already been defined. If Player 2 can win by playing a  $P$ -solution  $(X_0 \oplus \cdots \oplus X_{n-1})$ -computable of  $X_0$ , then he plays it and wins. Otherwise, let  $n = \langle m, e \rangle$ .

If  $\Phi_e(X_0 \oplus \cdots \oplus X_{n-1})$  is an instance of  $Q$ , then Player 2 plays it. Otherwise, Player 2 plays an  $X_0$ -computable instance of  $Q$ . Note that  $Q$  necessarily admits such an instance, otherwise  $\mathcal{I} = \{Z \in 2^{\mathbb{N}} : Z \leq_T X_0\}$  would be a Turing ideal satisfying  $Q$ , therefore satisfying  $P$ , therefore  $X_0$  would have an  $X_0$ -computable solution and the Player 2 would have won in round 1.

Suppose that Player 2 never finds himself in the case where he finds a solution to  $X_0$ . The game does not end, so Player 1 wins. Let  $\mathcal{I} = \{Z : \exists n Z \leq_T X_0 \oplus \cdots \oplus X_n\}$ . By Exercise 22-5.11,  $\mathcal{I}$  is a Turing ideal. Also,  $X_0 \in \mathcal{I}$ , but since Player 2 failed to compute a solution to  $X_0$  at any time,  $\mathcal{I}$  does not contain a solution to  $X_0$ , so  $\mathcal{I} \not\models P$ . Finally, for any  $Q$ -instance  $Z \in \mathcal{I}$ , let  $n$  be such that  $Z \leq_T X_0 \oplus \cdots \oplus X_n$  and be  $e$  such that  $\Phi_e(X_0 \oplus \cdots \oplus X_n) = Z$ . Then,  $Y_{\langle n, e \rangle} = Z$ , and  $X_{\langle n, e \rangle}$  is a  $Q$ -solution to  $Z$  in  $\mathcal{I}$ , so  $\mathcal{I} \models Q$ . It follows that  $P \not\leq_\omega Q$ . Contradiction. Player 2 therefore has a winning strategy.

Now suppose that  $P \not\leq_\omega Q$ . Let's define a winning strategy for Player 1. Let  $\mathcal{I}$  be a Turing ideal such that  $\mathcal{I} \models Q$  but  $\mathcal{I} \not\models P$ . Let  $X_0 \in \mathcal{I}$  be an instance of  $P$  with no solution in  $\mathcal{I}$ . Player 1's strategy is to keep the game in  $\mathcal{I}$  to prevent Player 2 from computing a  $P$ -solution to  $X_0$ . On turn 1, Player 1 plays  $X_0$ . By turn  $n > 1$ ,  $X_0, \dots, X_{n-2}$  and  $Y_0, \dots, Y_{n-1}$  have already been played, all in  $\mathcal{I}$ . As  $\mathcal{I} \models Q$ ,  $Y_{n-1}$  is a solution  $X_{n-1}$  to  $\mathcal{I}$ . This solution is played by Player 1. At any time during the construction, Player 2 only has oracles in  $\mathcal{I}$ , and can therefore only play instances of  $Q$  in  $\mathcal{I}$ . The game never stops, unless Player 2 gets stuck with no  $(X_0 \oplus \cdots \oplus X_{n-1})$ -computable  $Q$ -instance. In both cases, Player 1 wins. ■

The game formalism also enables to count the number of instances necessary for a reduction.

**Definition 4.3 (Hirschfeldt and Jockusch [87]).** We note  $P \leq_\omega^n Q$  if Player 2 has a winning strategy for the game  $G(Q \rightarrow P)$ , which guarantees him a victory in at most  $n + 1$  turns. ◇

In particular,  $P \leq_\omega^0 Q$  iff  $P$  is computably true. If  $P \leq_c Q$ , then  $P \leq_\omega^1 Q$ . Moreover, if  $Q$  admits computable instances, then the converse is true.

**Exercise 4.4. (★)** Recall that  $KL$  is the problem of König's lemma for finitely-branching trees, and that  $J$  is the problem of the Turing jump. Show that  $KL \leq_\omega^2 J$  but  $KL \not\leq_\omega^1 J$ . ◇

There are variants of the game reduction to formalize generalizations of the Weihrauch reduction (see Hirschfeldt and Jockusch [87]) as well as model-theoretic oriented variants which formalize implication over  $RCA_0$ . The reader interested in this approach will find an in-depth analysis in the works of Dzhafarov, Hirschfeldt and Reitzes [52].

## 5. Strong reductions

So far, we have seen three big reductions that coexist in Computability Theory. The  $\omega$ -reduction is the computational counterpart of the implication over  $\text{RCA}_0$ . The computational reduction refines this reduction by allowing only one application of the problem, while the Weihrauch reduction studies the uniform dimension of the reduction. For each of the reductions  $P \leq Q$ , the solution  $\tilde{Y}$  to the instance  $\tilde{X}$  of  $Q$  is allowed to use the instance  $X$  of  $P$  as an oracle, to compute a solution to  $X$ . However, it often happens that  $\tilde{Y}$  directly computes a solution to  $X$ .

**Definition 5.1.** A problem  $P$  is *strongly computably reducible* to  $Q$  (denoted  $P \leq_{sc} Q$ ) if for any instance  $X$  of  $P$ , there exists an instance  $\tilde{X}$  of  $Q$  computable in  $X$ , such that any  $Q$ -solution  $\tilde{Y}$  to  $\tilde{X}$  computes a  $P$ -solution to  $X$ .  $\diamond$

In the same way, we define the *strong Weihrauch reduction*, which we denote by  $\leq_{sW}$ . Strong reductions reflect a closer link between the solutions of  $P$  and  $Q$ , in the sense that we are able to encode all the information of a  $P$ -solution in a  $Q$ -solution.

**Example 5.2.** Most of the reduction proofs we have seen are strong. For example,  $\text{WKL} \leq_{sW} J \leq_{sW} \text{KL}$ . On the other hand, the reduction  $\text{LLPO} \leq_W \text{LPO}$  seen in Exercise 3.7 is not strong.

Some problems, like weak König's lemma, have the ability to encode sets in their solutions. These are called cylinders.

**Definition 5.3.** The *product* of two problems  $P$  and  $Q$ , is the problem  $P \times Q$  whose domain is  $\{X_0 \oplus X_1 : X_0 \in \text{dom } P \wedge X_1 \in \text{dom } Q\}$ , and such that for any instance  $X_0 \oplus X_1$ , the solutions are of the form  $Y_0 \oplus Y_1$  where  $Y_0$  is a  $P$ -solution to  $X_0$  and  $Y_1$  a  $Q$ -solution to  $X_1$ . A problem  $P$  is a *cylinder* for  $\leq_{sc}$  (resp.  $\leq_{sW}$ ) if  $\text{id} \times P \leq_{sc} P$  (resp.  $\text{id} \times P \leq_{sW} P$ ).  $\diamond$

Trivially, if  $P$  is a cylinder for  $\leq_{sW}$ , then it is a cylinder for  $\leq_{sc}$ . The notion of cylinder captures exactly the problems which are insensitive to the strong reduction, in the following sense.

**Proposition 5.4.** A problem  $Q$  is a cylinder for  $\leq_{sc}$  iff for any problem  $P$ ,  $P \leq_c Q \leftrightarrow P \leq_{sc} Q$ .  $\star$

**PROOF.** Note that for any problem  $Q$ , if  $P \leq_{sc} Q$ , then  $P \leq_c Q$ , because it suffices to ignore the instance  $X$  of  $P$ .

Suppose that  $Q$  is a cylinder for  $\leq_{sc}$  and that  $P \leq_c Q$ . Let  $X$  be an instance of  $P$ . Let  $\tilde{X}$  be the  $X$ -computable instance of  $Q$  such that for any  $Q$ -solution  $\tilde{Y}$ ,  $X \oplus \tilde{Y}$  compute a  $P$ -solution to  $X$ . Since  $Q$  is a cylinder,  $\text{id} \times$

$Q \leq_{sc} Q$ . Let  $X_1$  be the  $X \oplus \tilde{X}$ -computable instance of  $Q$  such that for any  $Q$ -solution  $Y_1$  to  $X_1$ ,  $Y_1$  computes a  $\text{Id} \times Q$ -solution to  $X \oplus \tilde{X}$ , in other words  $Y_1$  computes  $X \oplus \tilde{Y}$ , where  $\tilde{Y}$  is a  $Q$ -solution to  $\tilde{X}$ . As  $X \geq_T \tilde{X}$ ,  $X_1$  is  $X$ -computable, and since  $X \oplus \tilde{Y}$  computes a  $P$ -solution to  $X$ , then  $Y_1$  computes a  $P$ -solution to  $X$ . So  $P \leq_{sc} Q$ .

Now suppose that for any problem  $P$ , if  $P \leq_c Q$  then  $P \leq_{sc} Q$ . In particular, for  $P$  the problem  $\text{Id} \times Q$ , we have  $\text{Id} \times Q \leq_c Q$ , therefore  $\text{Id} \times Q \leq_{sc} Q$ . In other words,  $Q$  is a cylinder for  $\leq_{sc}$ . ■

Among the cylinders, we will count weak König's lemma, but also the problem of the jump  $J$ .

**Proposition 5.5.**  $\text{WKL}$  is a cylinder for  $\leq_{sW}$ . ★

PROOF. Let  $\Phi$  be the functional such that for any set  $X$  and any binary tree  $T$ ,  $\Phi(X \oplus T)$  is a binary tree satisfying

$$[\Phi(X \oplus T)] = \{X \oplus P : P \in [T]\}$$

More precisely, we can define

$$\Phi(X \oplus T) = \left\{ \sigma \in 2^{<\mathbb{N}} : \exists \tau \in T^{[|\sigma|]} \forall i < |\sigma| \begin{array}{ll} i \text{ even} & \rightarrow \sigma(i) = \tau(i/2) \\ i \text{ odd} & \rightarrow \sigma(i) = X((i-1)/2) \end{array} \right\}$$

where  $T^{[n]}$  is the set of nodes of  $T$  of size  $n$ . Then,  $\Phi$  and the identity functional witness the reduction  $\text{Id} \times \text{WKL} \leq_{sW} \text{WKL}$ . ■

Recall that, by Proposition 2.5, if  $P \leq_\omega \text{WKL}$ , then  $P \leq_c \text{WKL}$ . Thanks to Proposition 5.5 and Proposition 5.4, we can deduce that  $P \leq_{sc} \text{WKL}$ .

**Exercise 5.6.** Show the equivalent of Proposition 5.4 for the Weihrauch reduction. ◇

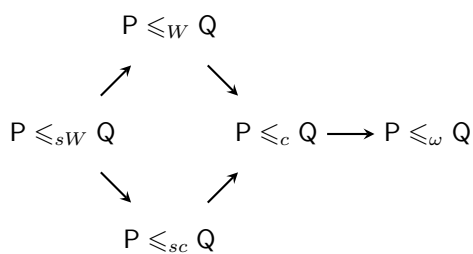


Figure 5.7: Summary diagram of computable reductions. Each implication is strict.

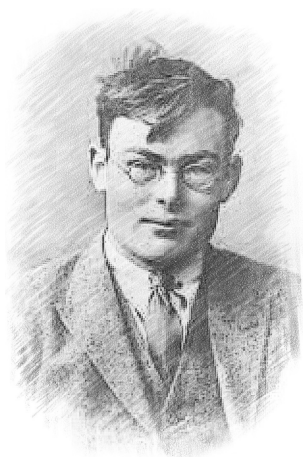


# Chapter 25

## Ramsey's theorem

In the opinion of his teachers and contemporaries, Frank Ramsey was an extremely brilliant mind. Unfortunately, he died too young, at the age of 26, from illness. Despite his young age, he had already accomplished a lot and his name will pass to posterity in particular through two works. The first in economics: in 1928 he published an article on the savings rate, in which he developed what would be called the Ramsey model. One of its professors, the world famous economist John Maynard Keynes, will call this work “one of the most remarkable contributions ever made to mathematical economic”. The second is in mathematical logic. Ramsey is interested in philosophy, logic,

probability, and the foundations of mathematics. In 1930, he published an article on this theme, in which he studied the consistency of certain logical formulas. It is not however the initial object of the article that we will retain, but a simple lemma proved at the turn of a page, of which Erdős and Szekeres were to get five years later the measure. This lemma will be the starting point of what we will call Ramsey's theory, an important field of research in combinatorial mathematics.



Frank Ramsey, 1903–1930

## 1. Overview

The great historical observation of Reverse Mathematics is its structural phenomenon: most ordinary theorems are either provable in  $\text{RCA}_0$ , or modulo  $\text{RCA}_0$  equivalent to one of the four big systems of axioms  $\text{WKL}_0$ ,  $\text{ACA}_0$ ,  $\text{ATR}_0$ , and  $\Pi^1_1\text{-CA}_0$ , linearly strictly ordered by the implication over  $\text{RCA}_0$ . This is above all an empirical phenomenon, and relates to “ordinary” theorems ; but it is easy to conceive artificial statements which escape this structure. Does the latter then simply stem from a human bias in the practice of mathematics, or does it nevertheless translate something deeper into it? This chapter may help the reader to form an opinion on the matter.

Ramsey's theorem for pairs of integers has received special attention, as it is historically one of the first natural theorems to escape this classification. The community has therefore looked in great detail on its metamathematical aspects, thus providing perhaps some answers to the question posed in the paragraph above. This is a fascinating mathematical adventure that led to the discovery of many techniques in Computability Theory and Proof Theory. Let us start by stating Ramsey's theorem in the general case.

### Notation

Let  $X \subseteq \mathbb{N}$ . We write  $[X]^n$  for the set of subsets of  $X$  of size  $n$ .

Unordered sets of size  $n$  are in bijection with the  $n$ -tuples  $(a_1, \dots, a_n)$  where  $a_i < a_{i+1}$ . We will therefore sometimes identify  $[X]^n$  with the set of strictly increasing  $n$ -tuples and we will use the word abuse of  $n$ -tuples to designate the elements of  $[X]^n$ . A *coloring* is a function  $f : [\mathbb{N}]^n \rightarrow \mathbb{N}$ . If the image of  $f$  is bounded, of size at most  $k$ , it is a  $k$ -*coloring* or *finite coloring*. Given a coloring  $f$  on  $[\mathbb{N}]^n$ , a set  $X \subseteq \mathbb{N}$  is *homogeneous* for  $f$  if the color  $f(\{x_1, \dots, x_n\})$  is the same for all  $\{x_1, \dots, x_n\} \in [X]^n$ . We will abbreviate  $f(\{x_1, \dots, x_n\})$  by  $f(x_1, \dots, x_n)$  — where we will then assume  $x_1 < \dots < x_n$  — according to our identification of  $[X]^n$  with the set of strictly increasing  $n$ -tuples. Finally, we will identify the integer  $k$  with the set  $\{0, 1, \dots, k-1\}$ .

### Theorem 1.1 (Generalized Ramsey's theorem)

Let  $n, k \in \mathbb{N}^*$ . Any coloring  $f : [\mathbb{N}]^n \rightarrow k$  admits an infinite homogeneous set.

The theorem originally proved by Ramsey is in fact the restriction of the previous theorem to the case  $n = 2$ , i.e., to pairs of integers. From  $n > 1$ , the proof is non-trivial and will be studied in Section 2.

### Notation

We denote Ramsey's theorem for  $n$ -tuples of integers and  $k$  colors by  $\text{RT}_k^n$ .

$\text{RT}_k^n$  can be seen as the problem whose instances are colorings  $f : [\mathbb{N}]^n \rightarrow k$ , and whose solutions are infinite homogeneous sets. The statement of the generalized Ramsey theorem may seem rather abstract at first glance. Let's look at its meaning for small values of  $n$ .

*Case  $n = 1$ .* This is quite simply the infinite pigeonhole principle, which says that if we color an infinite set using a finite number of colors, then an infinity of elements are assigned the same color. More formally,  $\text{RT}_k^1$  states, for any  $k$ -partition  $A_0 \sqcup \dots \sqcup A_{k-1} = \mathbb{N}$ , the existence of an infinite set  $H \subseteq A_i$  for a color  $i < k$ .

0	1	3	8	...
	2	5	6	...
		4	7	...

Figure 1.2: Instance of  $\text{RT}_3^1$  where each row corresponds to a color. If the first row is infinite,  $H = \{0, 1, 3, 8, \dots\}$  is a solution, but also every infinite subset of  $H$ .

The infinite pigeonhole principle is combinatorially trivial. From a computational point of view and from Proof Theory, the situation is more delicate. In the same way as combinatorially, the pigeonhole principle is the most fundamental principle at the origin of Ramsey's theory, we will see that the greatest part of the computational phenomena of Ramsey's theorem are found in the pigeonhole principle.

*Case  $n = 2$ .* Ramsey's theorem for pairs is the historical theorem proved by Frank Ramsey. It can be formulated in terms of cliques as follows: a *clique* is a graph whose vertices are pairwise connected. Ramsey's theorem for pairs and  $k$  colors ( $\text{RT}_k^2$ ) states, for all  $k$ -coloring of the edges of an infinite clique, the existence of an infinite subset of vertices whose induced subgraph is monochromatic. In the case of  $k = 2$ , we can also consider the presence and the absence of edges as each of the two colors.  $\text{RT}_2^2$  therefore states, for any infinite graph, the existence of a subset of vertices whose induced subgraph is either a clique or an anti-clique (with no edge).

Ramsey's theorem for pairs is combinatorially much less trivial than that for singletons, and this complexity translates computably. We will see that  $\text{RT}_k^2$  escapes the phenomenon of the computational structure of Reverse Mathematics. The proof of Ramsey's theorem for pairs is an interest-

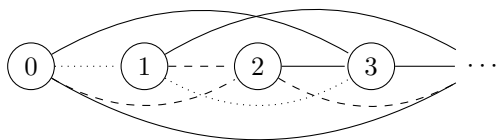


Figure 1.3: Instance of  $\text{RT}_3^2$ . The edge patterns correspond to colors. A solution is an infinite subset of vertices, whose edges share the same pattern.

ing exercise which requires a little advanced manipulation of infinite sets. We encourage the reader to seek proof of this for themselves.

**Exercise 1.4. (★)** Prove Ramsey's theorem for pairs.  $\diamond$

Ramsey's theorem admits a finite version which follows from its infinite version by compactness:

**Theorem 1.5 (Finite Ramsey's theorem)**

*For any  $n, k, p \in \mathbb{N}^*$ , there exists some  $m$  sufficiently large such that any  $k$ -coloring of  $[\{1, \dots, m\}]^n$  admits a homogeneous set of size  $p$ .*

PROOF. Suppose that Finite Ramsey's theorem is false. Then, there exists  $n, k, p$  such that for all  $m$  there exists a  $k$ -coloring of  $[\{1, \dots, m\}]^n$  with no homogeneous set of size  $p$ . We can then build a tree whose nodes of depth  $m$  are the  $k$ -colorings of  $[\{1, \dots, m\}]^n$  admitting no homogeneous set of size greater than  $p$ . Since there are only a finite number of possible  $k$ -coloring over  $[\{1, \dots, m\}]^n$ , our tree is finitely-branching.

By König's lemma, the tree therefore contains an infinite path, which is a coloring of  $[\mathbb{N}]^n$  with no homogeneous set of size greater than  $p$ , which contradicts the infinite Ramsey theorem.  $\blacksquare$

The optimal bound  $m$  as a function of  $n, k$  and  $p$  is called *Ramsey number*. The determination of Ramsey numbers as a function of its parameters is still a subject of research in Combinatorics.

**Ramsey's theorem and Weihrauch degrees**

There are two possible translations of Ramsey's theorem into a mathematical problem: in all cases, an instance is a coloring  $f : [\mathbb{N}]^n \rightarrow k$ . On the other hand, in the first case, a solution is a homogeneous set  $H$  for the instance  $f$ , while in the second case, the solution is of the form  $i \cap H$ , where  $H$  is homogeneous in color  $i$  for the instance  $f$ . We will call these problems  $\text{RT}_k^n$  and  $\text{cRT}_k^n$ , respectively. This distinction does not mat-

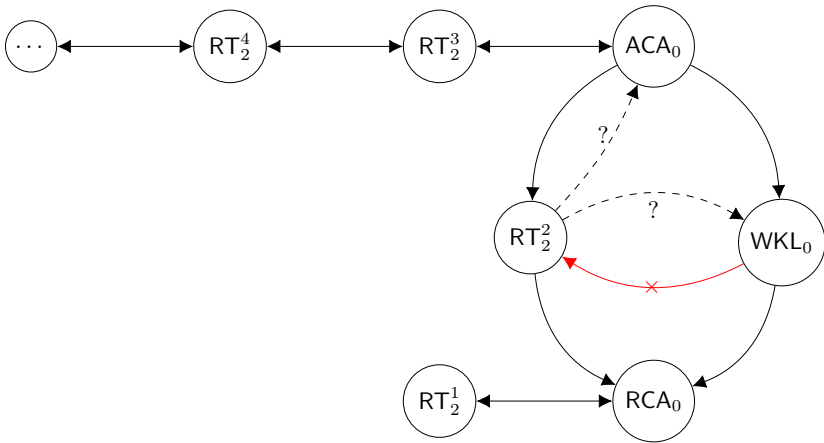


Figure 2.1: Jockusch's work. The implications which do not belong to the transitive closure do not hold.

ter for the Weihrauch reduction because, having access to the instance and the homogeneous set, it is possible to find the color of homogeneity. Thus,  $RT_k^n \equiv_W cRT_k^n$  for all  $n, k \geq 1$ . On the other hand, if we consider the strong Weihrauch reduction, where we lose access to the instance, the homogeneity color cannot be recovered in general, and we can prove that  $cRT_k^n \not\leq_{sW} RT_k^n$  for every  $n, k \geq 1$ . Ramsey's theorem from the point of view of Weihrauch degrees has been systematically studied by Brattka and Rakotoniaina [25].

## 2. Ramsey's theorem in the arithmetic hierarchy

Our adventure begins with Specker [212], who shows that  $RCA_0$  is not enough to prove Ramsey's theorem for pairs, by constructing a computable coloring with no infinite computable homogeneous set. A year later, in 1972, Carl Jockusch [98] improves this result in a seminal article, which marks the starting point of an in-depth study of Ramsey's theorem, and it is essentially this first work of Jockusch that we present in this section, summarized by Figure 2.1.

As illustrated in Figure 2.1, Jockusch's 1972 article left open two questions: the provability of  $ACA_0$  or that of  $WKL_0$  in  $RCA_0 + RT_2^2$ . It will take 23 years for David Seetapun to resolve the first in the negative (see Corollary 4.15),

and it will not be until 2012 that the second question will also be answered negatively, by Lu Liu (see Corollary 4.17), an undergraduate student at that time.

### 2.1. Preliminary study

Before embarking on the proof of Ramsey's theorem, we will prove two propositions useful for the rest of our analysis. The first proposition is a “local” version of Ramsey's theorem, which expresses in a way that it is a statement about sets, in that it does not depend on the nature of the set, but only on its cardinality.

**Proposition 2.2.**  $\text{RCA}_0 + \text{RT}_k^n$  proves that for any infinite set  $X \subseteq \mathbb{N}$  and any  $k$ -coloring  $f$  of  $[X]^n$ , there exists an infinite homogeneous set  $H \subseteq X$  for  $f$ . ★

PROOF. Let  $X$  be an infinite set and  $f : [X]^n \rightarrow k$  fixed. Let  $g : \mathbb{N} \rightarrow X$  be the canonical bijection which to  $m$  associates the  $(m + 1)$ -th element of  $X$ . We can then define a coloring  $h : [\mathbb{N}]^n \rightarrow k$  by  $h(\{x_1, \dots, x_n\}) = f(\{g(x_1), \dots, g(x_n)\})$ . Let  $Y \subseteq \mathbb{N}$  be an infinite homogeneous set for  $h$ . Then, the set  $H = \{g(x) : x \in Y\} \subseteq X$  is an infinite homogeneous set for  $f$ . We will verify that, under the assumption that  $X$  is infinite,  $\Sigma_1^0$  induction on the parameter  $X$  is sufficient to show that  $g$  is total and therefore that  $H$  is  $\Delta_1^0$  (see Section 23-3). ■

The second proposition shows that the number of colors  $k$  of the coloring has little impact in Reverse Mathematics. Indeed,  $\text{RT}_{k+1}^n$  trivially implies  $\text{RT}_k^n$  because any  $k$ -coloring is a  $(k + 1)$ -coloring. Conversely, a “color blindness” argument allows to prove  $\text{RT}_{k+1}^n$  from  $\text{RT}_k^n$ .

**Proposition 2.3.**  $\text{RCA}_0 \vdash \forall n \forall k \geq 2 (\text{RT}_k^n \rightarrow \text{RT}_{k+1}^n)$ . ★

PROOF. Suppose  $\text{RT}_k^n$  true. Consider a  $(k + 1)$ -coloring  $f : [\mathbb{N}]^n \rightarrow k + 1$ . Let us then define the  $k$ -coloring  $g : [\mathbb{N}]^n \rightarrow k$  which to any  $n$ -tuple  $\{x_1, \dots, x_n\}$ , associates the color  $f(\{x_1, \dots, x_n\})$  if  $f(\{x_1, \dots, x_n\}) < k$ , and the color  $k - 1$  if  $f(\{x_1, \dots, x_n\}) = k$ . By  $\text{RT}_k^n$ , there exists an infinite homogeneous set  $X$  for  $g$ . If the color of  $X$  is among  $\{0, \dots, k - 2\}$ , then  $X$  is also homogeneous for  $f$ . Suppose that  $X$  is  $g$ -homogeneous for color  $k - 1$ . Then,  $f$  restricted to  $[X]^n$  is a 2-coloring. By the local version of  $\text{RT}_k^n$  (see Proposition 2.2) applied to  $f : [X]^n \rightarrow \{k - 1, k\}$ , there exists an infinite set  $H \subseteq X$  homogeneous for  $f$ . ■

Note that the proof of Proposition 2.3 uses two instances of  $\text{RT}_k^n$  to prove  $\text{RT}_{k+1}^n$ . These are necessary multiple applications: we can show  $\text{RT}_{k+1}^n \not\leq_c \text{RT}_k^n$  for

any  $n, k \geq 1$  (see Patey [171]). In the following, we will allow ourselves to use the local version of Ramsey's theorem by Proposition 2.2, and we will mainly consider the case  $k = 2$  by virtue of Proposition 2.3.

Ramsey's theorem is proved inductively on the size of the colored  $n$ -tuples. The base case is the infinite pigeonhole principle  $\text{RT}_2^1$ .

**Proposition 2.4.**  $\text{RCA}_0 \vdash \text{RT}_2^1$ . ★

PROOF. Let  $f : \mathbb{N} \rightarrow 2$  be a coloring, and let  $C_i = \{x \in \mathbb{N} : f(x) = i\}$ . Note that  $C_i$  exists for all  $i < 2$  by  $\Delta_0^0$ -comprehension parameterized by  $f$ . If  $C_i$  is infinite for an  $i < 2$ , then  $C_i$  is an infinite homogeneous set for  $f$ . Otherwise, let  $n_0$  and  $n_1$  be the respective bounds of  $C_0$  and  $C_1$ . Then,  $\max(n_0, n_1)$  is a bound of  $C_0 \cup C_1 = \mathbb{N}$ , contradiction. ■

It follows that  $\text{RT}_k^1$  is provable in  $\text{RCA}_0$  for all  $k \in \mathbb{N}$ , and that  $\text{RT}_k^1$  is a computably true statement, in the sense that it always has a computable solution in its instance. We will study in more detail the infinite pigeonhole principle — and its unsuspected riches — in Section 3.

## 2.2. Ramsey's theorem for pairs

We now get to the heart of the matter by starting with Ramsey's theorem for pairs, and establishing very precise bounds on the complexity of solutions from the point of view of the arithmetic hierarchy.

### Theorem 2.5 (Jockusch [98])

For any  $k \geq 1$ , any instance  $f$  of  $\text{RT}_k^2$  has a  $\Pi_2^0(f)$  solution.

PROOF. *Preliminary explanations:* To simplify the notations, we are going to prove the case  $k = 2$ . The general case is a simple adaptation of the construction. Let  $f : [\mathbb{N}]^2 \rightarrow 2$  be a coloring. The goal is to build a  $\Pi_2^0(f)$  sequence of elements  $M = \{x_0 < x_1 < x_2 < \dots\}$  such that for all  $x_i$ , the color  $f(\{x_i, x_j\})$  is the same for all  $x_j > x_i$ . Let  $g : \mathbb{N} \rightarrow 2$  be the function which to  $i$  associates  $f(\{x_i, x_{i+1}\})$ . We will also ensure that the set  $Z = \{x_i \in M : g(x_i) = 0\}$  is  $\Pi_2^0(f)$ . We will have two cases: either  $Z$  is infinite, in which case we have an infinite  $\Pi_2^0(f)$  homogeneous set for color 0, or  $Z$  is finite, in which case  $M \setminus Z$  is also  $\Pi_2^0(f)$ , and is an infinite homogenous set for color 1.

Each element  $x_i$  will be approximated by a sequence  $(x_i^s)_{s \in \mathbb{N}}$ . Initially, each  $x_i^0$  is undefined. We will also use an approximation of sets  $(X_i^s)_{s \in \mathbb{N}}$  which are also initially all undefined. Each set  $X_i^s$  that is defined will initially be equal to  $\{x \in X_{i-1}^s : f(x_i^s, x) = 0\}$  assuming at the beginning that it is an infinite set. Then, if this is not the case, we will change it so

that it becomes equal to  $\{x \in X_{i-1}^s : f(x_i^s, x) = 1\}$ , and so on. We will also use a lower bound  $b_s$  on the new  $x_i^s$  that we want to define. The usefulness of this bound is essentially to be able to make our construction  $\Pi_2^0(f)$ .

*Construction:* In step 0, we define  $x_0^0 = 0$ ,  $X_0^0 = \{x : f(x_0^0, x) = 0\}$  and  $b_0 = 0$ . Suppose that in a step  $s$ , we have defined  $x_0^s < \dots < x_n^s$  for a certain  $n$  (the following  $x_{n+1}^s$  being undefined), with computable sets  $X_0^s \supseteq \dots \supseteq X_n^s$ . In step  $s+1$ , we search using  $\theta'$  for the largest  $i \leq n$  such that  $X_i^s$  has at least one element strictly greater than  $b_s$ . If such an  $i \geq 0$  does not exist, we set  $i = -1$ .

Case 1: If  $i = n$ , then we set  $X_j^{s+1} = X_j^s$  and  $x_j^{s+1} = x_j^s$  for  $j \leq n$ , we define  $x_{n+1}^{s+1}$  as being the smallest element of  $X_n^{s+1}$  strictly greater than  $b_s$ , and we define  $X_{n+1}^{s+1} = \{x \in X_n^{s+1} : f(x_{n+1}^{s+1}, x) = 0\}$ . We finally define  $b_{s+1} = x_{n+1}^{s+1}$ .

Case 2: If  $i < n$ , then for all  $j \leq i$ , we define  $X_j^{s+1} = X_j^s$  and  $x_j^{s+1} = x_j^s$ . Then, we define  $x_{i+1}^{s+1} = x_{i+1}^s$  and  $X_{i+1}^{s+1} = \{x \in X_i^{s+1} : f(x_{i+1}^{s+1}, x) = 1\}$ . For all  $j > i+1$ , the values  $x_j^{s+1}$  and  $X_j^{s+1}$  become undefined again if they were not already. We finally define  $b_{s+1} = b_s$ . This concludes the construction.

*Verification:* Let us show that each sequence  $(x_n^s)_{s \in \mathbb{N}}$  is eventually defined and converges to a value  $x_n$ , as well as each sequence  $(X_n^s)_{s \in \mathbb{N}}$ , which converges to a set  $X_n$  such that  $|X_n| = \infty$ . It is clear that  $(x_0^s)_{s \in \mathbb{N}}$  is always equal to 0. If the set  $X_0^0$  is infinite then it is also clear that it will never change. If it is finite, then by construction it will change only once to become co-finite and will therefore never change again from that point on. Suppose now that in step  $s$ , the variables  $x_0^s, \dots, x_n^s$  and  $X_0^s, \dots, X_n^s$  have reached their convergence values  $x_0, \dots, x_n$  and  $X_0, \dots, X_n$  with  $X_n$  infinite. There again, by construction, the sequence  $(x_{n+1}^t)_{t > s}$  will be defined and constant at each step  $t$ . It follows that if  $X_{n+1}^t \subseteq X_n$  is infinite for  $t > s$ , it will never change, otherwise it will change once to become co-finite.

Finally, let us show that the sequence  $x_0 < x_1 < \dots$  is  $\Pi_2^0(f)$ . For this, we show that it is co-enumerated using  $f'$ . You just need to use the  $b_s$  marker. Indeed, at step  $s+1$ , the element that is added is necessarily greater than  $b_s$ . Items that change from undefined to defined in step  $s$  are necessarily greater than  $b_s$ . The co-enumeration is therefore done at each step  $s$  by removing all the elements smaller than  $b_s$ , and which are different from a certain  $x_n^s$  for  $x_n^s$  defined. Also note that an element  $x_s^n$  changes its limit color at most once, from 0 to 1. Therefore, the set of  $x_i$  whose limit color is 0 is  $\Pi_2^0(f)$ . ■

Note that the previous theorem constitutes a proof of Ramsey's theorem for pairs. The fact that the exhibited solution is computable using the double jump suggests that the proof was made in  $\text{ACA}_0$ . To be sure, it is necessary to check that the proof of the fact that the  $\Pi_2^0(f)$  set which is constructed is indeed infinite and homogeneous for  $f$ , does not use more than the arithmetic induction. The reader will be able to note that this is indeed the case, and we will see otherwise that it is a consequence of Theorem 2.17.

Is the  $\Pi_2^0(f)$  upper bound of the previous theorem optimal from the point of view of the arithmetic hierarchy? Jockusch answered positively, via the following theorem.

**Theorem 2.6 (Jockusch [98])**

*There is a computable instance of  $\text{RT}_2^2$  which has no  $\Sigma_2^0$  solution.*

PROOF. Let us first show that there exists a computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$  with no infinite  $\Delta_2^0$  homogeneous set. For all  $e, s \in \mathbb{N}$ , let  $A_{e,s} = \{x < s : \Phi_e(\emptyset'_s, x)[s] \downarrow = 1\}$ . Note that  $(A_{e,s})_{e,s \in \mathbb{N}}$  is uniformly computable. Moreover, if  $x \mapsto \Phi_e(\emptyset', x)$  is a total function, then for all  $x$ ,  $\Phi_e(\emptyset', x) = 1$  iff  $\forall^\infty s \ x \in A_{e,s}$ . Note that if  $\Phi_e(\emptyset', x) \uparrow$  for  $x, e$ , then  $x$  can enter and exit  $A_{e,s}$  for a infinitely many  $s$ .

We define  $f : [\mathbb{N}]^2 \rightarrow 2$  as follows: at the stage of computation  $s$ , for all  $e < s$ , if ever  $|A_{e,s}| \geq 2(e+1)$ , then we choose the two smallest distinct elements  $a, b \in A_{e,s}$  such that  $f(\{a, s\})$  and  $f(\{b, s\})$  are not yet defined, and we define  $f(\{a, s\}) = 0$  and  $f(\{b, s\}) = 1$ .

Note that if  $|A_{e,s}| \geq 2(e+1)$ , there are necessarily always two elements  $a, b \in A_{e,s}$  on which  $f(\{a, s\})$  and  $f(\{b, s\})$  are not yet defined: by induction on  $e < s$ ,  $f(\{x, s\})$  is defined for at most  $2e$  values of  $x$  before choosing the elements of  $A_{e,s}$ . Then, if  $|A_{e,s}| \geq 2(e+1)$ , there are always at least two values available.

Let us show that  $f$  does not admit any infinite  $\Delta_2^0$  homogeneous set. Let  $A$  be an infinite  $\Delta_2^0$  set. Then, there exists  $e$  such that for all  $x$ ,  $x \in A$  iff  $\forall^\infty s \ x \in A_{e,s}$ . Let  $n \in \mathbb{N}$  be the smallest integer such that  $A \upharpoonright_n$  has  $2(e+1)$  elements, and let  $s \in A$  be sufficiently large such that  $A_{e,s} \upharpoonright_n = A \upharpoonright_n$ . In particular,  $|A_{e,s}| \geq 2(e+1)$ , and we will choose two elements  $a, b \in A \upharpoonright_n$  such that  $f(\{a, s\}) \neq f(\{b, s\})$ . As  $a, b, s \in A$ , the set  $A$  is not homogeneous for  $f$ .

Finally, it suffices to show that any infinite  $\Sigma_2^0$  set contains an infinite  $\Delta_2^0$  subset: given  $A = \{n : \exists x \forall y \ R(x, y, n)\}$  for a computable predicate  $R$ , using  $\emptyset'$  we look for  $n_0$  and  $x$  such that  $\forall y \ R(x, y, n_0)$ , then we look for  $n_1 > n_0$  and  $x$  such that  $\forall y \ R(x, y, n_1)$ , etc. As any homogeneous subset of a

homogeneous set is also homogeneous, a coloring which has no infinite  $\emptyset'$ -computable homogeneous set also has no infinite  $\Sigma_2^0$  homogeneous set. ■

In particular, the simple jump is not sufficient to compute an infinite homogeneous set for a computable coloring of pairs. This implies in particular that  $\text{RT}_2^2$  is not provable from weak König's lemma.

**Corollary 2.7**

$\text{RT}_2^2 \not\leq_\omega \text{WKL}$ . In particular,  $\text{WKL}_0 \not\vdash \text{RT}_2^2$ .

PROOF. By Exercise 22-7.5, there exists a Scott ideal  $\mathcal{I}$  containing only low sets, and therefore  $\Delta_2^0$  sets. By Proposition 22-7.2,  $\mathcal{I} \models \text{WKL}$ . On the other hand, by Theorem 2.6, there exists a computable instance  $f : [\mathbb{N}]^2 \rightarrow 2$  of  $\text{RT}_2^2$  with no  $\Delta_2^0$  solution. In particular,  $f \in \mathcal{I}$ , but  $f$  does not admit a solution in  $\mathcal{I}$ , so  $\mathcal{I} \not\models \text{RT}_2^2$ . It follows that  $\text{RT}_2^2 \not\leq_\omega \text{WKL}$ , therefore  $\text{WKL}_0 \not\vdash \text{RT}_2^2$ . ■

### 2.3. Generalized Ramsey theorem

We will now extend the previous analysis to Ramsey's theorem generalized to  $n$ -tuples. The inductive proof of Ramsey's theorem is done using the notion of pre-homogeneous set, which makes it possible to reduce a coloring of  $(n+1)$ -tuples to a coloring of  $n$ -tuples.

**Definition 2.8.** Let  $n, k \geq 1$  and  $f : [\mathbb{N}]^{n+1} \rightarrow k$  be a coloring. A set  $X \subseteq \mathbb{N}$  is *pre-homogeneous* for  $f$  if the color of any  $(n+1)$ -tuple of  $X$  depends only on the  $n$  first elements of the  $(n+1)$ -tuple. Formally for  $x_1 < \dots < x_n \in X$  fixed, for any  $y_1, y_2 \in X$  with  $y_1, y_2 > x_n$  we have  $f(\{x_1, \dots, x_n, y_1\}) = f(\{x_1, \dots, x_n, y_2\})$ . ◇

Pre-homogeneous sets form a bridge between the instances of  $\text{RT}_k^{n+1}$  and those of  $\text{RT}_k^n$  in the following direction.

**Fact**

Let us fix  $n, k \geq 1$ , a coloring  $f : [\mathbb{N}]^{n+1} \rightarrow k$  and an infinite set  $X$  pre-homogeneous for  $f$ . Let  $g : [\mathbb{N}]^n \rightarrow k$  be the coloring which to  $\{x_1, \dots, x_n\} \in [X]^n$  associates  $f(\{x_1, \dots, x_n, y\})$  for any  $y \in X$  satisfying  $y > x_n$ . Then, any homogeneous set  $Y \subseteq X$  for  $g$  is homogeneous for  $f$ .

It is therefore a matter of proving, for any coloring  $f : [\mathbb{N}]^{n+1} \rightarrow k$ , the existence of an infinite pre-homogeneous set for  $f$ .

**Proposition 2.9.** Let  $f : [\mathbb{N}]^{n+1} \rightarrow k$  be a coloring. There exists an  $f$ -computable, infinite finitely-branching tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ , such that any path is an infinite pre-homogeneous set for  $f$ . In particular, any PA degree relative to  $f'$  computes an infinite pre-homogeneous set for  $f$ . ★

PROOF. The tree we are going to build is known as *Erdős-Rado's tree*. Let  $T_0$  be the tree containing only the node 0. We will inductively define at each step  $s + 1$  a tree  $T_{s+1}$  which will contain the nodes of  $T_s$ , to which we will add the node  $\sigma \hat{\ } (s + 1)$  for a certain  $\sigma \in T_s$ .

In steps  $0 < s < n$ , we put in the tree  $T_s$  the integer  $s$  as child of the node  $0 \hat{\ } 1 \hat{\ } 2 \hat{\ } \dots \hat{\ } s - 1 \in T_{s-1}$ . Let  $T_s$  be the result of the computation step  $s \geq n - 1$ . In step  $s + 1$ , we will browse the tree  $T_s$  from its root to find the node  $\sigma \in T_s$  to which we will add  $s + 1$  as a child of  $\sigma$  in  $T_{s+1}$ . We define  $\sigma_0 = 0 \hat{\ } 1 \hat{\ } 2 \hat{\ } \dots \hat{\ } n - 1 \in T_s$ , then at each step  $i$ , assuming  $\sigma_i$  defined, we look for  $a$  such that  $\sigma_i a \in T_s$  and such that  $f(\sigma(m_1), \dots, \sigma(m_n), s + 1) = f(\sigma(m_1), \dots, \sigma(m_n), a)$  for all  $m_1 < \dots < m_n < |\sigma_i|$ . If we find such an integer  $a$  we define  $\sigma_{i+1} = \sigma_i a$ . Otherwise our search stops there and we define  $T_{s+1}$  as  $T_s \cup \{\sigma_i \hat{\ } (s + 1)\}$ .

The tree  $T = \bigcup_s T_s$  is computable and infinite because  $|T_{s+1}| = |T_s| + 1$  for all  $s$ . It is also finitely-branching because for any node  $\sigma a \in T$ , there is only a finite number of possibilities for the combinations of values  $f(\sigma(m_1), \dots, \sigma(m_n), a)$  for any  $m_1 < \dots < m_n < |\sigma|$ . Finally, by construction, it is clear that any path constitutes an infinite pre-homogeneous set. ■

Proposition 2.9 allows to transfer the upper bound of Jockusch of Ramsey's theorem for pairs to the generalized Ramsey theorem.

**Corollary 2.10 (Jockusch [98])**

Let  $n, k \geq 1$ . Any instance  $f$  of  $\text{RT}_k^n$  has a  $\Pi_n^0(f)$  solution.

PROOF. We proceed by induction on  $n$ . For  $n = 1$ , any  $f$ -computable instance of  $\text{RT}_k^1$  admits an  $f$ -computable solution, so in particular  $\Pi_1^0(f)$ . For  $n = 2$ , this is Theorem 2.5. Suppose the theorem is true for  $n$ , and let us show that it is true for  $n + 1$ . Let  $f : [\mathbb{N}]^{n+1} \rightarrow k$  be a coloring. By Theorem 8-4.3, there exists a set  $P$  of PA degree relative to  $f'$  such that  $P' \leq_T f''$ . By Proposition 2.9,  $P$  computes an infinite set  $X$  pre-homogeneous for  $f$ . Let  $g : [X]^n \rightarrow k$  be the  $P$ -computable coloring which to  $\{x_1, \dots, x_n\}$  associates  $f(\{x_1, \dots, x_n, y\})$  for any  $y \in X$  such that  $y > x_n$ . By induction hypothesis relativized to  $P$ , there exists a  $\Pi_n^0(g)$  infinite set  $Y \subseteq X$  homogeneous for  $g$ , therefore homogeneous for  $f$ . In particular,  $Y$  is  $\Pi_{n-1}^0(g')$ , so  $\Pi_{n-1}^0(f'')$ , and therefore  $\Pi_{n+1}^0(f)$ . ■

The notion of pre-homogeneous set gives the possibility, via Proposition 2.9, to transform a coloring on the  $(n+1)$ -tuples into a coloring on the  $n$ -tuples computable in any PA degree relative to  $\emptyset'$ . The following proposition is a kind of partial reciprocal.

**Proposition 2.11.** For any  $\emptyset'$ -computable coloring  $f : [\mathbb{N}]^n \rightarrow k$ , there exists a computable coloring  $g : [\mathbb{N}]^{n+1} \rightarrow k$  such that any infinite homogeneous set for  $g$  is homogeneous for  $f$ . ★

PROOF. Let  $f : [\mathbb{N}]^n \rightarrow k$  be a  $\emptyset'$ -computable coloring. By the Shoenfield limit lemma (see Lemma 4-7.2), there exists a  $\Delta_2^0$  approximation of the function  $f$ . We can see this  $\Delta_2^0$  approximation as a computable function  $g : [\mathbb{N}]^{n+1} \rightarrow k$  whose last parameter corresponds to the approximation time. Thus, for  $\{x_1, \dots, x_n\} \in [\mathbb{N}]^n$ ,  $\lim_z g(\{x_1, \dots, x_n, z\}) = f(\{x_1, \dots, x_n\})$ . Let  $H$  be an infinite homogeneous set for  $g$ , of color  $i < k$ . Let us show that  $H$  is homogeneous for  $f$  of color  $i$ . Let  $\{x_1, \dots, x_n\} \in [H]^n$ . Let  $y \in H$  be large enough so that we have  $g(\{x_1, \dots, x_n, y\}) = \lim_z g(\{x_1, \dots, x_n, z\})$ . In particular  $\lim_z g(\{x_1, \dots, x_n, z\}) = i$ , and so  $f(\{x_1, \dots, x_n\}) = i$ . ■

Note that the function  $g$  built in the previous proposition can have finite homogeneous sets which are not homogeneous for  $f$ . Just as the notion of pre-homogeneous set enabled to obtain an upper bound on the complexity of the generalized Ramsey theorem, Proposition 2.11 allows to transfer the lower bound of Jockusch to colorings on the  $n$ -tuples.

**Corollary 2.12 (Jockusch [98])**

*For any  $n \geq 2$ , there exists a computable instance of  $\text{RT}_2^n$  with no  $\Sigma_n^0$  solution.*

PROOF. Let us show by induction on  $n \geq 2$  that for all  $Z$ , there exists a  $Z$ -computable instance of  $\text{RT}_2^n$  which has no  $\Sigma_n^0(Z)$  solution. The case  $n = 2$  is a relativization of Theorem 2.6. Suppose true the case  $n$ , and let us show the case  $n + 1$ . Let  $Z$  be fixed. By induction hypothesis applied to  $Z'$ , there exists a  $Z'$ -computable instance  $f$  of  $\text{RT}_2^n$  which has no  $\Sigma_n^0(Z')$  solution, therefore no  $\Sigma_{n+1}^0(Z)$  solution. By Proposition 2.11 relativized, there exists a  $Z$ -computable instance  $g$  of  $\text{RT}_2^{n+1}$  such that any infinite homogeneous set for  $g$  is homogeneous for  $f$ . In particular,  $g$  does not have any  $\Sigma_{n+1}^0(Z)$  solution. ■

The situation of Ramsey's theorem with respect to the arithmetic hierarchy is therefore as follows.

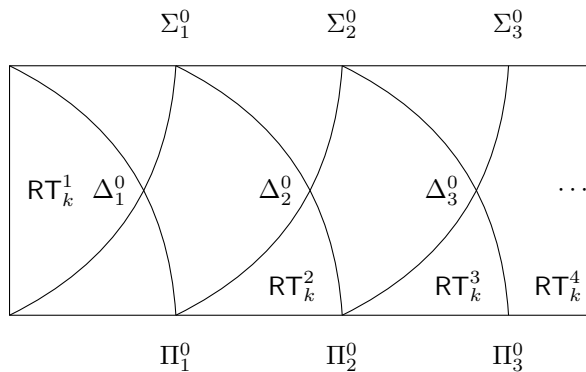


Figure 2.13: Ramsey's theorem in the arithmetic hierarchy

Jockusch's theorems chisel with clockwork precision the contours of the arithmetic complexity of the possible solutions for the generalized Ramsey theorem: for any  $n$ , any computable instance of  $\text{RT}_2^n$  has a  $\Pi_n^0$  solution, and one of these instances has no  $\Sigma_n^0$  solution. However, these theorems do not tell us about the “computational force” of Ramsey's theorem: we know how to create computable instances which require a lot of computational power to find solutions, but we do not know how to create computable instances whose all possible solutions have a lot of computational power. This is typically the kind of construction we need to do to show for example  $\text{RCA}_0 \vdash \text{RT}_k^n \rightarrow \text{ACA}_0$ .

Let us take the example of  $\text{RT}_2^2$ , whose computable instances always admit  $\Pi_2^0$  solutions. The double jump can therefore always compute solutions of  $\text{RT}_2^2$ . Would it be possible to build an instance, all the solutions of which compute the double jump? Jockusch answered this first question in the negative, showing that a  $\text{PA}(\emptyset')$  degree was in fact sufficient to compute solutions:

**Theorem 2.14 (Jockusch [98])**

*For any  $n, k \geq 1$  and any coloring  $f : [\mathbb{N}]^n \rightarrow k$ , any PA degree relative to  $f^{(n-1)}$  computes an infinite homogeneous set for  $f$ .*

PROOF. By induction on  $n \geq 1$ . For  $n = 1$ ,  $f$  computes an infinite homogeneous set for  $f$ . Any PA degree relative to  $f$  computes  $f$ , therefore computes an infinite homogeneous set for  $f$ .

Suppose this is the case for  $n \geq 1$ . Let  $f : [\mathbb{N}]^{n+1} \rightarrow k$  be a coloring, and  $P_0$  a  $\text{PA}(f')$  set such that  $P'_0 \leq_T f''$  and let  $P_1$  be a  $\text{PA}(f^{(n)})$  set. By Proposition 2.9,  $P_0$  computes an infinite set  $X$  pre-homogeneous for  $f$ . Let  $g : [X]^n \rightarrow k$  be the  $P_0$ -computable coloring which to  $\{x_1, \dots, x_n\}$

associates  $f(\{x_1, \dots, x_n, y\})$  for any  $y \in X$  such that  $y > x_n$ . In particular,  $P_0^{(n-1)} \leq_T f^{(n)}$ , therefore  $P_1$  is PA  $(P_0^{(n-1)})$  and therefore PA  $(g^{(n-1)})$ . By induction hypothesis,  $P_1$  computes an infinite set  $H \subseteq X$  homogeneous for  $g$ , therefore homogeneous for  $f$ . ■

In particular any PA  $(\emptyset')$  degree is sufficient to compute a solution to any computable instance of  $\text{RT}_2^2$ . Conversely, is it possible to find an instance of  $\text{RT}_2^2$  whose solutions are all of PA  $(\emptyset')$  degree? Jockusch could only show this for instances of  $\text{RT}_2^3$  and not  $\text{RT}_2^2$ . We will see in fact in Section 4 that it is impossible “encode” computational power in the possible solutions of a computable instance of  $\text{RT}_2^2$ . On the other hand, this is the case for those of  $\text{RT}_2^3$ :

**Exercise 2.15. (★)** Build a  $\emptyset'$ -computable instance of  $\text{RT}_2^2$  whose solutions all compute  $\emptyset'$ . Deduce that there are computable instances of  $\text{RT}_2^3$  whose solutions all compute  $\emptyset'$ .

Note: use the notion of modulus (see Definition 4-7.7). ◇

It is possible to reinforce the previous exercise, and to encode in the computable instances of  $\text{RT}_2^3$ , the power necessary for the computation of pre-homogeneous sets.

**Theorem 2.16 (Hirschfeldt and Jockusch [87])**

*There is a computable coloring  $f : [\mathbb{N}]^3 \rightarrow 2$  such that any infinite pre-homogeneous set for  $f$  is of PA degree relative to  $\emptyset'$ .*

PROOF. Let  $\Phi_0, \Phi_1, \dots$  be an enumeration of all  $\{0, 1\}$ -valued Turing functionals. We can assume without loss of generality that  $\Phi_e(\emptyset'_s, e)[s]$  returns a value at any computation time  $s$  (by assigning for example 0 if  $\Phi_e(\emptyset'_s, e)[s] \uparrow$ ). Let  $f : [\mathbb{N}]^3 \rightarrow 2$  be defined for all  $m < s < t$  by  $f(\{m, s, t\}) = 1$  if for all  $e < m$ ,  $\Phi_e(\emptyset'_s, e)[s] = \Phi_e(\emptyset'_t, e)[t]$ . Otherwise  $f(\{m, s, t\}) = 0$ . Let  $X$  be an infinite pre-homogeneous set for  $f$ . Let  $g : \mathbb{N} \rightarrow 2$  be the function which on  $e$ , searches for an integer  $m \in X$  such that  $e < m$ , and integers  $s, t \in X$  such that  $m < s < t$  and  $\Phi_e(\emptyset'_s, e)[s] = \Phi_e(\emptyset'_t, e)[t]$ , and returns  $1 - \Phi_e(\emptyset'_s, e)[s]$ . Note that such a search necessarily succeeds because there are only two possible values for each  $\Phi_e$ .

By pre-homogeneity of  $X$ ,  $g(e) = 1 - \Phi_e(\emptyset'_t, e)[t]$  for an infinity of  $t \in X$ . The function  $g$  is total  $X$ -computable. Let us show that  $g$  is a DNC function relative to  $\emptyset'$ . Let  $e$  be such that  $\Phi_e(\emptyset', e) \downarrow$ . Then,  $\lim_{t \in X} \Phi_e(\emptyset'_t, e)[t]$  exists, so  $g(e) = 1 - \Phi_e(\emptyset', e)$ . Any DNC function relative to  $\emptyset'$  with values in  $\{0, 1\}$  is of PA degree relative to  $\emptyset'$ . ■

The previous theorem is reformulated by  $\text{KL} \leq_c \text{RT}_2^3$  in the language of computable reductions.

## 2.4. Reverse Mathematics

The purely computational analysis performed by Jockusch [98] was formalized by Simpson [202] in second-order arithmetic to deduce the logical power of Ramsey's theorem in Reverse Mathematics. The proofs of Theorem 2.17 are essentially the constructions seen previously, but carefully analyzed to ensure that only the axioms available in  $\text{RCA}_0$  are used.

### Theorem 2.17 (Jockusch [98] and Simpson [202])

For all  $n \geq 3$ ,  $\text{RCA}_0 \vdash \forall k \text{RT}_k^n \leftrightarrow \text{ACA}_0$ .

PROOF. Let us show by induction on  $n \in \mathbb{N}$  that  $\text{ACA}_0 \vdash \forall k \text{RT}_k^n$ .  $\text{ACA}_0 \vdash \forall k \text{RT}_k^1$ , as we will see in Proposition 3.1. Suppose that  $\text{ACA}_0 \vdash \text{RT}_k^n$ . Let  $f : [\mathbb{N}]^{n+1} \rightarrow k$  be an instance of  $\text{RT}_k^{n+1}$ . Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be the tree of Erdős-Rado of Proposition 2.9. It is clear that the definition of  $T$  can be done primitively recursively with respect to  $f$ . According to Theorem 23-4.6,  $\Sigma_1^0$  induction is enough to show that the function which on  $n$  returns the node  $\sigma \hat{\ } n \in T$  is total, and therefore that the tree admits a  $\Delta_1^0$  definition. It therefore exists by the  $\Delta_1^0$  comprehension scheme, by definition it is infinite and  $\text{RCA}_0$  is sufficient to prove that it is infinite with finite branching. By König's lemma, provable in  $\text{ACA}_0$ , there exists a path  $P \in [T]$ , therefore an infinite pre-homogeneous set for  $f$ . Let  $g : [P]^n \rightarrow k$  be the coloring which to  $\{x_1, \dots, x_n\}$  associates  $f(\{x_1, \dots, x_n, y\})$  for any  $y \in X$  such that  $y > x_n$ . The coloring  $g$  exists by the  $\Delta_1^0$  comprehension scheme. By induction hypothesis and Proposition 2.2,  $\text{ACA}_0$  shows the existence of an infinite set  $H \subseteq P$  homogeneous for  $g$ . In particular,  $H$  is homogeneous for  $f$ .

Let us show that  $\text{RCA}_0 + \text{RT}_2^3 \vdash \forall X \exists Y \ Y = X'$ . Let  $X$  be a set, and let  $f : [\mathbb{N}]^3 \rightarrow 2$  be the function defined for  $x < s < t$  by  $f(\{x, s, t\}) = 1$  iff  $\forall e < x \ ((\Phi_e(X, e)[s] \downarrow \wedge \Phi_e(X, e)[t] \downarrow) \vee (\Phi_e(X, e)[s] \uparrow \wedge \Phi_e(X, e)[t] \uparrow))$ . Let  $H$  be an infinite homogeneous set for  $f$ . Let us show that  $H$  is homogeneous with color 1. Let  $x \in H$  and  $g : H \rightarrow 2^x$  defined by  $g(s) = \langle a_e : e < x \rangle$  where  $a_e = 1$  if  $\Phi_e(X, e)[s] \downarrow$  and  $a_e = 0$  otherwise.  $\text{RCA}_0$  proves that  $g$  is not injective, so there exists  $s < t \in H$  such that  $g(s) = g(t)$ . In particular,  $f(\{x, s, t\}) = 1$ . Let  $p_H : \mathbb{N} \rightarrow \mathbb{N}$  be the principal function of  $H$ . Then, the set  $Y = \{e \in \mathbb{N} : \Phi_e(X, e)[p_H(e)] \downarrow\}$  exists by the  $\Delta_1^0$  comprehension scheme, and  $Y = X'$ . ■

Note that the induction on  $n$  of the proof of Theorem 2.14 is done in the meta-theory. It is indeed an induction on a second-order formula.

Corollary 2.12 combined with the theorem of Jockusch and Solovay (see Theorem 22-8.1) allow us to show that the statement  $\forall n \text{RT}_2^n$  is not provable in  $\text{ACA}_0$ .

**Proposition 2.18.**  $\text{ACA}_0 \not\vdash \forall n \text{RT}_2^n$ . ★

PROOF. Suppose that  $\text{ACA}_0 \vdash \forall n \text{RT}_2^n$ . By the theorem of Jockusch and Solovay (see Theorem 22-8.1), there exists a bound  $k \in \mathbb{N}$  such that  $\text{ACA}_0$  proves that for all  $n \in \mathbb{N}$ , any instance  $f : [\mathbb{N}]^n \rightarrow 2$  admits a  $\Sigma_k^0(f)$  solution. In particular, for  $n = k$ , by Corollary 2.12, there exists a computable instance  $f : [\mathbb{N}]^k \rightarrow 2$  with no  $\Sigma_k^0$  solution. Contradiction. ■

As explained in Section 22-8.1, such a separation is necessarily in non-standard models since  $\text{ACA}'_0$  and  $\text{ACA}_0$  share the same  $\omega$ -models. By adapting the proof of Theorem 2.14, McAloon [152] proved that the statement  $\forall n \text{RT}_2^n$  was equivalent to  $\text{ACA}'_0$ .

The equivalence of Theorem 2.17 is not satisfactory from the point of view of Computability Theory insofar, as it is not sensitive to the levels of the arithmetic hierarchy. Corollary 2.12 shows that the larger the size of the  $n$ -tuples, the more computational power is required to find solutions. The proof over  $\text{RCA}_0$  of  $\text{RT}_2^n \rightarrow \text{RT}_2^{n+1}$  when  $n \geq 3$  is carried out in several steps: a first instance of  $\text{RT}_2^n$  allows to compute an infinite pre-homogeneous set for the instance of  $\text{RT}_2^{n+1}$ , which reduces it to a second instance of  $\text{RT}_2^n$ . The computational reduction, sensitive to the number of applications, makes it possible to account for this difference:

**Proposition 2.19.** For all  $n \geq 1$ ,  $\text{RT}_2^{n+1} \not\leq_c \text{RT}_2^n$ . ★

PROOF. By Corollary 2.12, there exists a computable instance  $f$  of  $\text{RT}_2^{n+1}$  with no  $\Sigma_{n+1}^0$  solution. For any instance  $g$  of  $\text{RT}_2^n$  computable in  $f$ , by Corollary 2.10,  $g$  admits a  $\Pi_n^0(f)$  solution  $H$ , so in particular  $\Pi_n^0$ . Everything that  $H$  computes is  $\Delta_{n+1}^0$  and therefore also  $\Sigma_{n+1}^0$ . So this solution  $H$  to  $g$  does not compute (and therefore does  $f$ -compute) any solution to  $f$ . ■

From the point of view of Reverse Mathematics, Ramsey's theorem for singletons is provable in  $\text{RCA}_0$  and Ramsey's theorem for  $n$ -tuples is equivalent to  $\text{ACA}_0$  when  $n \geq 3$ . The case of Ramsey's theorem for pairs is more intriguing. By Theorem 2.17,  $\text{RT}_2^2$  is provable in  $\text{ACA}_0$  and by Corollary 2.7, it is not provable in  $\text{WKL}_0$ . What about the reciprocals? One way to prove that Ramsey's theorem for pairs implies  $\text{ACA}_0$  would be to adapt the proof of Theorem 2.16 to build a computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$ , all homogeneous sets of which compute the halting problem, and to check that the argument is formalized in  $\text{RCA}_0$ . David Seetapun [194] succeeds in showing

in 1995 that such a thing is impossible. In 2012, Lu Liu [145] showed that it is also impossible to build a computable instance of  $\text{RT}_2^2$  whose solutions are all of PA degree. These two major results require a good understanding of the infinite pigeonhole principle.

### 3. Infinite pigeonhole principle

The notion of pre-homogeneous set  $P$  allows to reduce a computable instance of  $\text{RT}_k^{n+1}$  into a  $P$ -computable instance of  $\text{RT}_k^n$ . In particular, Ramsey's theorem for pairs is reduced to a non-computable instance of the infinite pigeonhole principle. It is therefore natural to study the non-computable instances of  $\text{RT}_k^1$ .

We therefore pause in the computational analysis of Ramsey's theorem to focus on its base case: the infinite pigeonhole principle. We will see that this principle, although combinatorially elementary, presents all the same some subtleties, and that the analysis of arbitrary instances of  $\text{RT}_k^1$  is far from trivial.

#### 3.1. Number of colors and uniformity

We have seen with Proposition 2.3 and Proposition 2.4 that  $\text{RCA}_0 \vdash \text{RT}_k^1$  for any standard integer  $k$ . In particular that  $\text{RT}_k^1$  is computably true, in the sense that any instance computes its own solution.

On the other hand, the restricted induction of  $\text{RCA}_0$  is not sufficient to deduce  $\text{RCA}_0 \vdash \forall k \text{RT}_k^1$ . We show it formally with the following proposition, via the equivalence over  $\text{RCA}_0$  of  $\forall k \text{RT}_k^1$  and the collection principle for  $\Pi_1^0$  formulas, which as we have seen is not provable over  $\text{RCA}_0$ :

**Proposition 3.1 (Hirst [90]).**  $\text{RCA}_0 \vdash \forall k \text{RT}_k^1 \leftrightarrow \text{B}\Pi_1^0$ . ★

PROOF.  $\text{B}\Pi_1^0 \rightarrow \forall k \text{RT}_k^1$ : Let  $f : \mathbb{N} \rightarrow k$  be an instance of  $\forall k \text{RT}_k^1$ . Let  $F(x, y)$  be the  $\Pi_1^0$  formula  $\forall z > y \ f(z) \neq x$ . If there exists  $x < k$  such that for all  $y$ ,  $F(x, y)$  is false, then the set  $\{z \in \mathbb{N} : f(z) = x\}$  is infinite, and exists by the  $\Delta_0^0$  comprehension scheme. Otherwise, for all  $x < k$ , there exists  $y$  such that  $F(x, y)$  is true. By  $\text{B}\Pi_1^0$ , there exists  $p \in \mathbb{N}$  such that for all  $x < k$ , there exists  $y < p$  such that  $F(x, y)$  is true. Let  $x$  be the color of  $f(p+1)$ . So  $p+1$  contradicts  $F(x, p)$ .

$\forall k \text{RT}_k^1 \rightarrow \text{B}\Pi_1^0$ : Let  $F(x, y) = \forall z G(x, y, z)$  be a  $\Pi_1^0$  formula and  $k \in \mathbb{N}$  such that  $\forall x < k \ \exists y \ F(x, y)$ . Let  $f : \mathbb{N} \rightarrow k$  be defined by  $f(t) = n$  for the smallest integer  $n$  such that  $\forall x < k \ \exists y < n \ \forall z < t \ G(x, y, z)$ , if such  $n$  exists. Otherwise  $f(t) = t$ . If there exists an infinite set  $H$  homogeneous for  $f$  for some color  $\ell \in \mathbb{N}$ , then  $\forall x < k \ \exists y < \ell \ \forall z \ G(x, y, z)$ .

If there is no infinite set  $H$  homogeneous for  $f$ , then by  $\forall k\text{RT}_k^1$ , the image of  $f$  is unbounded. By  $\Delta_1^0$  comprehension, it is possible to construct a strictly increasing sequence  $(t_i)_{i \in \mathbb{N}}$  such that  $f(t_i) < f(t_{i+1})$  for all  $i \in \mathbb{N}$ . Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be defined as follows:  $g(i)$  is the smallest  $x < k$  such that  $\forall y < f(t_i) - 1 \exists z < t_i \neg G(x, y, z)$ . Let  $S$  be an infinite homogeneous set for  $g$ , for color  $x_0$ . Let  $y \in \mathbb{N}$ . Since  $S$  is infinite, there exists  $i \in S$  such that  $f(t_i) - 1 > y$ , so  $\exists z < t_i \neg G(x_0, y, z)$ . Thus,  $\forall y \exists z \neg G(x_0, y, z)$ , contradiction. ■

Note that the proof of  $\text{RCA}_0 \vdash \forall k\text{RT}_k^1 \rightarrow \text{BII}_1^0$  involves two instances of  $\forall k\text{RT}_k^1$ .

We now see that Ramsey's theorem for pairs is sufficient to show  $\forall k\text{RT}_k^1$  over  $\text{RCA}_0$ . We are here in a typical case where a statement which relates to the second order has consequences to the first order. This is because the solutions of an instance of  $\text{RT}_2^2$  can be used as parameters in the  $\Sigma_1^0$  induction scheme.

**Proposition 3.2.** For all  $n \geq 1$ ,  $\text{RCA}_0 \vdash \text{RT}_2^{n+1} \rightarrow \forall k\text{RT}_k^n$ . In particular,  $\text{RCA}_0 \vdash \text{RT}_2^2 \rightarrow \text{BII}_1^0$ . ★

PROOF. Let  $f : [\mathbb{N}]^n \rightarrow k$  be an instance of  $\text{RT}_k^n$  for a  $k \in \mathbb{N}$ . Let  $g : [\mathbb{N}]^{n+1} \rightarrow 2$  be defined for all  $F \in [\mathbb{N}]^{n+1}$  by  $g(F) = 1$  iff  $F$  is homogeneous for  $f$ . Let  $H$  be an infinite homogeneous set for  $g$ .

Let us show that  $G$  is homogeneous for  $g$  for color 1. By Finite Ramsey's theorem (provable in  $\text{RCA}_0$ , see Theorem 1.10 in Hájek and Pudlák [80]), there exists a homogeneous subset  $A \in [H]^{n+1}$  for  $f$ . So  $g(A) = 1$ .

Let us show that  $H$  is homogeneous for  $f$ . Let  $A, B \in [H]^n$ . Then, there exists a finite sequence of sets  $A_0, \dots, A_\ell \in [H]^{n+1}$  such that  $A \subseteq A_0, B \subseteq A_\ell$  and for all  $i < \ell - 1$ ,  $|A_i \cap A_{i+1}| \geq n$ . As  $A_0, \dots, A_\ell \in [H]^{n+1}$ , then they are all  $f$ -homogeneous, and as  $|A_i \cap A_{i+1}| \geq n$ , then  $A_i$  and  $A_{i+1}$  are  $f$ -homogeneous for the same color. It follows that  $f(A) = f(B)$ . For  $n = 1$ ,  $\text{RCA}_0 \vdash \text{RT}_2^2 \rightarrow \forall k\text{RT}_k^1$ , so by 3.1,  $\text{RCA}_0 \vdash \text{RT}_2^2 \rightarrow \text{BII}_1^0$ . ■

The question of whether  $\text{RCA}_0 + \text{RT}_2^2$  allows us to prove other first-order statements than  $\text{BII}_1^0$  is still open:

**Question 3.3.** Is  $\text{RT}_2^2$  a conservative extension of  $\text{RCA}_0 + \text{BII}_1^0$  for arithmetic formulas? ★

Progress was made in this direction by Patey and Yokoyama, who showed that  $\text{RCA}_0 + \text{RT}_2^2$  could not show any  $\Pi_3^0$  statement that were not already provable in  $\text{RCA}_0$ .

**Theorem 3.4 (Patey, Yokoyama [172])**

$\text{RT}_2^2$  is a conservative extension of  $\text{RCA}_0$  for  $\Pi_3^0$  formulas.

The reader can check that there is no contradiction with the fact that  $\text{RCA}_0 + \text{RT}_2^2$  implies  $\text{B}\Pi_1^0$ .

**Exercise 3.5. (★)** Show that an instance of  $\text{B}\Pi_1^0$  is a  $\Sigma_4^0$  statement, and provably equivalent to a  $\Sigma_3^0$  statement over  $\text{RCA}_0$ .  $\diamond$

Now, let's refocus on the  $\omega$ -models. The proof of  $\text{RCA}_0 \vdash \text{RT}_2^1$  proceeds by case analysis on the colors: either the color 0 appears infinitely often, or not, in which case the color 1 appears infinitely often. In particular, the choice of color is not uniform in a computable way, because it results from the answer to a  $\Pi_2^0$  question. The Weihrauch reduction makes it possible to reflect this non-uniformity. Brattka and Rakotoniaina [25] and Hirschfeldt and Jockusch [87] independently proved the following proposition.

**Proposition 3.6 ([25],[87]).** For all  $k \geq 1$ ,  $\text{RT}_{k+1}^1 \not\leq_W \text{RT}_k^1$ .  $\star$

PROOF. Suppose that  $\text{RT}_{k+1}^1 \leq_W \text{RT}_k^1$  via the functionals  $\Phi$  and  $\Psi$ . Let  $P(n)$  be the predicate “there exists a sequence  $\sigma_0 \prec \dots \prec \sigma_n \in (k+1)^{<\mathbb{N}}$  of initial segments of the instance of  $\text{RT}_{k+1}^1$ , a sequence of non-empty finite sets  $F_0, \dots, F_{n-1}$  of partial solutions for the partial instances  $\Phi(\sigma_0), \dots, \Phi(\sigma_{n-1})$  of  $\text{RT}_k^1$ , of respective pairwise distinct colors  $i_0, \dots, i_{n-1} \in \{0, \dots, k-1\}$ , such that for all  $s < n$ , there is  $x$  for which  $\sigma_{s+1}(x) = s$  and  $\Psi(\sigma_{s+1} \oplus F_s, x) \downarrow = 1$ ”. We have  $P(0)$  with the string  $\sigma_0 = \epsilon$ . Let us show that we have  $P(n) \rightarrow P(n+1)$  for all  $n \leq k$ , which implies a contradiction for  $P(k+1)$  because we then have a sequence of pairwise different colors  $i_0, \dots, i_k \in \{0, \dots, k-1\}$ .

Suppose that  $P(n)$  is verified with  $\sigma_0, \dots, \sigma_n, F_0, \dots, F_{n-1}$  and  $i_0, \dots, i_{n-1}$ . Let  $H$  be a solution to the instance  $\Phi(\sigma_n \smallfrown n^\infty)$  of  $\text{RT}_k^1$  and  $i < k$  be its color. We can assume that  $\min H > |\sigma_n|$ .

If  $i = i_s$  for an  $s < n$ , then  $F_s \cup H$  is still homogeneous. In particular  $\Psi((\sigma_n \smallfrown n^\infty) \oplus (F_s \cup H))$  is infinite, so must be homogeneous for  $\sigma_n \smallfrown n^\infty$  for color  $n$ . However, by induction hypothesis, there are  $x$  such that  $\sigma_{s+1}(x) = s$  and  $\Psi(\sigma_{s+1} \oplus F_s, x) \downarrow = 1$ , so  $\Psi((\sigma_n \smallfrown n^\infty) \oplus (F_s \cup H))$  contains elements of color  $n$  and  $s$ , contradiction. So  $i$  is different from  $i_s$  for all  $s < n$ .

Let  $i_n = i$ . Let  $F_n$  be a homogeneous non-empty finite set for  $\Phi(\sigma_n \smallfrown n^\infty)$  for color  $i$  and  $x \geq |\sigma_n|$  such that  $\Psi((\sigma_n \smallfrown n^\infty) \oplus F_n, x) \downarrow = 1$ , and let  $\sigma_{n+1}$  be an initial segment of  $\sigma_n \smallfrown n^\infty$  sufficiently long so that  $F_n$  is a partial solution to  $\Phi(\sigma_{n+1})$  and so that  $\Psi(\sigma_{n+1} \oplus F_n, x) \downarrow = 1$ . We have  $P(n+1)$ . ■

**Exercise 3.7.** Check that the proof of Proposition 3.2 is also a proof of  $\text{RT}_k^n \leq_W \text{RT}_2^{n+1}$  for all  $n, k \geq 1$ .  $\diamond$

### 3.2. Infinite pigeonhole principle and cone avoidance

We now deal with the non-computable instances of  $\text{RT}_2^1$ . The aim of this section is to prove the following theorem.

**Theorem 3.8 (Dzhafarov and Jockusch [53])**

*For any set  $A$  and any non-computable set  $C$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $C \not\leq_T H$ .*

We will see that this theorem constitutes a key step to show that  $\text{RT}_2^2$  preserves the weakness property given by  $\{X : X \not\leq_T C\}$  for any non-computable  $C$  (see Definition 24-1.6). Via the expansions of Section 24-1.1, this will allow us in particular to construct an  $\omega$ -model of  $\text{RCA}_0 + \text{RT}_2^2$  not containing  $\emptyset'$  and thus to establish Theorem 4.14 of Seetapun to come, which separates  $\text{RT}_2^2$  from  $\text{ACA}_0$ . Note that we do not follow Seetapun's original proof, which is more direct, but also less informative, notably on the combinatorial weakness of arbitrary instances of  $\text{RT}_2^1$ .

We invite the reader to reflect on the meaning of the above theorem, starting perhaps by putting it in parallel with something that we have already used in the proof of Lemma 14-4.6, and that we give here as an exercise:

**Exercise 3.9. (★)** Let  $C$  be any set. Show that there exists an infinite set  $A$ , all the infinite subsets of which allow to compute  $C$ .  $\diamond$

The proof of Theorem 3.8 is done via a forcing notion defined by Dzhafarov and Jockusch, which consists of an effective variant of *Mathias forcing*, used initially in Set Theory to construct ZFC models in which, for any class  $\mathcal{C} \subseteq 2^{\mathbb{N}}$ , there exists an infinite set, all infinite subsets of which are in  $\mathcal{C}$  or in  $2^{\mathbb{N}} \setminus \mathcal{C}$  ([150] and [151], see [105]). Let's start by defining some notations that will simplify the presentation.

#### Notation

We will consider the strings  $\sigma \in 2^{<\mathbb{N}}$  as finite sets of integers via the following notations:

1. The string  $\sigma \cup \tau$  is the string of size  $\max(|\sigma|, |\tau|)$  such that  $\sigma \cup \tau(n) = 1$  iff  $\sigma(n) = 1$  or  $\tau(n) = 1$ .
2. The string  $\tau - \sigma$  is the string  $\tau$ , from which we subtract the 1 from  $\sigma$ . So we have  $(\tau - \sigma)(n) = 1$  iff  $\tau(n) = 1$  and  $\sigma(n) = 0$  for  $n < \min(|\sigma|, |\tau|)$  and  $(\tau - \sigma)(n) = 1$  iff  $\tau(n) = 1$  for  $n$  such

that  $|\tau| > n \geq \min(|\sigma|, |\tau|)$

3. The set  $X - \sigma$  is the set  $X$  from which we subtract the integers  $n < |\sigma|$  such that  $\sigma(n) = 1$ .
4. The predicate  $\sigma \subseteq X$  means  $\sigma(n) = 1$  implies  $X(n) = 1$ .

Mathias forcing, as used in combinatorics, is defined as follows:

**Definition 3.10.** A *Mathias condition* is a pair  $(\sigma, X)$  such that

1.  $\sigma \in 2^{<\mathbb{N}}$
2.  $X$  is an infinite subset of  $\mathbb{N}$
3.  $X \cap \{0, \dots, |\sigma|\} = \emptyset$

We will say, for Mathias conditions  $(\tau, Y)$  and  $(\sigma, X)$ , that  $(\tau, Y)$  *extend*  $(\sigma, X)$ , and we will write  $(\tau, Y) \leq (\sigma, X)$  if  $\tau \supseteq \sigma$ ,  $Y \subseteq X$  and  $\tau - \sigma \subseteq X$ . The partial order of Mathias conditions can be enriched with a Cantor forcing by interpreting any maximal filter  $F$  by  $\dot{F}$  as being the unique element of  $\bigcap_{(\sigma, X) \in F} [\sigma]$ .  $\diamond$

Given a Mathias condition  $(\sigma, X)$ , we denote by  $[\sigma, X]_\infty$  the set of maximal filters containing  $(\sigma, X)$ , and  $[\sigma, X] = \{\dot{F} : F \in [\sigma, X]_\infty\}$ . We then have

$$[\sigma, X] = \{G \in \mathcal{P}(\mathbb{N}) : \sigma \prec G \text{ and } G - \sigma \subseteq X\}$$

Thus, a Mathias condition  $(\sigma, X)$  can be considered as a refinement of Cohen forcing. Indeed, the finite part  $\sigma$  is an initial segment of the constructed element, which makes it possible to fix  $\Sigma_1^0$  properties. We add an infinite reservoir  $X$  of the values that we authorize to add to the initial segment. The reservoir only contains negative information, in the sense that if  $G \in [\sigma, X]$  and  $n \in X$ , then it is not guaranteed that  $n \in G$ . On the other hand, if  $n \notin X$ , then  $n \notin G$ .

**Definition 3.11.** Let  $(\sigma, X)$  be a Mathias condition and  $\Phi_e$  a Turing functional.

1.  $(\sigma, X) \Vdash \Phi_e(G, n) \downarrow = m$  if  $\Phi_e(\sigma, n) \downarrow = m$
2.  $(\sigma, X) \Vdash \Phi_e(G, n) \uparrow$  if  $\forall \tau \subseteq X \ \Phi_e(\sigma \cup \tau, n) \uparrow$

$\diamond$

We are now ready to define the notion of Dzhafarov-Jockusch forcing, which presents a certain additional conceptual difficulty compared to a classic forcing: given some non-computable sets  $A$  and  $C$ , we will not know in

advance if the generic set  $G \not\leq_T C$  that we want to build will be an infinite subset of  $A$  or  $\mathbb{N} \setminus A$ . We will therefore build in parallel two generic sets  $G^0, G^1$ , one of them being included in  $A$  and the other in  $\mathbb{N} \setminus A$ .

**Definition 3.12.** Let  $A^0 \sqcup A^1$  be a 2-partition of  $\mathbb{N}$  and  $\mathcal{I}$  be a Scott ideal. A *Dzhafarov-Jockusch condition* for  $A^0 \sqcup A^1$  and  $\mathcal{I}$  is a triplet  $(\sigma_0, \sigma_1, X)$  such that

- (1)  $(\sigma_i, X)$  is a Mathias condition for all  $i < 2$
- (2)  $\sigma_i \subseteq A^i$  for all  $i < 2$
- (3)  $X \in \mathcal{I}$

A condition  $(\tau_0, \tau_1, Y)$  *extends*  $(\sigma_0, \sigma_1, X)$  if  $(\tau_i, Y)$  extends  $(\sigma_i, X)$  in Mathias' sense for each  $i < 2$ . ◇

A Dzhafarov-Jockusch condition will therefore depend on a partition  $A^0 \sqcup A^1$  and on a Scott ideal  $\mathcal{I}$ .

### Notation

In the remainder of the proof, depending on the context, we will use  $A, \mathbb{N} \setminus A$  or  $A^0, A^1$  where  $A^0$  is a notation for  $A$  and  $A^1$  a notation for  $\mathbb{N} \setminus A$ .

To build a generic set  $G \not\leq_T C$  for a certain fixed non-computable set  $C$ , we will use a Scott ideal not containing  $C$ .

A maximal filter  $F$  can be interpreted in two sets  $\dot{F} = (G^0, G^1)$  defined for all  $i < 2$  by  $\{G^i\} = \bigcap_{(\sigma_0, \sigma_1, X) \in F} [\sigma_i]$ . For a condition  $(\sigma_0, \sigma_1, X)$ , we define

$$[\sigma_0, \sigma_1, X] = \{(G^0, G^1) : \forall i < 2 \ \sigma_i \preceq G^i \wedge G^i - \sigma_i \subseteq X\}$$

Thus, for any maximal filter  $F$ , we have

$$\{\dot{F}\} = \bigcap_{(\sigma_0, \sigma_1, X) \in F} [\sigma_0, \sigma_1, X]$$

### Forcing relation

Note a subtlety here: for a Cantor forcing notion  $(\mathbb{P}, \leq)$ , and  $c \in \mathbb{P}$ , we had defined  $[c]_\in$  as the class of maximum filters containing  $c$ , and  $[c] = \{\dot{F} : F \in [c]\}$ . The definition of  $[c]$  for Dzhafarov-Jockusch forcing is different. Indeed, for any maximal filter  $F$ ,  $\dot{F} \in [\sigma_0, \sigma_1, X]$  but the reverse is not true, and there exist pairs  $(G^0, G^1) \in [\sigma_0, \sigma_1, X]$  such that  $G^i \not\subseteq A^i$ , while it is necessarily the case for  $\dot{F}$  by (2) of the definition of a condition.

We will also impose the following hypothesis that will have to be justified for any application of the Dzhafarov-Jockusch forcing:

For every  $i < 2$ , there is no infinite set  $H \subseteq A^i$  in  $\mathcal{I}$ . (H1) ★

The idea is that the elements of  $\mathcal{I}$  will satisfy the property that we want to show, typically not to compute a fixed set  $C$ . It follows that if an infinite set  $H \subseteq A^i$  is in  $\mathcal{I}$ , then  $H$  is the desired solution and there is no need to do any further construction. The assumption (H1) allows to prove that the two sets produced by a sufficiently generic filter for the Dzhafarov-Jockusch forcing are infinite, as shows the following lemma.

**Lemma 3.13.** Suppose (H1). Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $i < 2$ . There is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and an integer  $n > |\sigma_i|$  such that  $n \in \tau_i$ . ★

PROOF. If  $X \cap A^i$  is empty, then  $X \subseteq A^{1-i}$ , but  $X \in \mathcal{I}$ , which contradicts (H1). Thus, there is  $n \in X \cap A^i$ . Let  $\tau_i = \sigma_i \cup \{n\}$ , and  $\tau_{1-i} = \sigma_{1-i}$ . Then,  $(\tau_0, \tau_1, X \setminus \{0, \dots, n-1\})$  is an extension of  $(\sigma_0, \sigma_1, X)$  such that  $n \in \tau_i$ . ■

We saw in Section 11-4 that the computational properties of a set produced by a sufficiently generic filter were closely related to the existence of a *forcing question* with good definitional properties. Recall that a forcing question for a Cantor forcing  $\mathbb{P}$  is a relation  $c ?\vdash \mathcal{R}$  where  $c \in \mathbb{P}$  and  $\mathcal{R}$  is a requirement, such that if  $c ?\vdash \mathcal{R}$  is true, then there exists an extension  $d \leq c$  for which  $d \Vdash \mathcal{R}$ , and otherwise, there is an extension  $d \leq c$  for which  $d \Vdash \neg \mathcal{R}$ . The situation of Dzhafarov-Jockusch forcing is a little more complex, because two distinct sets are created in parallel. It would be natural to define a forcing question for each set constructed:

$$(\sigma_0, \sigma_1, X) ?\vdash^i \exists n \Phi_e(G, n) \downarrow$$

if there is a string  $\rho \subseteq X \cap A^i$  and an integer  $n \in \mathbb{N}$  such that  $\Phi_e(\sigma_i \cup \rho, n) \downarrow$ . Thus, if  $(\sigma_0, \sigma_1, X) ?\vdash^i \exists n \Phi_e(G, n) \downarrow$ , there exists an extension  $(\tau_0, \tau_1, Y)$  such that  $(\tau_i, Y \cap A^i) \Vdash \exists n \Phi_e(G, n) \downarrow$ , and if not, there is an extension  $(\tau_0, \tau_1, Y)$  such that  $(\tau_i, Y \cap A^i) \Vdash \forall n \Phi_e(G, n) \uparrow$ . However, this definition of forcing question does not satisfy the good definitional properties, because it is  $\Sigma_1^0(X \oplus A)$ , but  $A$  can be arbitrarily complex.

We will therefore define a variant of the forcing question for the  $\Sigma_1^0$  and  $\Pi_1^0$  requirements which is freed from  $A$ , at the cost of a disjunction.

**Definition 3.14.** Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ .

$$c ?\vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$$

if for any 2-partition  $Z^0 \sqcup Z^1 = \mathbb{N}$ , there exists  $i < 2$ , a string  $\rho \subseteq Z^i \cap X$  and an integer  $n \in \mathbb{N}$  such that  $\Phi_{e_i}(\sigma_i \cup \rho, n) \downarrow$ .  $\diamond$

At first glance, we abstracted ourselves from the set  $A$  and replaced it with an otherwise more complicated formula that universally quantifies over 2-partitions. However, by a compactness argument, the relation of Definition 3.14 magically simplifies into a  $\Sigma_1^0(X)$  formula.

**Lemma 3.15.** Let  $(\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ . The relation

$$(\sigma_0, \sigma_1, X) \vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$$

is  $\Sigma_1^0(X)$  uniformly in  $e_0$  and  $e_1$ .  $\star$

PROOF. Let  $\mathcal{C}$  be the class of all xsets  $Z^0 \oplus Z^1$  such that  $Z^0 \sqcup Z^1 = \mathbb{N}$ , and such that for all  $i < 2$ , all  $\rho \subseteq Z^i \cap X$  and all  $n \in \mathbb{N}$ ,  $\Phi_{e_i}(\sigma_i \cup \rho, n) \uparrow$ . Then,  $(\sigma_0, \sigma_1, X) \vdash \exists n \Phi_{e_0}(G, n) \downarrow \vee \exists n \Phi_{e_1}(G, n) \downarrow$  iff  $\mathcal{C} = \emptyset$ . Since  $\mathcal{C}$  is a  $\Pi_1^0(X)$  class, the relation  $\mathcal{C} = \emptyset$  is  $\Sigma_1^0(X)$ .  $\blacksquare$

Definition 3.14 can be made explicitly  $\Sigma_1^0(X)$  as follows:

**Definition 3.16 (Alternative definition).** Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ .

$$c \vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$$

if there is  $r \in \mathbb{N}$  such that for any 2-partition  $Z^0 \sqcup Z^1 = \{0, \dots, r-1\}$ , there is  $i < 2$ , a string  $\rho \subseteq Z^i \cap X$  and an integer  $n \in \mathbb{N}$  such that  $\Phi_{e_i}(\sigma_i \cup \rho, n) \downarrow$ .  $\diamond$

Let us dwell on the meaning of Definition 3.14. This definition avoids mentioning the set  $A$  by doing an “over-approximation”. The forcing question asks if for *all* instances of  $\text{RT}_2^1$ , it is possible to find a finite extension which forces on one side the  $\Sigma_1^0$  requirement. So it is clear that if the answer is yes, then in particular for  $Z^i = A^i$  there is an extension  $(\tau_0, \tau_1, Y)$  such that  $(\tau_i, Y) \vdash \exists n \Phi_{e_i}(G, n)$  for some  $i < 2$ . On the other hand, if the answer to the forcing question is no, then this failure may be due to an instance  $Z^0 \sqcup Z^1 = \mathbb{N}$  of  $\text{RT}_2^1$  which has nothing to do with the original instance  $A^0 \sqcup A^1 = \mathbb{N}$ . This is where the combinatorial nature of Ramsey's theory comes into play: it suffices to construct a set which is both a solution to the instance  $Z^0 \sqcup Z^1 = \mathbb{N}$  and of the instance  $A^0 \sqcup A^1 = \mathbb{N}$  to force the property.

**Proposition 3.17.** Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ .

(1) If  $c \vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$ , then there is an integer  $i < 2$

and an extension  $(\tau_0, \tau_1, Y)$  such that  $(\tau_i, Y) \Vdash \exists n \Phi_{e_i}(G, n) \downarrow$ .

- (2) If  $c \not\Vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$ , then there is an integer  $i < 2$  and an extension  $(\tau_0, \tau_1, Y)$  such that  $(\tau_i, Y) \Vdash \forall n \Phi_{e_i}(G, n) \uparrow$ . ★

PROOF. (1) If  $c \not\Vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$ , then in particular, for  $Z^i = A^i$ , there is  $i < 2$ , a string  $\rho \subseteq A^i \cap X$  and an integer  $n \in \mathbb{N}$  such that  $\Phi_{e_i}(\sigma_i \cup \rho, n) \downarrow$ . Suppose  $i = 0$ , the other case being symmetrical. The condition  $(\sigma_0 \cup \rho, \sigma_1, X \setminus \{0, \dots, |\rho|\})$  is an extension of  $c$  such that  $(\sigma_0 \cup \rho, X \setminus \{0, \dots, |\rho|\}) \Vdash \Phi_{e_0}(G, n) \downarrow$ .

(2) If  $c \not\Vdash \exists n \Phi_{e_0}(G^0, n) \downarrow \vee \exists n \Phi_{e_1}(G^1, n) \downarrow$ . Let  $\mathcal{C}$  be the class of all sets  $Z^0 \oplus Z^1$  such that  $Z^0 \sqcup Z^1 = \mathbb{N}$ , and such that for all  $i < 2$ , all  $\rho \subseteq Z^i \cap X$  and all  $n \in \mathbb{N}$ ,  $\Phi_{e_i}(\sigma_i \cup \rho, n) \uparrow$ . By hypothesis,  $\mathcal{C} \neq \emptyset$ . Moreover,  $\mathcal{C}$  is a  $\Pi_1^0(X)$  class with  $X \in \mathcal{I}$ , so since  $\mathcal{I}$  is a Scott ideal, there is  $Z^0 \oplus Z^1 \in \mathcal{I} \cap \mathcal{C}$ . Let  $i < 2$  be such that  $X \cap Z^i$  is infinite. Then, the condition  $(\sigma_0, \sigma_1, X \cap Z^i)$  is an extension of  $c$  such that  $(\sigma_i, X \cap Z^i) \Vdash \forall n \Phi_{e_i}(G, n) \uparrow$ . Note that  $X \in \mathcal{I}$  and  $Z^i \in \mathcal{I}$ , hence  $X \cap Z^i \in \mathcal{I}$ . ■

Note that in each case, it would have been sufficient to ensure that  $(\tau_i, Y \cap A^i)$  forces the property, but this is also the case for  $(\tau_i, Y)$  which is a stronger result.

We now have all the elements necessary to prove the theorem of Dzhafarov and Jockusch which we recall here.

**Theorem (3.8)**

*For any set  $A$  and any non-computable set  $C$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $C \not\leq_T H$ .*

PROOF. Let  $A$  and  $C$  be fixed. By Proposition 22-7.3, there exists a Scott ideal  $\mathcal{I}$  such that  $C \notin \mathcal{I}$ . Suppose (H1), otherwise we already have the desired solution. We are going to construct two infinite sets  $G^0 \subseteq A^0$ ,  $G^1 \subseteq A^1$  satisfying for all  $e_0, e_1 \in \mathbb{N}$  the requirement

$$\mathcal{R}_{e_0, e_1} : \Phi_{e_0}^{G^0} \neq C \vee \Phi_{e_1}^{G^1} \neq C$$

The sets  $G^0$  and  $G^1$  will be built by Dzhafarov-Jockusch forcing with the ideal  $\mathcal{I}$ .

**Lemma 3.19.** Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ . There is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  forcing  $\mathcal{R}_{e_0, e_1}$ . ★

PROOF. Let  $W = \{(x, v) \in \mathbb{N} \times \{0, 1\} : c \not\Vdash \Phi_{e_0}^{G^0}(x) \downarrow = v \vee \Phi_{e_1}^{G^1}(x) \downarrow = v\}$ . By Lemma 3.15,  $W$  is  $\Sigma_1^0(X)$  with  $X \in \mathcal{I}$ . Three cases arise:

- Case 1:  $(x, 1 - C(x)) \in W$  for some  $x \in \mathbb{N}$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and an integer  $i < 2$  such that  $(\tau_i, Y) \Vdash \Phi_{e_i}^G(x) \downarrow = 1 - C(x)$ , so  $(\tau_i, Y) \Vdash \Phi_{e_i}^G \neq C$ .
- Case 2:  $(x, C(x)) \notin W$  for some  $x \in \mathbb{N}$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and an integer  $i < 2$  such that  $(\tau_i, Y) \Vdash \Phi_{e_i}^G(x) \uparrow \vee \Phi_{e_i}^G(x) \downarrow \neq C(x)$ , so  $(\tau_i, Y) \Vdash \Phi_{e_i}^G \neq C$ .
- Case 3: For all  $(x, v) \in W$ ,  $C(x) = v$ , and for all  $x \in \mathbb{N}$ , there is a  $v < 2$  such that  $(x, v) \in W$ . Then,  $C \leq_T X$ , because for  $x \in \mathbb{N}$ , it suffices to wait for a couple  $(x, v)$  to be  $X$ -enumerated in  $W$ , and return  $v$ . By hypothesis on  $X$ , this case cannot happen. ■

Let  $F$  be a sufficiently generic filter for the Dzhafarov-Jockusch forcing, and let  $(G^0, G^1) = \dot{F}$ . By Lemma 3.13,  $G^0$  and  $G^1$  are both infinite. By definition of a forcing condition,  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . By Lemma 3.19,  $(G^0, G^1)$  satisfy  $\mathcal{R}_{e_0, e_1}$  for all  $e_0, e_1 \in \mathbb{N}$ . There are then two possibilities, either  $C \not\leq_T G^0$ , or there exists  $e_0$  such that  $\Phi_{e_0}^{G^0} = C$ , in which case for all  $e_1$  we have  $\Phi_{e_1}^{G^1} \neq C$  because  $\mathcal{R}_{e_0, e_1}$  is satisfied. In this case  $C \not\leq_T G^1$ . ■

**Exercize 3.20. (★★)** Show that for any set  $A$  and any non- $\Sigma_1^0$  set  $C$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \bar{A}$  such that  $C$  is not  $\Sigma_1^0(H)$ . ◇

**Exercize 3.21. (★★★)** Show that for every set  $A$  and every hyperimmune function  $f$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \bar{A}$  such that  $f$  is  $H$ -hyperimmune. ◇

**Exercize 3.22. (★★)** Adapt the previous exercise to show that for any set  $A$  and every triplet of hyperimmune functions  $f_0, f_1, f_2$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \bar{A}$  such that at least two among the three functions are  $H$ -hyperimmunes. ◇

Theorem 3.8 has been generalized by Monin and Patey [160] to the whole arithmetic hierarchy.

**Theorem 3.23 (Monin and Patey [160])**

Let  $n \geq 1$ . For any set  $A$  and any non- $\Delta_n^0$  set  $C$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $C$  is not  $\Delta_n^0(H)$ .

A consequence of the previous theorem is that for any set  $A$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  which is not of high degree. The proof is done with the help of a more complex forcing notion which goes beyond the introductory nature of this book.

### 3.3. Infinite pigeonhole principle and PA degrees

The theorem of Dzhafarov and Jockusch will be used to prove Seetapun's theorem:  $\text{RT}_2^2$  does not imply  $\text{ACA}_0$ . We now tackle Liu's theorem, which strengthens Seetapun's theorem by showing that  $\text{RT}_2^2$  does not imply  $\text{WKL}_0$ . Here again, the heart of the proof lies in the arbitrary instances of  $\text{RT}_2^1$ .

**Theorem 3.24 (Liu [145])**

*For any set  $A$ , there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  which is not of PA degree.*

The proof that we present here is that of Monin and Patey [158], shorter than the original proof of Liu, thanks in particular to the use of the concept of *large class*:

**Definition 3.25.** A non-empty class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is *large* if:

1. It is *upward-closed*:  $\forall X \in \mathcal{A} \forall Y \supseteq X \ Y \in \mathcal{A}$ .
2. For all  $k \in \mathbb{N}^*$ , for all  $X_0 \sqcup \dots \sqcup X_{k-1} = \mathbb{N}$  there exists  $i < k$  such that  $X_i \in \mathcal{A}$ .  $\diamond$

For the rest of this section, we fix a set  $A$  with  $A^0 = A$  and  $A^1 = \mathbb{N} \setminus A$ . We also fix an enumeration  $(\mathcal{U}_e)_{e \in \mathbb{N}}$  of all the upward-closed  $\Sigma_1^0$  classes.

**Definition 3.26.** Let  $\mathbb{P}$  be the set of forcing conditions of the form  $\langle (\sigma_s, X_s)_{s < k}, C \rangle$  for  $k \in \mathbb{N}^*$  such that:

- (a) For any  $s < k$ , there exists  $i < 2$  such that  $\sigma_s \cup X_s \subseteq A^i$ .
- (b)  $X_0 \sqcup \dots \sqcup X_{k-1} = \mathbb{N} - \{0, \dots, \max_s |\sigma_s|\}$ .
- (c)  $\bigcap_{e \in C} \mathcal{U}_e$  is a large class containing only infinite sets.
- (d)  $C$  is computable.  $\diamond$

Note that we have no restriction of effectiveness on reservoirs  $X_0 \sqcup \dots \sqcup X_{k-1}$ .

**Definition 3.27.** The partial order on  $\mathbb{P}$  is defined by  $\langle (\tau_s, Y_s)_{s < \ell}, D \rangle \leq \langle (\sigma_s, X_s)_{s < k}, C \rangle$  if  $C \subseteq D$ , if  $\ell \geq k$ , and if there is a surjection  $f : \ell \rightarrow k$  such that for all  $s < \ell$ , we have  $\sigma_{f(s)} \preceq \tau_s$ ,  $Y_s \subseteq X_{f(s)}$ , and  $\tau_s - \sigma_{f(s)} \subseteq X_{f(s)}$ .  $\diamond$

A condition  $c = \langle (\sigma_s, X_s)_{s < k}, C \rangle$  must be seen as a set of branches  $(\sigma_s, X_s)$ , each being a Mathias condition, except that we do not require that  $X_s$  be

infinite. An extension  $d \leq c$  is a condition where each branch will be an extension of one of the branches of  $c$ .

**Definition 3.28.** Let  $\langle(\sigma_s, X_s)_{s < k}, C\rangle \in \mathbb{P}$ . We call *branch*  $c^{[s]}$  of  $c$  the Mathias condition  $(\sigma_s, X_s)$ . Such a branch is *valid* if  $X_s \in \bigcap_{e \in C} \mathcal{U}_e$ .  $\diamond$

The notion of validity is crucial: the branches of our conditions  $c_0 \geq c_1 \geq \dots$  will form a tree, and the generic set that we will use will come from an infinite path of valid branches in our tree.

### Notation

For two conditions  $c = \langle(\tau_s, Y_s)_{s < \ell}, D\rangle$  and  $d = \langle(\sigma_s, X_s)_{s < k}, C\rangle$ , we will write  $d \leq_f c$  if  $d \leq c$  via the function  $f : \ell \rightarrow k$ . If  $f$  is the identity function then  $d$  is a *simple extension*. Given  $s < k$ , if  $f^{-1}(t)$  is a singleton for all  $t \neq s$  then  $d$  is an *s-extension* of  $d$ .

The following lemma will be used to show that the generic set that we construct is infinite.

**Lemma 3.29.** Let  $c \in \mathbb{P}$  be a condition with a valid branch  $c^{[s]} = (\sigma, X)$ . Then there is a simple extension  $d \leq c$  of branch  $d^{[s]} = (\tau, Y)$  and an integer  $n > |\sigma|$  such that  $n \in \tau$ .  $\star$

PROOF. Let  $c = \langle(\sigma_s, X_s)_{s < k}, C\rangle$ . Since  $c^{[s]}$  is valid, we have  $X_s \in \bigcap_{e \in C} \mathcal{U}_e$ . Since  $\bigcap_{e \in C} \mathcal{U}_e$  only contains infinite sets, then  $X_s$  is infinite. Let  $n \in X_s$  be such that  $n > |\sigma_s|$ . Then, the condition  $d = \langle(\tau_s, X_s \cap ]n, \infty[)_{s < k}, C\rangle$  defined by  $\tau_s = \sigma_s \cup \{n\}$  and  $\tau_t = \sigma_t$  for  $t \neq s$  is a simple extension of  $c$  which satisfies the lemma.  $\blacksquare$

The following definition will be needed to develop the core of the argument.

**Definition 3.30.** Let  $\xi : \mathbb{N} \times 2^{<\mathbb{N}} \rightarrow \mathbb{N}$  be a computable function which takes the code  $e$  of a Turing functional  $\Phi_e(G, n)$ , a string  $\sigma$ , and which returns the code of the open class:

$$\{X : \exists \rho \subseteq X - \{0, \dots, |\sigma|\} \exists n \Phi_e(\sigma \cup \rho, n) \downarrow = \Phi_n(n)\}$$

We finally come to the key lemma of the proof.

**Lemma 3.31.** Let  $c^{[s]}$  be the branch of a condition  $c \in \mathbb{P}$ . Let  $\Phi_e(G, n)$  be a  $\{0, 1\}$ -valued Turing functional on all its oracles. There is an  $s$ -extension  $d \leq_f c$  such that for any valid branch  $d^{[t]}$  of  $d$  for which  $f(t) = s$ ,

there exists  $n$  for which:

$$d^{[t]} \Vdash \Phi_e(G, n) \downarrow = \Phi_n(n) \text{ or } d^{[t]} \Vdash \Phi_e(G, n) \uparrow \quad \star$$

PROOF. Let  $c = \langle (\sigma_s, X_s)_{s < k}, C \rangle$ . Let  $P(n, k, v)$  be the predicate:

$$\begin{aligned} & \forall Z_0 \sqcup \dots \sqcup Z_{k-1} = \mathbb{N} \exists j < k \\ & Z_j \in \bigcap_{e \in C, e < k} \mathcal{U}_e \text{ and } \exists \rho \subseteq Z_j - \{0, \dots, |\sigma_s|\} \Phi_e(\sigma_s \cup \rho, n) \downarrow = v \end{aligned}$$

The predicate  $P(n, k, v)$  is true iff the  $\Pi_1^0$  class

$$\left\{ Z_0 \sqcup \dots \sqcup Z_{k-1} = \mathbb{N} : \forall j < k \begin{array}{l} Z_j \notin \bigcap_{e \in C, e < k} \mathcal{U}_e \text{ or} \\ \forall \rho \subseteq Z_j - \{0, \dots, |\sigma_s|\} \Phi_e(\sigma_s \cup \rho, n) \uparrow \neq v \end{array} \right\}$$

is empty. The notation  $\Phi_e(\sigma_s \cup \rho, n) \uparrow \neq v$  above means  $\Phi_e(\sigma_s \cup \rho, n) \uparrow \vee \Phi_e(\sigma_s \cup \rho, n) \downarrow \neq v$ . By weak König's lemma, the predicate  $P(n, k, v)$  is therefore  $\Sigma_1^0$ .

Suppose first that for all  $k \in \mathbb{N}$  we have  $\forall n \exists v < 2 P(n, k, v)$ . Note that for all  $k \in \mathbb{N}$ , the set  $W_k = \{(n, v) : P(n, k, v)\}$  is c.e. We must therefore have an integer  $n \in \mathbb{N}$  such that  $(n, \Phi_n(n)) \in W_k$ , otherwise we would compute a DNC<sub>2</sub> function. So in particular:

$$\forall Z_0 \sqcup \dots \sqcup Z_{k-1} = \mathbb{N} \exists j < k Z_j \in \bigcap_{e \in C} \mathcal{U}_e \cap \mathcal{U}_{\xi(e, \sigma_s)}$$

It follows that  $\bigcap_{e \in C} \mathcal{U}_e \cap \mathcal{U}_{\xi(e, \sigma_s)}$  is a large class. If  $X_s \notin \bigcap_{e \in C} \mathcal{U}_e \cap \mathcal{U}_{\xi(e, \sigma_s)}$ , then  $d = \langle (\sigma_s, X_s)_{s < k}, C \cup \{\xi(e, \sigma_s)\} \rangle$  is an  $s$ -extension of  $c$  on which the branch  $d^{[s]}$  is not valid, and there is nothing more to check.

If  $X_s \in \bigcap_{e \in C} \mathcal{U}_e \cap \mathcal{U}_{\xi(e, \sigma_s)}$ , then there exists  $n$  and there is  $\rho \subseteq X_s - \{0, \dots, |\sigma_s|\}$  such that  $\Phi_e(\sigma_s \cup \rho, n) \downarrow = \Phi_n(n)$ . The condition  $d = \langle (\tau_s, X_s \setminus \{0, \dots, |\rho|\})_{s < k}, C \rangle$  defined by  $\tau_s = \sigma_s \cup \rho$  and  $\tau_t = \sigma_t$  for  $t \neq s$  is an  $s$ -extension of  $c$  such that  $d^{[s]} \Vdash \Phi_e(G, n) \downarrow = \Phi_n(n)$ .

Suppose now that there exists  $k \in \mathbb{N}$  such that

$$\exists n \forall v < 2 \neg P(n, k, v).$$

In particular, for integers  $k, n \in \mathbb{N}$ , we have two  $k$ -partitions  $Z_0^0 \sqcup \dots \sqcup Z_{k-1}^0 = \mathbb{N}$  and  $Z_0^1 \sqcup \dots \sqcup Z_{k-1}^1 = \mathbb{N}$  such that for  $v < 2$  we have:

$$\forall j < k \left( Z_j^v \notin \bigcap_{e \in C, e < k} \mathcal{U}_e \text{ or } \forall \rho \subseteq Z_j^v - \{0, \dots, |\sigma_s|\} \Phi_e(\sigma_s \cup \rho, n) \uparrow \neq v \right)$$

Let  $d = \langle (\tau_s, Y_s)_{s < \ell}, C \rangle$  be the  $s$ -extension of  $c$  obtained by duplicating the branches  $c^{[s]}$  into  $k^2$  branches  $c^{[s_{0,0}]}, \dots, c^{[s_{k-1,k-1}]}$ , such that  $\tau_{s_{i,j}} = \sigma_s$  and  $Y_{s_{i,j}} = X_s \cap Z_i^0 \cap Z_j^1$ . Every other branch  $c^{[t]}$  for  $t \neq s$  is identical in  $d$ .

Note that if the branch  $d^{[s_{i,j}]}$  is valid, then  $X_s \cap Z_i^0 \cap Z_j^1 \in \bigcap_{e \in C} \mathcal{U}_e$  and therefore  $\forall \rho \subseteq X_s \cap Z_i^0 \cap Z_j^1 - \{0, \dots, |\sigma_s|\} \Phi_e(\sigma_s \cup \rho, n) \uparrow \notin \{0, 1\}$ . Since  $\Phi_e$  is  $\{0, 1\}$ -valued, we therefore have  $d^{[s_{i,j}]} \Vdash \Phi_e(G, n) \uparrow$  for any  $i, j < k$  such that  $d^{[s_{i,j}]}$  is valid. This completes the proof of the lemma. ■

**Lemma 3.32.** Let  $c_0 \geq c_1 \geq \dots$  be a decreasing sequence of conditions. Then, there exists a function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $c_0^{[g(0)]} \geq c_1^{[g(1)]} \geq \dots$  and such that for all  $n$ ,  $c_n^{[g(n)]}$  is a valid branch of  $c_n$ . ★

PROOF. Let us first show the following: let  $c = \langle (\sigma_s, X_s)_{s < k}, C \rangle$  and  $d = \langle (\tau_s, Y_s)_{s < \ell}, D \rangle$  with  $d \leq_f c$ . If  $d^{[s]}$  is valid then  $c^{[f(s)]}$  is valid. Suppose  $d^{[s]}$  valid. Then,  $Y_s \in \bigcap_{e \in D} \mathcal{U}_e \subseteq \bigcap_{e \in C} \mathcal{U}_e$ . As  $Y_s \subseteq X_{f(s)}$ , and since each  $\mathcal{U}_e$  is upward-closed, then  $X_{f(s)} \in \bigcap_{e \in C} \mathcal{U}_e$  and therefore  $c^{[f(s)]}$  is valid.

The valid branches of each condition  $(c_i)_{i \in \mathbb{N}}$  thus form a tree for the order  $\leq$  between the Mathias conditions. Since the tree has is finitely-branching, it has an infinite path iff it is infinite. It is therefore sufficient to show that each  $c_i$  has a valid branch. Let  $c_i = \langle (\sigma_s, X_s)_{s < k}, C \rangle$ . Let  $m = \sup_{s < k} |\sigma_s|$ . Then,  $[0, m] \sqcup X_0 \sqcup \dots \sqcup X_{s-1} = \mathbb{N}$ . Since  $\bigcap_{e \in C} \mathcal{U}_e$  is large and contains only infinite sets, there necessarily exists  $s < k$  such that  $X_s \in \bigcap_{e \in C} \mathcal{U}_e$ . The  $c_i^{[s]}$  branch is therefore valid.

The function  $g$  of the lemma is given by the infinite path of our tree of valid branches. ■

PROOF OF THEOREM 3.24. Let  $c_0 \geq c_1 \geq \dots$  be a sufficiently generic filter. Note that we can start with the condition  $\langle (\epsilon, A^0), (\epsilon, A^1), C \rangle$ , where  $C$  is a computable set of codes such that  $\bigcap_{e \in C} \mathcal{U}_e$  is the class of infinite sets.

By Lemma 3.32, let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be such that  $c_0^{[g(0)]} \geq c_1^{[g(1)]} \geq \dots$  and such that for all  $i$ ,  $c_i^{[g(i)]}$  is a valid branch of  $c_i$ . According to Lemma 3.29, the resulting generic set  $G$  is infinite. According to Lemma 3.31, for any functional  $\Phi_e(G, n)$ , there exists  $n$  such that  $c_i^{[g(i)]} \Vdash \Phi_e(G, n) \downarrow = \Phi_n(n)$  or such that  $c_i^{[g(i)]} \Vdash \Phi_e(G, n) \uparrow$ . It follows that our generic is not of PA degree. Finally, by nature of the forcing conditions, we have  $G \subseteq A^i$  for  $i = 0$  or  $i = 1$ . ■

Liu's result is far from obvious, and the question of his iterated version is still open to this day:

**Question 3.33.** Let  $n \geq 1$ . For any set  $A$ , does there exist an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $H^{(n)}$  is not of PA degree relative to  $\emptyset^{(n)}$ ? ★

Monin and Patey [159] gave a partial positive answer showing that this was the case for the  $\Delta_2^0$  instances of  $\text{RT}_2^1$ .

### 3.4. $\Delta_2^0$ instances of $\text{RT}_2^1$

The  $\Delta_2^0$  instances are particularly important for the study of Ramsey's theorem for pairs, because they play a key role in the connection between  $\text{RT}_2^1$  and  $\text{RT}_2^2$ , as we will see in Section 4. The infinite pigeonhole principle being computably true, any  $\Delta_2^0$  instance of  $\text{RT}_2^1$  admits a  $\Delta_2^0$  solution. Is there a  $\Delta_2^0$  instance of  $\text{RT}_2^1$  with no computable solution? The question can be rephrased as follows: is there a  $\Delta_2^0$  set  $A$  that is both immune, and of immune complement (see Definition 7-1.1)? Proposition 3.34 gives a stronger result, showing the existence of a  $\Delta_2^0$  instance of  $\text{RT}_k^1$  whose solutions cannot even be approximated by blocks.

**Proposition 3.34.** There is a  $\Delta_2^0$  set  $A$  such that  $A$  and  $\mathbb{N} \setminus A$  are both hyperimmune. ★

PROOF. Recall that the *principal function* of an infinite set  $H$  is the function  $p_H : \mathbb{N} \rightarrow \mathbb{N}$  which to  $n$  associates the  $(n+1)$ -th element of  $H$ . Also recall that  $H$  is hyperimmune iff  $p_H$  is not bounded by any computable function.

We define a  $\emptyset'$ -computable sequence  $\sigma_0 \prec \sigma_1 \prec \dots$  by alternating the concatenation of long sequences of 1 with that of long sequences of 0, in order to diagonalize against all computable functions.

Suppose  $\sigma_{2e}$  defined, then  $\sigma_{2e+1} = \sigma_{2e}0^{k+1}1$  if  $\Phi_e(|\sigma_{2e}|+1) \downarrow = k$  and  $\sigma_{2e+1} = \sigma_{2e}1$  otherwise. Suppose  $\sigma_{2e+1}$  defined, then  $\sigma_{2e+2} = \sigma_{2e+1}1^{k+1}0$  if  $\Phi_e(|\sigma_{2e+1}|+1) \downarrow = k$  and  $\sigma_{2e+2} = \sigma_{2e+1}0$  otherwise.

We easily verify that if  $\Phi_e$  is total, then for  $m = |\sigma_{2e}|+1$ , we have  $p_A(m) \geq |\sigma_{2e+1}|-1 > \Phi_e(m)$ , and for  $m = |\sigma_{2e+1}|+1$  we have  $p_{\mathbb{N} \setminus A}(m) \geq |\sigma_{2e+2}|-1 > \Phi_e(m)$ . ■

Note that an infinite subset of a hyperimmune set is hyperimmune. Proposition 3.34 therefore gives us a  $\Delta_2^0$  instance  $B_0 \sqcup B_1 = \mathbb{N}$  of  $\text{RT}_2^1$  such that any solution is of hyperimmune degree. In particular, there exists a  $\Delta_2^0$  instance of  $\text{RT}_2^1$  with no computable solution.

**Exercise 3.35.** (★) Show that there exists a  $\Delta_2^0$  set  $A$  such that  $A$  and  $\mathbb{N} \setminus A$  are indeed immune. Deduce that any infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  is of DNC degree. ◇

**Exercize 3.36. (★)** Show that for all  $k \geq 2$ , there exists a  $\Delta_2^0$   $k$ -partition  $A_0 \sqcup A_1 \sqcup \dots \sqcup A_{k-1} = \mathbb{N}$  such that for all  $i < k$ ,  $\mathbb{N} \setminus A_i$  is hyperimmune.  $\diamond$

**Exercize 3.37. (★★)** Show that there exists an instance  $g$  of  $\text{RT}_3^1$  such that for any instance  $h$  of  $\text{RT}_2^1$ , there exists an  $h$ -homogeneous set which does not compute any  $g$ -homogeneous set. Hint: use exercises 3.36 and 3.22.  $\diamond$

The  $\Delta_2^0$  instance of  $\text{RT}_k^1$  created by Proposition 3.34 is however not too complex, and in particular has a low solution, as proved Hirschfeldt, Jockusch, Kjos-Hanssen and Lempp [88]:

**Proposition 3.38 ([88]).** Let  $A$  be a  $\Delta_2^0$  set such that  $\mathbb{N} \setminus A$  is hyperimmune. Then, there exists an infinite subset  $H \subseteq A$  of low degree.  $\star$

PROOF. We are going to build a  $\Delta_2^0$  sequence  $\sigma_0 \prec \sigma_1 \prec \dots \in 2^{<\mathbb{N}}$  of strings such that for all  $n$ ,  $\sigma_n \subseteq A$ , and such that for all  $e$ ,

$$\Phi_e^{\sigma_{e+1}}(e) \downarrow \text{ or } \forall \tau \succeq \sigma_{e+1} \Phi_e^\tau(e) \uparrow \quad (1)$$

Note that in the second clause of (1),  $\tau$  is not necessarily only included in  $A$ . Thus,  $\emptyset'$  manages to decide if  $\Phi_e^H(e) \downarrow$  for  $\{H\} = \bigcap_n [\sigma_n]$  by carrying out the construction up to  $\sigma_{e+1}$ , and by deciding using  $\emptyset'$  if we are in the first case or the second case.

*Construction.* Suppose we have built  $\sigma_e \in 2^{<\mathbb{N}}$ . Look  $\emptyset'$ -computably for an extension  $\sigma_{e+1} \succeq \sigma_e$  such that  $\sigma_{e+1} \subseteq A$  and such that (1) is satisfied. Let us show that there is always such an extension. For any  $n > |\sigma_e|$ , we look for  $\tau_n$ , an extension of  $0^n$ , such that  $\Phi_e^{\sigma_e \cup \tau_n}(e) \downarrow$ . If there is an  $n$  such that we do not find  $\tau_n$ , then  $\sigma_{e+1} = \sigma_e \cup 0^n$  is an extension of  $\sigma_e$  satisfying the second clause of (1). If  $\tau_n$  is always defined, then by hyperimmunity of  $\mathbb{N} \setminus A$ , there exists  $n > |\sigma_e|$  such that  $\tau_n \subseteq A$ . Then,  $\sigma_{e+1} = \sigma_e \cup \tau_n$  is an extension of  $\sigma_e$  satisfying the first clause of (1). In any case, such an extension exists. This concludes the construction and the proof of Proposition 3.38.  $\blacksquare$

Can we get a low solution for any  $\Delta_2^0$  instance of the infinite pigeonhole principle? The answer is no, and is proven using an infinite injury priority argument. We will omit here the proof which is long and of great complexity.

**Theorem 3.39 (Downey, Hirschfeldt, Lempp and Solomon [47])**  
*There exists a  $\Delta_2^0$  set  $A$  such that neither it nor its complement contains an infinite low subset.*

The instance of  $\text{RT}_2^1$  created by Theorem 3.39 is very different from the one created by Proposition 3.34, since any  $\Delta_2^0$  set whose complement is hyperimmune admits an infinite subset of low degree. Cholak, Jockusch and Slaman [32] have however proved that any  $\Delta_2^0$  instance of  $\text{RT}_2^1$  admits an “almost low” solution, in the sense that any PA degree relative to  $\emptyset'$  computes its Turing jump.

**Theorem 3.40 (Cholak, Jockusch and Slaman [32])**

*Let  $A$  be a  $\Delta_2^0$  set and  $P$  be a PA degree relative to  $\emptyset'$ . There exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $H' \leq_T P$ .*

PROOF. The proof of Theorem 3.40 is done using an effective construction of a generic filter for the Dzhafarov-Jockusch forcing. Recall that a *lowness code* of a low set  $X$  is an integer  $e$  such that  $\{n : \Phi_e(\emptyset', n) \downarrow\} = X'$  (see Section 5-7). The  $\Pi_1^0$  low basis theorem is uniform, in the sense that there exists a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that if  $\Phi_e$  is an infinite binary tree, then  $g(e)$  is the lowness code of a path to  $\Phi_e$ . By Exercise 22-7.5, there exists a Scott ideal  $\mathcal{I}$  whose elements are all low. The construction is uniform, and there is a computable function  $h_{\text{WKL}} : \mathbb{N} \rightarrow \mathbb{N}$  such that if  $e$  is the lowness code of an infinite binary tree in  $\mathcal{I}$ ,  $h_{\text{WKL}}(e)$  is the lowness code of an infinite path from this tree in  $\mathcal{I}$ . In addition, there is a computable function  $h_{\oplus} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  which, given two indices of lowness  $e_0, e_1$  of sets  $X$  and  $Y$ , returns a lowness code of the set  $X \oplus Y$  in  $\mathcal{I}$ . A *condition code*  $(\sigma_0, \sigma_1, X)$  is a triplet  $(\sigma_0, \sigma_1, e)$  be such that  $e$  is a lowness code of  $X$ . Thus, any Dzhafarov-Jockusch condition in the ideal  $\mathcal{I}$  can be finitely encoded.

Suppose (H1) (defined in Section 3.2). Indeed, otherwise, there exists an infinite set  $H \subseteq A^0$  or  $H \subseteq A^1$  of low degree. In particular,  $H' \leq_T \emptyset' \leq_T P$ . Knowing that  $A$  is fixed and  $\Delta_2^0$ , Lemma 3.13 is uniform in  $\emptyset'$ , in the sense that there exists a  $\emptyset'$ -computable procedure which takes as input a code  $(\sigma_0, \sigma_1, a)$  of condition  $(\sigma_0, \sigma_1, X)$  and an integer  $i < 2$ , and returns a code  $(\tau_0, \tau_1, b)$  of condition  $(\tau_0, \tau_1, Y)$  extending  $(\sigma_0, \sigma_1, X)$  such that there is an integer  $n \geq |\sigma_i|$  for which  $\tau_i(n) = 1$ . Knowing that  $\emptyset' \leq_T P$ ,  $P$  computes two infinite sets  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . However, we want to decide the Turing jump of  $G^0$  and  $G^1$ , using the forcing question.

Consider a condition  $c = (\sigma_0, \sigma_1, X)$  and two integers  $e_0, e_1 \in \mathbb{N}$ . By Lemma 3.15, the predicate  $c \Vdash \Phi_{e_0}^{G^0}(e_0) \downarrow \vee \Phi_{e_1}^{G^1}(e_1) \downarrow$  is  $\Sigma_1^0(X)$ . Thus,  $\emptyset'$  can decide, given a code of the condition  $c$ , whether the predicate is true or not. If the predicate is true, then  $\emptyset'$  can search for an integer  $i < 2$  and a finite string  $\rho \subseteq X \cap A^i$  such that  $\Phi_{e_i}^{\sigma_i \cup \rho}(e_i) \downarrow$ , and compute a lowness code of the set  $X \setminus \{0, \dots, |\rho|\}$ . Then, if the predicate is true,  $\emptyset'$  can compute a

code with an extension  $d$  of  $c$  and find an  $i < 2$  forcing  $\Phi_{e_i}^{G^i}(e_i) \downarrow$ . If the predicate is false, then  $\emptyset'$  can compute a lowness code of a set  $Z^0 \oplus Z^1$  such that  $Z^0 \sqcup Z^1 = X$ , and for all  $i < 2$  and all  $\rho \subseteq Z^i \cup X$ ,  $\Phi_{e_i}^{\sigma_i \cup \rho}(e_i) \uparrow$ . Until now, all operations could be performed using  $\emptyset'$ . On the other hand, it is necessary to resort to  $P$  to find an integer  $i < 2$  such that  $Z^i \cap X$  is infinite, and therefore to compute a code of the extension  $(\sigma_0, \sigma_1, Z^i \cap X)$ . A PA degree relative to  $\emptyset'$  is able to find, among two  $\Pi_2^0$  formulas  $F_0$  and  $F_1$ , knowing that at least one of the two is true, an integer  $i < 2$  such that  $F_i$  is true. Thus, if the predicate is false,  $P$  can compute a code of an extension  $d$  of  $c$  and  $i < 2$  such that  $d$  forces  $\Phi_{e_i}^{G^i}(e_i) \uparrow$ .

*Construction.* Build a  $P$ -computable sequence of condition codes  $c_0 \geq c_1 \geq \dots$  with  $c_s = (\sigma_0^s, \sigma_1^s, X^s)$  such that for all  $s \in \mathbb{N}$ :

- (1) there are  $n_0, n_1 > s$  such that  $\sigma_0^s(n_0) = 1$  and  $\sigma_1^s(n_1) = 1$ ;
- (2) if  $s = \langle e_0, e_1 \rangle$ , then there is  $i < 2$  such that  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_{e_i}^G(e_i) \downarrow$  or  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_{e_i}^G(e_i) \uparrow$ .

Let  $\{G^0\} = \bigcap_s [\sigma_0^s]$  and  $\{G^1\} = \bigcap_s [\sigma_1^s]$ . By (1),  $G^0$  and  $G^1$  are both infinite. By construction,  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . By (2), there exists  $i < 2$  such that for any  $e \in \mathbb{N}$ , there exists  $s$  such that  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_e^G(e) \downarrow$  or  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_e^G(e) \uparrow$ . Let us show that the Turing jump of  $G^i$  is  $P$ -computable. Indeed, to decide if  $\Phi_e^{G^i}(e) \downarrow$ , it suffices to execute, using  $P$ , the construction until you find an integer  $s$  such that  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_e^G(e) \downarrow$  or  $(\sigma_i^{s+1}, X^{s+1}) \Vdash \Phi_e^G(e) \uparrow$ . This completes the proof of Theorem 3.40. ■

Recall that a set  $H$  is  $\text{low}_2$  if  $H'' \leq_T \emptyset''$ . The  $\Pi_1^0$  low basis theorem (see Theorem 8-4.3) allows us to deduce the following corollary.

**Corollary 3.41**

Let  $A$  be a  $\Delta_2^0$  set. There exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  of  $\text{low}_2$  degree.

PROOF. By Theorem 8-4.3 relativized to  $\emptyset'$ , there exists a set  $P$  of PA degree relative to  $\emptyset'$  such that  $P' \leq_T \emptyset''$ . By Theorem 3.40, there exists an infinite set  $H \subseteq A$  or  $H \subseteq \mathbb{N} \setminus A$  such that  $H' \leq_T P$ . In particular,  $H'' \leq_T P' \leq_T \emptyset''$ , so  $H$  is of  $\text{low}_2$  degree. ■

## 4. Ramsey's theorem for pairs

Theorem 2.17 states that for all  $n \geq 3$ ,  $\forall k \text{RT}_k^n$  is equivalent to  $\text{ACA}_0$ . This theorem is proved by showing that any computable instance of  $\text{RT}_k^n$  admits an arithmetic solution, while there exists a computable coloring  $f : [\mathbb{N}]^3 \rightarrow 2$  for which any infinite homogeneous set computes  $\emptyset'$ . The pigeonhole principle  $\text{RT}_k^1$ , meanwhile, is computably true, and provable in  $\text{RCA}_0$  for any integer  $k$ , while the statement  $\forall k \text{RT}_k^1$  is equivalent to the principle  $\text{B}\Pi_1^0$  over  $\text{RCA}_0$ . In this section, we will see as stated that things are quite different for Ramsey's theorem for pairs. In particular, there exists a computable instance of  $\text{RT}_2^2$ , no solution of which computes  $\emptyset'$ , which will allow us to deduce Seetapun's theorem. We will also see that there exists a computable instance of  $\text{RT}_2^2$ , no solution of which is of PA degree, which will bring us to Liu's theorem.

### 4.1. Cohesiveness principle

The proof of Ramsey's theorem is done inductively using the notion of pre-homogeneous set. In the case of Ramsey's theorem for pairs, the correct notion to consider is that of cohesive set, which is a form of jump inversion of the notion of pre-homogeneous set.

**Definition 4.1.** Given two sets  $X$  and  $Y$ , we note  $X \subseteq^* Y$  if  $X \setminus Y$  is a finite set. Let  $(R_n)_{n \in \mathbb{N}}$  be a sequence of sets. An infinite set  $C$  is *cohesive* for  $(R_n)_{n \in \mathbb{N}}$  if it is infinite and if  $C \subseteq^* R_n$  or  $C \subseteq^* \mathbb{N} \setminus R_n$  for all  $n$ .  $\diamond$

Note that the notion of cohesive set makes sense only to infinite sets. We will therefore always assume that our cohesive sets are infinite.

#### Notation

We denote by COH the principle: for any sequence  $(R_n)_{n \in \mathbb{N}}$  there exists a cohesive set.

The existence of a cohesive set for all computable sets is easily proved using the restriction of Mathias forcing to conditions whose reservoirs are computable.

**Exercise 4.2. (\*)** Let  $\mathbb{P}$  be the set of Mathias conditions  $(\sigma, X)$  such that  $X$  is computable. Show that any set sufficiently generic for  $\mathbb{P}$  is cohesive for all computable sets.  $\diamond$

Let  $f : [\mathbb{N}]^2 \rightarrow 2$  be a computable instance of  $\text{RT}_2^2$ . While a pre-homogeneous set  $P$  transforms  $f$  into a  $P$ -computable instance of  $\text{RT}_2^1$ , a cohesive set  $C$  transforms  $f$  into a  $\Delta_2^0(P)$  instance of  $\text{RT}_2^1$ .

**Fact**

Let's fix  $k \geq 1$  and a coloring  $f : [\mathbb{N}]^2 \rightarrow k$ . Let  $(R_{n,i})_{n \in \mathbb{N}, i < k}$  be the sequence of sets defined by  $R_{n,i} = \{y \in \mathbb{N} : f(\{n, y\}) = i\}$ . Let  $C$  be an infinite cohesive set for  $(R_{n,i})_{n \in \mathbb{N}, i < k}$ . Then, for all  $x \in C$ ,  $\lim_{y \in C} f(\{x, y\})$  exists. In addition, let  $g : C \rightarrow k$  be the coloring  $\Delta_2^0(C \oplus f)$  defined by  $g(x) = \lim_{y \in C} f(\{x, y\})$ . For any infinite set  $Y \subseteq C$  homogeneous for  $g$ ,  $Y \oplus f$  computes an infinite homogeneous set for  $f$ .

PROOF. Let  $i < k$  be the homogeneity color of  $Y$  for  $g$ . Then, for all  $x \in Y$ ,  $\lim_{y \in C} f(\{x, y\}) = i$ . Then, let  $(x_n)_{n \in \mathbb{N}}$  be the sequence of elements of  $Y$  defined as follows: if  $x_s$  is defined for all  $s < n$ ,  $x_n$  is the smallest element of  $Y$  such that for all  $s < n$ ,  $f(\{x_s, x_n\}) = i$ . Such an  $x_n$  exists by the limit hypothesis. The set  $\{x_n : n \in \mathbb{N}\}$  is homogeneous for  $f$ . ■

Fact 4.1 therefore reduces the study of computable instances of  $\text{RT}_2^2$  to that of  $\Delta_2^0$  instances of  $\text{RT}_2^1$  and of computable instances of  $\text{COH}$ . We have already extensively studied the infinite pigeonhole principle in Section 3, which brings us to the study of  $\text{COH}$ .

The following proposition tells us that  $\text{COH}$  is not computably more complicated than  $\text{RT}_2^2$ . The reduction from  $\text{COH}$  to  $\text{RT}_2^2$  is a bit subtle, and requires some care to be formalized in  $\text{RCA}_0$ .

**Proposition 4.3 (Mileti [155]).**  $\text{RCA}_0 \vdash \text{RT}_2^2 \rightarrow \text{COH}$  and  $\text{COH} \leq_W \text{RT}_2^2$ . ★

PROOF. Assume  $\text{RT}_2^2$ . Let  $(R_n)_{n \in \mathbb{N}}$  be a sequence of sets. By computably adding sets to  $(R_n)_{n \in \mathbb{N}}$ , we can assume that for any  $x < y$ , there exists a smallest  $i \in \mathbb{N}$  such that  $R_i(x) \neq R_i(y)$ . Let  $i_{x,y}$  be this integer. We then define the coloring  $f : [\mathbb{N}]^2 \rightarrow 2$  for all  $x < y$  by  $f(\{x, y\}) = 1$  if  $x \in X_{i_{x,y}}$ , and  $f(\{x, y\}) = 0$  otherwise.

Let  $C = \{a_0 < a_1 < a_2 < \dots\}$  be a homogeneous set for  $f$  of color 0 (case 1 is symmetrical). Let  $(H_{n,b}^i)$  be the property “For any finite set  $F$ , if  $R_n(a_s) = i$  and  $R_n(a_{s+1}) = 1 - i$  for all  $s \in F$ , then  $|F| < b$ .”

Let us show that  $(H_{n,b}^1)$  implies  $(H_{n,b+1}^0)$ . Let  $F = \{s_0 < s_1 < \dots < s_{\ell-1}\}$  be a finite set such that  $R_n(a_s) = 0$  and  $R_n(a_{s+1}) = 1$  for all  $s \in F$ . Then, for all  $t < \ell - 1$ , there exists  $s \in ]s_t, s_{t+1}[$  such that  $R_n(a_s) = 1$  and  $R_n(a_{s+1}) = 0$ . Hence, by  $(H_{n,b}^1)$ ,  $\ell - 1 < b$ , therefore  $|F| < b + 1$ . Thus,  $(H_{n,b+1}^0)$  is true.

Let us show by induction on  $n$  that  $(H_{n,2^n}^1)$  is true. The formula  $(H_{n,2^n}^1)$  being  $\Pi_1^0$ , this induction is valid because  $\text{RCA}_0 \vdash \text{III}_1^0$ . Suppose that

$(H_{m,2^m}^1)$  is true for all  $m < n$ . Let  $F$  be a set such that  $R_n(a_s) = 1$  and  $R_n(a_{s+1}) = 0$  for all  $s \in F$ . As  $C$  is homogeneous for  $f$  of color 0, for all  $s \in F$ ,  $i_{a_s, a_{s+1}} < n$ , because otherwise, we would have  $i_{a_s, a_{s+1}} = n$ , and therefore  $f(\{a_s, a_{s+1}\}) = 1$ .

For all  $m < n$ , let  $F_m = \{s \in F : i_{a_s, a_{s+1}} = m\}$ . Note that for all  $s \in F_m$ ,  $R_m(a_s) = 0$  and  $R_m(a_{s+1}) = 1$  because  $C$  is homogeneous for  $f$  of color 0. So by  $(H_{m,2^m+1}^0)$ ,  $|F_m| \leq 2^m$ . As  $\bigcup_{m < n} F_m = F$ ,  $|F| \leq \sum_{m < n} 2^m < 2^n$ . Then,  $(H_{n,2^n}^1)$  is true.

It is clear that if  $(H_{n,2^n}^1)$  is true for all  $n$ , then  $C$  is cohesive for  $(R_n)_{n \in \mathbb{N}}$ . ■

Jockusch and Stephan, [102] studied the computational power of cohesive sets for some particular sequences of sets. The computational power of COH is highlighted by the notion of decision tree.

### Notation

Given a sequence  $(R_n)_{n \in \mathbb{N}}$  and  $\sigma \in 2^{<\mathbb{N}}$ , we denote by  $R^\sigma$  the intersection

$$\bigcap_{\sigma(n)=0} \mathbb{N} \setminus R_n \quad \bigcap_{\sigma(n)=1} R_n$$

We note  $\mathcal{C}((R_n)_{n \in \mathbb{N}}) = \{P \in 2^{\mathbb{N}} : \forall \sigma \prec P \ |R^\sigma| = \infty\}$ .

Intuitively, during the construction of an infinite set  $C$  cohesive for  $(R_n)_{n \in \mathbb{N}}$ , one is confronted for each  $n$  with the decision of ensuring that  $C \subseteq^* R_n$  or  $C \subseteq^* \mathbb{N} \setminus R_n$ . If we write  $R_n^0 = \mathbb{N} \setminus R_n$  and  $R_n^1 = R_n$ , then  $R^\sigma$  represents the decisions made for each  $n < |\sigma|$ . A decision is bad if  $R^\sigma$  is a finite set. Note that if the sequence  $(R_n)_{n \in \mathbb{N}}$  is computable, then the class  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$  is  $\Pi_1^0(\emptyset')$ .

The two following propositions clarify the links between the power required to compute a path in a  $\Delta_2^0$  tree and that allowing to compute a cohesive set for a computable sequence of sets. They have been independently proven by Brattka and Rakotoniaina [25] and Patey [171].

**Proposition 4.4.** Let  $(R_n)_{n \in \mathbb{N}}$  be a computable sequence of sets. A set computes a cohesive set for  $(R_n)_{n \in \mathbb{N}}$  iff its Turing jump computes a member of  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$ . ★

PROOF. Let  $C$  be a cohesive set for  $(R_n)_{n \in \mathbb{N}}$ . Let  $P \in 2^{\mathbb{N}}$  be the unique sequence such that for all  $\sigma \prec P$ ,  $C \subseteq^* R^\sigma$ . The set  $P$  is  $\Delta_2^0(C)$ , and for all  $\sigma \prec P$ ,  $R^\sigma$  is infinite, so  $P \in \mathcal{C}((R_n)_{n \in \mathbb{N}})$ .

Conversely, let  $X$  be a set whose Turing jump computes a member  $P$  of  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$ . By Shoenfield's limit lemma, there exists an  $X$ -computable

function  $f : \mathbb{N}^2 \rightarrow 2$  such that  $\forall x \lim_y f(x, y) = P(x)$ . Let  $(x_n)_{n \in \mathbb{N}}$  be the strictly increasing sequence of integers defined  $X$ -computably as follows: Initially,  $x_0 = 0$ . At each step  $s > 0$ , look for a string  $\sigma \in 2^{<\mathbb{N}}$  of length  $s$  and an integer  $x_s \in R^\sigma$  greater than  $x_{s-1}$  such that  $f(x, x_s) = \sigma(x)$  for all  $x < s$ . Let us show that such  $\sigma$  and  $x_s$  still exist. Indeed, there is a threshold  $N_0 \in \mathbb{N}$  such that for all  $x_s > N_0$  and all  $x < s$ ,  $f(x, x_s) = P(x)$ . In particular, for  $\sigma = P \upharpoonright_s$ ,  $R^\sigma$  is infinite, so any  $x_s \in R^\sigma$  such that  $x_s > N_0$ , satisfies the desired property.

Let  $C = \{x_n : n \in \mathbb{N}\}$ . Let us show that  $C$  is cohesive for  $(R_n)_{n \in \mathbb{N}}$ . For all  $x \in \mathbb{N}$ , there exists a threshold  $N_1 \in \mathbb{N}$  such that for all  $s > N_1$ ,  $f(x, x_s) = P(x)$ . In particular,  $x_s \in R_x^{P(x)}$  for all  $s > N_1$ , so  $C \subseteq^* R_x^{P(x)}$ . ■

#### Corollary 4.5

*Let  $(R_n)_{n \in \mathbb{N}}$  be a computable sequence of sets. Any high degree computes a cohesive set for  $(R_n)_{n \in \mathbb{N}}$ .*

PROOF. Using  $\emptyset''$ , we can inductively compute  $\sigma_0 \prec \sigma_1 \prec \dots$ , looking from  $\sigma_n$  for a value  $i \in \{0, 1\}$  such that  $|R^{\sigma_n i}| = \infty$ . In particular if  $X' \geq_T \emptyset''$  then  $X'$  computes a member of  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$  and therefore  $X$  computes a cohesive set for  $(R_n)_{n \in \mathbb{N}}$ . ■

It is possible to improve the preceding corollary to obtain a characterization of the ability to compute a  $(R_n)_{n \in \mathbb{N}}$ -cohesive set for any computable sequence  $(R_n)_{n \in \mathbb{N}}$ .

**Proposition 4.6.** A non-empty class  $\mathcal{D} \subseteq 2^\mathbb{N}$  is  $\Pi_1^0(\emptyset')$  iff there exists a computable sequence of sets  $(R_n)_{n \in \mathbb{N}}$  such that  $\mathcal{C}((R_n)_{n \in \mathbb{N}}) = \mathcal{D}$ . ★

PROOF. Let  $(R_n)_{n \in \mathbb{N}}$  be a computable sequence of sets. Then,  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$  is the  $\Pi_1^0(\emptyset')$  class given by  $\{P \in 2^\mathbb{N} : \forall \sigma \prec P \forall x |R^\sigma| > x\}$ . Note that we can obtain a code of the computable set  $R^\sigma$  uniformly in  $\sigma$ , and therefore that  $\emptyset'$  can uniformly answer the question of whether  $|R^\sigma| > x$ . This makes the class  $\mathcal{C}((R_n)_{n \in \mathbb{N}})$  well  $\Pi_1^0(\emptyset')$ .

For the reverse direction, let  $T \subseteq 2^{<\mathbb{N}}$  be a  $\Delta_2^0$  tree such that  $[T] = \mathcal{D}$ . By Shoenfield's limit lemma, there exists a computable function  $f : 2^{<\mathbb{N}} \times \mathbb{N} \rightarrow 2$  such that for all  $\sigma \in 2^{<\mathbb{N}}$ ,  $\lim_s f(\sigma, s) = T(\sigma)$ .  $\mathcal{D}$  being non-empty,  $T$  is an infinite tree, so we can assume that

- (1) for any  $s \in \mathbb{N}$ , the set  $U_s = \{\sigma \in 2^s : f(\sigma, s) = 1\}$  is not empty.
- (2) for all  $\sigma \prec \tau$  and  $s$ , if  $g(\tau, s) = 1$ , then  $g(\sigma, s) = 1$ .

We will build a computable sequence of sets  $(R_n)_{n \in \mathbb{N}}$  such that  $\mathcal{C}((R_n)_{n \in \mathbb{N}}) = \mathcal{D}$  as follows: In step 0,  $R_n = \emptyset$  for all  $n \in \mathbb{N}$ . Suppose we have already decided  $\mathcal{R}_n \upharpoonright_{k_s}$  for every  $n \in \mathbb{N}$ . In step  $s+1$ , let  $p = |U_{s+1}|$ . We will add elements  $[n_s, k_s+p[$  to  $R_0, \dots, R_s$  such that for any string  $\sigma \in 2^{<\mathbb{N}}$  of length  $s+1$ ,  $R^\sigma \cap [k_s, k_s+p[ \neq \emptyset$  iff  $\sigma \in U_{s+1}$ . For that, let  $\{\sigma_0, \dots, \sigma_{p-1}\} = U_{s+1}$ . For any  $j \leq s$ , add to  $R_j$  the set  $\{k_s + i : \sigma_i(j) = 1, i < p\}$ . Let  $k_{s+1} = k_s + p$ . This completes the construction. Note that by (1),  $(k_s)_{s \in \mathbb{N}}$  is a strictly increasing sequence, so the sets  $(R_n)_{n \in \mathbb{N}}$  are defined for arbitrarily large initial segments.

Let us show that  $R^\sigma$  is infinite iff  $\sigma \prec P$  for a  $P \in \mathcal{D}$ . If  $\sigma$  is not a prefix of any member of  $\mathcal{D}$ , then by compactness, there exists a  $t$  such that no  $\tau \succeq \sigma$  of length  $t$  is in  $T$ . Let  $N_0 \in \mathbb{N}$  be a threshold such that for any  $s > N_0$ ,  $\sigma$  has no extension in  $U_{s+1}$ . Then, by construction, for all  $s > N_0$ ,  $R^\sigma \cap [k_s, k_{s+1}[ = \emptyset$ , so  $R^\sigma$  is finite.

Conversely, if  $\sigma \prec P$  for a  $P \in \mathcal{D}$ , then by (2), there is a threshold  $N_1$  such that for all  $s > N_0$ ,  $\sigma$  has an extension in  $U_{s+1}$ . Then, by construction, for all  $s > N_0$ ,  $R^\sigma \cap [k_s, k_{s+1}[ \neq \emptyset$ , so  $R^\sigma$  is infinite. ■

**Corollary 4.7 (Jockusch and Stephan [102])**

Let  $X \in 2^{\mathbb{N}}$ . The two statements are equivalent:

- (1) For any computable sequence of sets  $(R_n)_{n \in \mathbb{N}}$ ,  $X$  computes a cohesive set for  $(R_n)_{n \in \mathbb{N}}$ .
- (2)  $X'$  is PA  $(\emptyset')$ .

PROOF. Immediate by Proposition 4.4 and Proposition 4.6. ■

**Universal instance for COH**

We saw in Chapter 8 that the  $\Pi_1^0$  class of the completions of Peano arithmetic is *universal*, in the sense that any PA degree computes a member of any  $\Pi_1^0$  class not empty (see Remark 8-6). Likewise, COH has a universal instance  $(R_n)_{n \in \mathbb{N}}$ , in the sense that any cohesive set for  $(R_n)_{n \in \mathbb{N}}$  computes a cohesive set for any computable sequence. Indeed, by Proposition 4.4 and Proposition 4.6, there exists a computable sequence of sets such that any cohesive set has a Turing jump of PA degree relative to  $\emptyset'$ . By Corollary 4.7, any set whose Turing jump is PA relative to  $\emptyset'$  computes a cohesive set for any computable sequence of sets.

Let us end the computational study of the cohesiveness principle with the following proposition which will be very useful in Section 4.2. **TODO**

**Proposition 4.8.** Let  $A$  be a set and  $\mathcal{B} \subseteq \mathbb{N}^{\mathbb{N}}$  a non-empty closed class in Baire space having no computable member. There is a set  $X$  such that  $X' \geq_T A$  and  $X$  does not compute a member of  $\mathcal{B}$ . ★

PROOF. The proof is similar to that of Proposition 4-10.2. Consider the following forcing notion: a *condition* is a couple  $(g, n)$  for  $n \in \mathbb{N}$ , where  $g$  is a function of type  $\{0, \dots, p\}^2 \rightarrow 2$  for  $p \in \mathbb{N}$ . A condition  $(h, m)$  *extends*  $(g, n)$  if  $g \subseteq h$ ,  $m \geq n$ , and for all  $(x, y) \in \text{dom } h \setminus \text{dom } g$ , if  $x < n$ , then  $y = A(x)$ . Thus, for any condition  $(g, n)$ , the first  $n$  columns of  $g$  are stabilized on the value of  $A$ . Any maximal filter  $F$  can be interpreted as a function  $G = \bigcup_{(g, n) \in F} g$ . It is easy to verify that  $\text{dom } G = \mathbb{N} \times \mathbb{N}$ , and that if  $F$  is sufficiently generic, then for all  $x \in \mathbb{N}$ ,  $\lim_y G(x, y) = A(x)$ . Thus,  $A$  is  $\Delta_2^0(G)$ .

**Lemma 4.9.** For any condition  $c = (g, n)$  and any  $e \in \mathbb{N}$ , there is an extension  $d$  of  $c$  forcing  $\Phi_e^G \notin \mathcal{B}$ . ★

PROOF. Three cases arise. Case 1: There are  $(h, n) \leq (g, n)$  and  $\sigma \in \mathbb{N}^{<\mathbb{N}}$  such that  $\Phi_e^h \upharpoonright_{|\sigma|} = \sigma$  with  $[\sigma] \cap \mathcal{B} = \emptyset$ . In this case,  $(h, n)$  forces  $\Phi_e^G \notin \mathcal{B}$ . Case 2: There exists  $(h, n) \leq (g, n)$  and  $x$  such that  $\Phi_e^f(x) \uparrow$  for all  $(f, m) \leq (h, n)$ . Then,  $(h, n)$  forces  $\Phi_e^G(x) \uparrow$ , so  $\Phi_e^G \notin \mathcal{B}$ . Case 3: cases 1 and 2 are false. We then obtain a contradiction by computing a sequence  $(h, n) \geq (h_0, n) \geq (h_1, n) \geq \dots$  of conditions such that  $\Phi_e^{h_{x+1}} \succeq \sigma_x$  with  $|\sigma_x| \geq x$ . According to the negation of case 2 we can always continue such a sequence (let us note in this respect that the search for condition extensions  $(h_x, n)$  is computable because it only needs to know the first  $n$  bits of  $A$ , and that if  $(h_x, m)$  is an extension of  $(h_{x-1}, n)$ ,  $(h_x, n)$  is also a valid extension of  $(h_{x-1}, n)$ ). From the negation of case 1, we have  $[\sigma_x] \cap \mathcal{B} \neq \emptyset$  for all  $x$ . Let  $P \in \bigcap_x [\sigma_x]$ . Since  $\mathcal{B}$  is closed,  $P \in \mathcal{B}$ , which contradicts the fact that  $\mathcal{B}$  does not have any computable member. ■

Let  $F$  be a sufficiently generic filter for this forcing notion, and let  $G = \bigcup_{(g, n) \in F} g$ . By Lemma 4.9,  $G$  does not compute any member of  $\mathcal{B}$ , and by definition of the forcing notion,  $A$  is  $\Delta_2^0(G)$ . ■

#### Corollary 4.10

Let  $(R_n)_{n \in \mathbb{N}}$  be a computable sequence and  $\mathcal{B} \subseteq \mathbb{N}^{\mathbb{N}}$  a non-empty closed class in Baire space having no computable member. There exists a cohesive set  $C$  for  $(R_n)_{n \in \mathbb{N}}$  computing no member of  $\mathcal{B}$ .

PROOF. Let  $A = \emptyset''$ . By Proposition 4.8, there exists a set  $X$  such that  $X' \geq_T \emptyset''$  and  $X$  does not compute any member of  $\mathcal{B}$ . By Corollary 4.7,  $X$  computes a cohesive set for  $(R_n)_{n \in \mathbb{N}}$ . ■

**Exercise 4.11** (Wang [231]). (★) Show with the help of computable Mathias forcing that if  $C$  is not  $\Sigma_1^0$ , for any computable sequence  $(R_n)_{n \in \mathbb{N}}$ , there exists a cohesive set  $D$  such that  $C$  is not  $\Sigma_1^0(D)$ .  $\diamond$

## 4.2. Computational weakness of $\text{RT}_2^2$

The cohesiveness principle being computably very weak, a large part of the computational properties of the non-computable instances of the infinite pigeonhole principle is found in the computable instances of Ramsey's theorem for pairs.

Recall that a problem  $P$  preserves a weakness property  $\mathcal{W}$  if for any  $Z \in \mathcal{W}$  and any instance  $X \leq_T Z$  of  $P$ , there exists a solution  $Y$  such that  $Z \oplus Y \in \mathcal{W}$  (see Section 24-1.1). We can define a strong notion of preservation by removing the effectiveness constraint on  $X$ :

**Definition 4.12.** Let  $\mathcal{W}$  be a weakness property. A problem  $Q$  *strongly preserves*  $\mathcal{W}$  if for all  $Z \in \mathcal{W}$ , any (arbitrary) instance  $X$  of  $Q$  admits a solution  $Y$  such that  $Z \oplus Y \in \mathcal{W}$ .  $\diamond$

We have the following formal correspondence between the computable instances of  $\text{RT}_2^2$  and the arbitrary instances of  $\text{RT}_2^1$ :

**Proposition 4.13.** Let  $\mathcal{B} \subseteq \mathbb{N}^{\mathbb{N}}$  be a non-empty closed class in Baire space and  $\mathcal{W} = \{Z : Z \text{ computes no member of } \mathcal{B}\}$ . Then,  $\text{RT}_2^2$  preserves  $\mathcal{W}$  iff  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ .  $\star$

PROOF. Suppose that  $\text{RT}_2^2$  preserves  $\mathcal{W}$ . Let  $Z$  be a set computing no member of  $\mathcal{B}$  and  $g : \mathbb{N} \rightarrow 2$  be an instance of  $\text{RT}_2^1$ . By Proposition 4.8, there is a set  $X$  such that  $g$  is  $\Delta_2^0(X)$  and  $X \oplus Z$  does not compute any member of  $\mathcal{B}$ . By Shoenfield's limit lemma, there exists an  $X$ -computable function  $f : [\mathbb{N}]^2 \rightarrow 2$  such that, for all  $x \in \mathbb{N}$ ,  $\lim_y f(\{x, y\}) = g(x)$ . As  $\text{RT}_2^2$  preserves  $\mathcal{W}$ , there exists an infinite set  $H$  homogeneous for  $f$  such that  $H \oplus X \oplus Z$  does not compute any member of  $\mathcal{B}$ . In particular,  $H$  is homogeneous for  $g$  and  $H \oplus Z$  does not compute any member of  $\mathcal{B}$ , so  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ .

Suppose that  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ . Let  $Z$  be a set computing no member of  $\mathcal{B}$ , and  $f \leq_T Z$  be an instance of  $\text{RT}_2^2$ . Let  $(R_n)_{n \in \mathbb{N}}$  be the  $Z$ -computable sequence defined by  $R_x = \{y : f(\{x, y\}) = 1\}$ . By Corollary 4.10, there exists a cohesive set  $C$  for  $(R_n)_{n \in \mathbb{N}}$  such that  $C \oplus Z$  does not compute any member of  $\mathcal{B}$ . As  $C$  is cohesive for  $(R_n)_{n \in \mathbb{N}}$ , for all  $x \in C$ ,  $\lim_{y \in C} f(\{x, y\})$  exists. Let  $g : C \rightarrow 2$  be the coloring defined by  $g(x) = \lim_{y \in C} f(\{x, y\})$ . By Proposition 2.2 and knowing that  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ , there exists an infinite set  $H \subseteq C$  homogeneous

for  $g$  such that  $H \oplus C \oplus Z$  does not compute any member of  $\mathcal{B}$ . By Fact 4.1,  $H \oplus C \oplus Z$  computes an infinite set  $Y$  homogeneous for  $f$ . In particular,  $Y \oplus Z$  does not compute any member of  $\mathcal{B}$ , so  $\text{RT}_2^2$  preserves  $\mathcal{W}$ . ■

We now have the necessary elements to prove Seetapun's theorem, which states that  $\text{RT}_2^2$ , unlike  $\text{RT}_2^3$ , does not imply  $\text{ACA}_0$  over  $\text{RCA}_0$ .

**Theorem 4.14 (Seetapun [194])**

*For any non-computable set  $C$  and any computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$ , there exists an infinite set  $H$  homogeneous for  $f$  such that  $C \not\leq_T H$ .*

PROOF. Let  $C$  be a non-computable set and  $\mathcal{W} = \{Z : C \not\leq_T Z\}$ . By Theorem 3.8 relativized,  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ , so by Proposition 4.13,  $\text{RT}_2^2$  preserves  $\mathcal{W}$ . As  $C$  is not computable,  $\emptyset \in \mathcal{W}$ , so by the definition of preservation for  $Z = \emptyset$ , we obtain the statement of Theorem 4.14. ■

The proof of Seetapun's theorem can be relativized well, and thus allows to create, for any non-computable set  $C$ , a Turing ideal satisfying  $\text{RT}_2^2$  but not containing  $C$ .

**Corollary 4.15 (Seetapun [194])**

$\text{RCA}_0 + \text{RT}_2^2 \not\vdash \text{ACA}_0$ .

PROOF. Let  $\mathcal{W} = \{X : \emptyset' \not\leq_T X\}$ . By Theorem 4.14,  $\text{RT}_2^2$  preserves  $\mathcal{W}$ , but  $\text{ACA}_0$  does not preserve  $\mathcal{W}$ , so by Corollary 24-1.12,  $\text{RCA}_0 + \text{RT}_2^2 \not\vdash \text{ACA}_0$ . ■

Jockusch [98] proved the existence of a computable instance of  $\text{RT}_2^2$  with no  $\Delta_2^0$  solution, thus showing that  $\text{WKL}_0$  does not imply  $\text{RT}_2^2$  over  $\text{RCA}_0$ . Knowing that  $\text{RT}_2^2$  is strictly weaker than  $\text{ACA}_0$ , the last open question in connection with the five big systems of Reverse Mathematics is to know whether  $\text{RT}_2^2$  implies  $\text{WKL}_0$  over  $\text{RCA}_0$ . This question, known as *Seetapun's enigma*, also remained open for several decades, before Liu [145] answered it negatively, thus showing the existence of a natural theorem which is not linearly ordered with the Big Five.

**Theorem 4.16 (Liu [145])**

*For any computable function  $f : [\mathbb{N}]^2 \rightarrow 2$ , there exists an infinite set  $H$  homogeneous for  $f$  which is not of PA degree.*

PROOF. Let  $\mathcal{W} = \{X : X \text{ is not of PA degree}\}$ . By Theorem 3.24 relativized,  $\text{RT}_2^1$  strongly preserves  $\mathcal{W}$ , so by Proposition 4.13,  $\text{RT}_2^2$  preserves  $\mathcal{W}$ .

As  $\emptyset \in \mathcal{W}$ , by the definition of preservation for  $Z = \emptyset$ , we obtain the statement of Theorem 4.16. ■

**Corollary 4.17**

$\text{RCA}_0 + \text{RT}_2^2 \not\vdash \text{WKL}_0$ .

PROOF. Let  $\mathcal{W} = \{X : X \text{ is not of PA degree}\}$ . By Theorem 4.16,  $\text{RT}_2^2$  preserves  $\mathcal{W}$ , but  $\text{WKL}_0$  does not preserve  $\mathcal{W}$ , so by Corollary 24-1.12,  $\text{RCA}_0 + \text{RT}_2^2 \not\vdash \text{WKL}_0$ . ■

**Exercise 4.18.** Show that for any non- $\Sigma_1^0$  set  $C$  and any computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$ , there exists an infinite set  $H$  homogeneous for  $f$  such that  $C$  is not  $\Sigma_1^0(H)$ . ◇

We have seen, with Proposition 4.4 and Proposition 4.4; the existence of a computable instance of COH whose solutions have a Turing jump of PA degree relative to  $\emptyset'$ . By Proposition 4.3,  $\text{COH} \leq_W \text{RT}_2^2$ , therefore there exists a computable instance of  $\text{RT}_2^2$  whose solutions have a Turing jump of PA degree relative to  $\emptyset'$ . Can we do better? The following theorem of Cholak, Jockusch and Slaman [32] affirms in a certain way the optimality of this bound by showing that it is always possible to obtain a solution whose Turing jump is bounded by a PA degree relative to  $\emptyset'$ .

**Theorem 4.19 (Cholak, Jockusch and Slaman [32])**

*For any computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$  and any set  $P$  of PA degree relative to  $\emptyset'$ , there exists an infinite set  $H$  homogeneous for  $f$  such that  $H' \leq_T P$ .*

PROOF. By Exercise 14-1.12 relativized to  $\emptyset'$ , there exists a set  $P_1$  of PA degree relative to  $\emptyset'$ , such that  $P$  is of PA degree relative to  $P_1$ . By Friedberg's jump inversion theorem (see Corollary 10-3.33), there exists a set  $X$  such that  $X' \equiv_T P_1$ . In particular,  $P$  is of PA degree relative to  $X'$ .

Let  $(R_n)_{n \in \mathbb{N}}$  be the computable sequence defined by  $R_x = \{y : f(\{x, y\}) = 1\}$ . By Corollary 4.7,  $X$  computes a cohesive set  $C$  for  $(R_n)_{n \in \mathbb{N}}$ . In particular, for all  $x \in C$ ,  $\lim_{y \in C} f(\{x, y\})$  exists. Let  $g : C \rightarrow 2$  be the  $\Delta_2^0(C)$  coloring defined by  $g(x) = \lim_{y \in C} f(\{x, y\})$ . By Proposition 2.2 and Theorem 3.40 relativized to  $C$ , there exists an infinite set  $H \subseteq C$  homogeneous for  $g$  such that  $(H \oplus C)' \leq_T P \oplus C \leq_T P$ . By Fact 4.1,  $H \oplus C$  computes an infinite set  $Y$  homogeneous for  $f$ . In particular,  $Y' \leq_T (H \oplus C)' \leq_T P$ . ■

**Corollary 4.20 (Cholak, Jockusch and Slaman [32])**

For any computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$ , there exists an infinite set  $H$  homogeneous for  $f$  of low<sub>2</sub> degree.

PROOF. By Theorem 8-4.3 relativized to  $\emptyset'$ , there exists a set  $P$  of PA degree relative to  $\emptyset'$  such that  $P' \leq_T \emptyset''$ . By Theorem 4.19, there exists an infinite set  $H$  homogeneous for  $f$  such that  $H' \leq_T P$ . In particular,  $H'' \leq_T P' \leq_T \emptyset''$ , so  $H$  is of low<sub>2</sub> degree. ■

Seetapun's theorem generalizes to the whole arithmetic hierarchy:

**Proposition 4.21 (Wang [231]).** For any  $n \geq 1$ , any non- $\Delta_n^0$  set  $C$  and any computable coloring  $f : [\mathbb{N}]^2 \rightarrow 2$ , there exists an infinite set  $H$  homogeneous for  $f$  such that  $C$  is not  $\Delta_n^0(H)$ . ★

PROOF. For  $n = 1$ , it is Theorem 4.14. Let  $n \geq 2$ . If  $C$  is not  $\Delta_n^0$ , then it is not  $\Delta_{n-1}^0(\emptyset')$ . By the cone avoidance basis theorem relativized to  $\emptyset'$  (see Theorem 8-4.7), there exists a set  $P$  of PA degree relative to  $\emptyset'$  such that  $C$  is not  $\Delta_{n-1}^0(P)$ . By Theorem 4.19, there exists an infinite set  $H$  homogeneous for  $f$  such that  $H' \leq_T P$ . In particular,  $C$  is not  $\Delta_{n-1}^0(H')$ , so it is not  $\Delta_n^0(H)$ . ■

### 4.3. Combinatorial invariants of Ramsey's theorem

Let us recall here the notion of strong preservation of Definition 4.12. We will say that a problem  $P$  *strongly avoids a cone* if for any non-computable  $C$ , any set  $Z$  not computing  $C$ , any arbitrary instance of  $P$  has a solution  $X$  such that  $Z \oplus X \not\leq_T C$ . The notion of (non-strong) cone avoidance of Definition 24-1.13 is identical, but for  $Z$ -computable instances of  $P$ .

There is a big combinatorial difference from the point of view of Computability Theory between  $\text{RT}_k^1$  and  $\text{RT}_k^2$ . The first strongly avoids a cone, but not the second: we have indeed seen that one could build a  $\Delta_2^0$  instance of  $\text{RT}_2^2$  of which all the solutions compute  $\emptyset'$ . Wang found that a slight weakening of Ramsey's theorem for pairs restored strong cone avoidance.

#### Notation

We denote by  $\text{RT}_{k,\ell}^n$  the statement “For any coloring  $f : [\mathbb{N}]^n \rightarrow k$ , there exists an infinite set  $H \subseteq \mathbb{N}$  such that  $|f([H]^n)| \leq \ell$ .”

In particular,  $\text{RT}_{k,1}^n$  is the statement  $\text{RT}_k^n$ . Much of the computational analysis of Ramsey's theorem generalizes to  $\text{RT}_{k,\ell}^n$ . In particular, for all  $k \geq$

$\ell + 1$ ,  $\text{RT}_{k,\ell}^n$  implies  $\text{RT}_{k+1,\ell}^n$  over  $\text{RCA}_0$  by a color blindness argument (see Proposition 2.3). When  $k \leq \ell$ ,  $\text{RT}_{k,\ell}^n$  is trivial and therefore computably true. Thus, only the  $\text{RT}_{\ell+1,\ell}^n$  case is really interesting from the point of view of Reverse Mathematics.

Wang [230] proved that for any  $n$ , when  $\ell$  is sufficiently large compared to  $n$ , the statement  $\forall k \text{RT}_{k,\ell}^n$  strongly avoids a cone. Cholak and Patey [33] studied the exact bound  $\ell$  as a function of  $n$  for which  $\forall k \text{RT}_{k,\ell}^n$  strongly avoids a cone, and this one coincides in a surprising way with the Catalan numbers, well known in Combinatorics.

**Definition 4.22.** The *Catalan sequence* is defined inductively as follows:

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$$

The first values of the Catalan sequence are 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16 796, 58 786, ... The Catalan sequence admits many combinatorial characterizations.

**Example 4.23.** A *Dyck word* is a string with values in  $\{ (, ) \}$  such that the number of open parentheses is equal to the number of close parentheses, and such that for any prefix, the number of close parentheses is never greater than the number of open parentheses. For example,  $((()))()$  is a valid Dyck word, while  $)()$  and  $((()))()$  are not. Note that a Dyck word is necessarily of even length.  $C_n$  is the number of Dyck words of length  $2n$ . For example, there is only one Dyck word of length 0, namely the empty string, so  $C_0 = 1$ . There is also a unique Dyck word of length 2, which is  $()$ , hence  $C_1 = 1$ . On the other hand, there are 2 Dyck words of length 4, which are  $()()$  and  $(())$ .

It is with the following elegant characterization that we conclude our analysis of Ramsey's theorem from the point of view of Computability Theory.

**Theorem 4.24 (Wang [230], Cholak and Patey [33])**

For all  $n \geq 1$ ,  $\forall k \text{RT}_{k,\ell}^n$  strongly avoids 1 cone iff  $\ell \geq C_n$ . Thus,  $\forall k \text{RT}_{k,\ell}^{n+1}$  avoids 1 cone iff  $\ell \geq C_n$ .



## Part IV

# Higher Computability Theory



# Chapter 26

## Introduction

It is not as easy to define Higher Computability Theory *per se* as it was to define each of the first three parts of this book, namely Classical Computability, Algorithmic Randomness and Reverse Mathematics. Indeed, Higher Computability Theory does not handle mathematical objects with immediate epistemological correspondence, as is the case of computable functions or random sequences. This field of research is above all a bridge, between Classical Computability Theory on one hand, and Set Theory and Descriptive Set Theory on the other hand. For these reasons, Higher Computability Theory is more easily defined by reference to these theories, by adopting either a *top-down* approach, starting from Descriptive Set Theory viewpoint and considering Higher Computability Theory as an effectivization of its concepts; or a *bottom-up* approach, by considering Higher Computability Theory as a generalization of Classical Computability Theory for which we allow ourselves *infinitely many* stages of computation, via the notion of *ordinals*.

**For the set theorist**, Higher Computability Theory can be seen as the study of the first levels of the *constructible universe* hierarchy. Constructibility is a field of study initiated by Gödel, who from a model of ZF, showed how to construct a sub-model of ZF satisfying both the axiom of choice and the continuum hypothesis. Gödel's model is the so-called constructible universe: the sets that we can build recursively along the ordinals (which we will formally define soon). However, this is a point of view that we will not develop further in this book, and the reader curious to learn more on that can refer to [34] or [191].

Higher Computability Theory can also be seen as an effective version of Descriptive Set Theory. This branch of mathematics classifies sets according to their definitional complexity. It was born from the continuation of the work of Borel, Baire and Lebesgue, by Nikolai Luzin and Mikhail Souslin [218] at the beginning of the 20th century. We will discuss historical aspects in detail in Section 29-1. Descriptive Set Theory is not initially linked to Computability Theory, and it will be mainly Stephen Cole Kleene who will develop the effective aspects [113] [114] [116] [115] between the years 1938 and 1955. These are especially the work of Kleene and his colleagues that we will present in this part.

**For the computability theorist**, Higher Computability Theory is a generalization of the concepts of Classical Computability Theory to computations and constructions in infinite time, via the concept of ordinal. This allows to lift the notion of computation and the intuitions behind it to other definability notions. As this book deals with Computability Theory, it is natural to approach Higher Computability Theory mainly from this prism.

More concretely, many definitions of computability made along integers can be generalized to definitions along ordinals. For example, the arithmetic hierarchy can be extended to ordinal levels, to form the hyperarithmetic hierarchy. We will see a whole correspondence between Classical Computability Theory and Higher Computability Theory, where the hyperarithmetic sets will correspond to finite sets, the  $\Pi_1^1$  sets to computably enumerable sets, and Kleene's  $\mathcal{O}$  to the Turing jump. These concepts will all be defined in this part.

## 1. Motivations

We begin by presenting an example, which starts from notions of Classical Computability Theory seen so far, in order to show its limits as well as the necessity of stronger computation notions to deal with certain questions.

Let us remember Proposition 8-3.6 which states that if a  $\Pi_1^0$  class contains exactly one element  $X$ , then  $X$  is computable. We will call such elements  $\Pi_1^0$  *singletons*. What about  $\Pi_2^0$  singletons? The proof of Proposition 19-1.2 shows that any  $\emptyset'$ -computable set is a  $\Pi_2^0$  singleton. It would be tempting to try to generalize the situation of  $\Pi_1^0$  singletons by conjecturing that the  $\Pi_2^0$  singletons are exactly the  $\emptyset'$ -computable sets, but the situation is in this case much more complex ...

In order to see this, we first present an alternative construction to show that  $\emptyset'$  is a  $\Pi_2^0$  singleton. For this we use (2) from Example 17-3.7: the

class  $\mathcal{S}$  of sets which are Turing jumps of other sets is  $\Pi_2^0$ . The example used the existence of a functional  $\Phi_e$  such that  $\Phi_e(X)$  is total for all  $X$  and such that  $\Phi_e(X') = X$  for all  $X$ . Using  $\Phi_e$  we defined  $\mathcal{S}$  as follows:

$$\bigcap_n \left( \{X : X(n) = 1 \wedge \Phi_n(\Phi_e(X), n) \downarrow\} \cup \{X : X(n) = 0 \wedge \Phi_n(\Phi_e(X), n) \uparrow\} \right)$$

The class  $\mathcal{S}$  is indeed a  $\Pi_2^0$  class, which can be used to easily show that the halting set  $\emptyset'$  is a  $\Pi_2^0$  singleton: it is the only element of the class  $\mathcal{S} \cap \{X : \Phi_e(X) = \emptyset\}$ . We can easily verify that the class  $\{X : \Phi_e(X) = \emptyset\}$  is a  $\Pi_1^0$  class which is therefore according to Lemma 17-3.12 also  $\Pi_2^0$ . The intersection of the two classes is therefore indeed a  $\Pi_2^0$  class. Moreover it is clear from the respective definitions of these classes that the intersection contains only the element  $\emptyset'$ .

We can now continue to show that  $\emptyset''$  is also a  $\Pi_2^0$  singleton. Let  $\mathcal{S}_1$  be the  $\Pi_2^0$  class containing exactly  $\emptyset'$ . Then, we can define  $\mathcal{S}_2$  the  $\Pi_2^0$  class containing exactly  $\emptyset''$  by  $\mathcal{S}_2 = \mathcal{S} \cap \{X : \Phi_e(X) \in \mathcal{S}_1\}$ . We can easily see how to continue with  $\mathcal{S}_{n+1} = \mathcal{S} \cap \{X : \Phi_e(X) \in \mathcal{S}_n\}$ , making each set  $\emptyset^{(n)}$  a  $\Pi_2^0$  singleton.

The  $\Pi_2^0$  singletons can therefore be of arbitrary arithmetic complexity. Note before continuing that the converse is no longer true: there exist  $\emptyset''$ -computable sets which are not  $\Pi_2^0$  singletons. It suffices for example to notice that any  $\Pi_2^0$  singleton is a  $\emptyset'$ -Martin-Löf test, and to consider a 2-random and  $\emptyset''$ -computable set. Arithmetic sets are therefore not all  $\Pi_2^0$  singletons. For their part, are the  $\Pi_2^0$  singletons at least all arithmetic? Here again the situation is not so simple ...

Consider the set  $\emptyset^{(\omega)} = \bigoplus_n \emptyset^{(n)}$ :  $\langle n, m \rangle \in \emptyset^{(\omega)}$  iff  $n \in \emptyset^{(m)}$ . The set  $\emptyset^{(\omega)}$  cannot be arithmetic, i.e.  $\Sigma_n^0$  for some  $n$ . Indeed if  $\emptyset^{(\omega)}$  is  $\Sigma_n^0$  then  $\emptyset^{(n)} \geq_T \emptyset^{(\omega)}$ . Moreover  $\emptyset^{(\omega)} \geq_T \emptyset^{(n+1)}$  and therefore  $\emptyset^{(n)} \geq_T \emptyset^{(n+1)}$  which is a contradiction. If  $\emptyset^{(\omega)}$  is not arithmetic, it is nonetheless a  $\Pi_2^0$  singleton via the following class:

$$S_\omega = \left\{ \bigoplus_n X_n : \forall n \ X_n \in \mathcal{S}_n \right\}.$$

We can of course continue with the sets  $S_{\omega+n+1} = \mathcal{S} \cap \{X : \Phi_e(X) \in \mathcal{S}_{\omega+n}\}$ . How far does it go? This question brings us to the study and the definition of the so-called *hyperarithmetic* sets, which as we will see are exactly the sets which are Turing computable by a  $\Pi_2^0$  singleton.

## 2. Panorama of higher computability

The arithmetic hierarchy admits several characterizations. It is originally defined in terms of the alternation of first-order quantifiers. By Post's famous theorem, the  $\Delta_n^0$  and  $\Sigma_n^0$  sets are respectively the computable and computably enumerable sets using the oracle  $\emptyset^{(n-1)}$ . The arithmetic hierarchy can be extended via this correspondence, using transfinite iterations of the Turing jump.

**Iterations of the Turing jump.** The Turing jump can be seen as an operation on sets, defined by  $X \mapsto X' = \{e : \Phi_e^X(e) \downarrow\}$ . The sets  $\emptyset^{(n)}$  are defined by iterating this operation along the natural numbers, namely

$$\emptyset^{(0)} = \emptyset \text{ and } \emptyset^{(n+1)} = (\emptyset^{(n)})'$$

It is however possible to go beyond the finite iterations of the Turing jump, by defining its  $\omega$ -iteration  $\emptyset^{(\omega)} = \bigoplus_n \emptyset^{(n)}$ , then its  $(\omega+1)$ -iteration  $\emptyset^{(\omega+1)} = (\emptyset^{(\omega)})'$  and so on.

**Ordinals.** The iteration of the Turing jump beyond finite numbers requires extending the notion of natural numbers to transfinite numbers, called *ordinals*. This is a concept initially introduced by Cantor, and which can be seen as an extension of the notion of natural number for which the following principle remains true: any non-empty subset of ordinals admits a smallest element. Standard arithmetic operations are also generalized to ordinals. We will therefore denote by  $0, 1, 2, \dots$  the first ordinals and  $\omega$  the smallest infinite ordinal. In general, we will denote the ordinals with Greek letters:  $\alpha, \beta, \gamma, \dots$

Ordinals can be arbitrarily complex, and most of them are uncountable. We will therefore restrict ourselves, for the needs of Higher Computability Theory, to a sub-collection of the ordinals that we can represent using natural numbers and computability. Kleene defined a set  $\mathcal{O} \subseteq \mathbb{N}$  of codes of ordinals, associating each integer  $n \in \mathcal{O}$  with an ordinal  $|n|$ . Ordinals admitting a code in  $\mathcal{O}$  are called *constructibles*, and form an initial segment of them. We will denote by  $\omega_1^{ck}$  the smallest non-constructible ordinal. These notions will be studied in Chapter 27.

**Hyperarithmetical sets.** Turing jumps can be iterated along constructible ordinals. We can extend the arithmetical hierarchy to ordinal levels, by defining for any constructible ordinal  $\alpha$  the  $\Sigma_\alpha^0$  class of sets which are computably enumerable in  $\emptyset^{(\alpha)}$ . We then obtain the *hyperarithmetical hierarchy*. The correspondence between computability and definability could be formally extended to hyperarithmetical sets, using infinite formulas (once these latter are formally defined). We won't develop further this aspect, but we hope to give sufficient intuition so that the reader feels why this is true.

Hyperarithmetical sets form a relatively natural class, and admit in particular many characterizations. A function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is a *modulus* of a set  $X$  if any function  $g$  dominating  $f$  computes  $X$ . Hyperarithmetical sets are exactly the sets admitting a modulus, or the sets computable by a  $\Pi_2^0$  singleton in  $2^{\mathbb{N}}$ . These notions will be studied in Chapter 28.

**Effective Borel hierarchy.** In the same way that the hyperarithmetical hierarchy classifies sets of natural integers according to their definitional complexity, classes in Cantor space can be classified into effective hierarchies. We had defined the  $\Sigma_n^0$  and  $\Pi_n^0$  classes of real numbers for all  $n \in \mathbb{N}$ . This hierarchy can be extended along constructible ordinals. We obtain the *effective Borel hierarchy*, or the *hyperarithmetical classes* which will be presented in Chapter 28.

**$\Sigma_1^1$  and  $\Pi_1^1$  sets and classes.** Instead of iterating constructions along constructible ordinals, we can extend the arithmetic hierarchy and the effective Borel hierarchy by considering second-order formulas. A set or a class is  $\Sigma_1^1$  if it is definable by a formula of the form  $\exists X F(X)$  where  $F$  is an arithmetic formula in front of which the quantification is done over the sets of integers. In the same way, a set or a class is  $\Pi_1^1$  if it is definable by formula of the form  $\forall X F(X)$  where the quantification is done over the sets of integers. The  $\Delta_1^1$  sets and classes are those which are both  $\Sigma_1^1$  and  $\Pi_1^1$ . The use of second-order quantifiers gives a much higher descriptive power, and in particular, all hyperarithmetical sets and classes are  $\Delta_1^1$ . Kleene showed that the  $\Delta_1^1$  sets and classes are exactly the hyperarithmetical ones, thus giving a new purely definitional characterizations of hyperarithmetical objects without the use of ordinals. These notions will be approached in Section 29-1.

### 3. Correspondence with Classical Computability Theory

The constructions presented in the previous section are mainly definitional, and do not shed light on the computational nature of sets beyond the arithmetic ones. However, there is an informal correspondence between certain concepts of Classical Computability Theory, and those of Higher Computability Theory. Just as the notion of computable set induces Turing reduction by relativization, we define the *hyperarithmetical reduction* by  $X \leq_h Y$  if  $X$  is hyperarithmetical relative to  $Y$ . The correspondence between Classical Computability Theory and Higher Computability Theory is sketched in Section 3:

This correspondence may seem surprising under certain aspects, in particular the fact that the higher computably enumerable sets are the  $\Pi_1^1$  sets

Classical Computability	Higher Computability
Finite or computable	Hyperarithmetic
C.e. sets	$\Pi_1^1$ sets
Halting set $\emptyset'$	Kleene's $\mathcal{O}$
$\Pi_1^0$ classes	$\Sigma_1^1$ classes
Turing reduction	hyperarithmetic reduction

(rather than the  $\Sigma_1^1$  sets). Here are some elements to better understand Section 3:

**Kleene's  $\mathcal{O}$ .** A characteristic property of the halting problem is its  $\Sigma_1^0$ -completeness for many-one reduction. In other words, the halting problem is a form of a computably enumerable universal set. Likewise, Kleene's  $\mathcal{O}$ , defined as the set of codes of constructible ordinals, is  $\Pi_1^1$ -complete for the many-one reduction.

Just like the halting problem  $\{e : \Phi_e(e) \downarrow\}$  induced by its relativization an operator  $X \mapsto \{e : \Phi_e^X(e) \downarrow\}$  called Turing jump, we can define the notion of  $X$ -constructible ordinal, which induces an operator  $X \mapsto \mathcal{O}^X = \{e : e \text{ codes for an } X\text{-constructible ordinal}\}$ .

**$\Pi_1^1$  sets.** If we note  $\mathcal{O}_{<\alpha}$  restriction of Kleene's  $\mathcal{O}$  to codes of ordinals less than  $\alpha$ , for a constructible ordinal  $\alpha < \omega_1^{ck}$ , the set  $\mathcal{O}_{<\alpha}$  is a  $\Sigma_{\alpha+1}^0$  set, therefore hyperarithmetic. We can therefore approximate Kleene's  $\mathcal{O}$  by increasing hyperarithmetic sets  $\mathcal{O}_{<0}, \mathcal{O}_{<1}, \mathcal{O}_{<2}, \dots$ , along constructible ordinals, with  $\mathcal{O} = \bigcup_{\alpha < \omega_1^{ck}} \mathcal{O}_{<\alpha}$ . Kleene's  $\mathcal{O}$  being  $\Pi_1^1$  complete for the many-one reduction, its approximations induce approximations of any  $\Pi_1^1$  set using hyperarithmetic sets along constructible ordinals. The  $\Pi_1^1$  sets are therefore the sets that can be higher computably enumerable.

**$\Sigma_1^1$  classes.** In the same way that a  $\Pi_1^0$  class can be seen as the set of paths of a computable tree in  $2^{<\mathbb{N}}$ , a  $\Sigma_1^1$  class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  can be encoded by a tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ , so that  $\mathcal{B} = \{X : \exists f \in \mathbb{N}^{\mathbb{N}} X \oplus f \in [T]\}$ . We will prove several basis theorems for the  $\Sigma_1^1$  classes which admit a direct correspondence with the basis theorems for the  $\Pi_1^0$  classes. For example, any non-empty  $\Sigma_1^1$  class admits a *hyperlow* member, i.e., a set  $X$  such that  $\mathcal{O}^X \equiv_T \mathcal{O}$ . Moreover, if  $Y$  is a non-hyperarithmetic set, then any non-empty  $\Sigma_1^1$  class contains an element  $X$  such that  $Y \not\leq_h X$ . These two basis theorems are analogues of the low and cone avoidance basis theorems for  $\Pi_1^0$  classes.

# Chapter 27

## Transfinite numbers

We will approach in this chapter a fundamental mathematical concept allowing to carry out iterative constructions beyond the natural numbers, in the sense that the iterations will go beyond reaching the finite stages: they will reach their limit, and will continue beyond. In order to illustrate this mechanism, we will show how the Turing jump can be iterated to infinity and beyond. We will then find ourselves confronted with new problems in dealing with infinite stages, which will find their resolution through the notion of ordinal, a conceptual revolution initiated by Cantor in 1883.

### 1. Motivation: computable iterations of the jump

Let us come back to the construction of the Turing jump iterations. We have an operation  $X \mapsto X' = \{e : \Phi_e^X(e) \downarrow\}$  on the sets, which we iterate from the empty set  $\emptyset$ . Initially,  $\emptyset^{(0)} = \emptyset$ . Then, we define at step 1 the first Turing jump  $\emptyset^{(1)} = (\emptyset^{(0)})'$ , then at step 2 the double jump  $\emptyset^{(2)} = (\emptyset^{(1)})'$ , and so on, by defining at step  $n + 1$  the set  $\emptyset^{(n+1)} = (\emptyset^{(n)})'$ . We also saw in Section 26-1 that the construction did not stop at finite iterations. It is possible to define the  $\omega$ -jump at the so-called “ $\omega$ ” step by  $\emptyset^{(\omega)} = \bigoplus_n \emptyset^{(n)}$ . It is clear that  $\emptyset^{(\omega)} >_m \emptyset^{(n)}$  for all  $n$ , where  $\leq_m$  denotes the many-one reduction (see Section 5-4). The set  $\emptyset^{(\omega)}$  contains the answer to all arithmetic questions, and as we have seen, it is not itself arithmetic. From there, we can continue to iterate the jump: for any  $n$  we define  $\emptyset^{(\omega+n+1)} =$

$(\emptyset^{(\omega+n)})'$  then  $\emptyset^{(\omega+\omega)} = \bigoplus_n \emptyset^{(\omega+n)}$ . There again we will have  $\emptyset^{(\omega+\omega)} >_m \emptyset^{(\omega+n)}$  for all  $n$ . Let us stop in order to identify, for the moment in an informal way, the fundamental stages of this process.

- (1) *Initial stage.* This is the base case: the process has not yet been iterated. In the case of the Turing jump, the zero iteration  $\emptyset^{(0)}$  is simply the set  $\emptyset$  itself.
- (2) *Successor step:* suppose that we have iterated the process for  $\alpha$  steps, obtaining the set  $\emptyset^{(\alpha)}$ . It is always possible to iterate once more, we therefore go to the *successor* step  $\alpha+1$ . In the case of the Turing jump, we define  $\emptyset^{(\alpha+1)} = (\emptyset^{(\alpha)})'$ .
- (3) *Limit step:* given the  $\emptyset^{(\alpha_n)}$  iterations of the jump for  $n \in \mathbb{N}$  and with  $\emptyset^{(\alpha_n)} <_m \emptyset^{(\alpha_{n+1})}$ , we can define its *limit* iteration  $\emptyset^{(\gamma)} = \bigoplus_n \emptyset^{(\alpha_n)}$ .

We call this kind of process a *transfinite* iteration, characterized by successor and limit steps. It is an extension of constructions by induction on the natural numbers, to which the addition of limit steps makes it possible to go beyond the finite case. This new type of steps deserves our attention. The first limit case  $\emptyset^{(\omega)} = \bigoplus_n \emptyset^{(n)}$  is simple, but it is important to notice that within Turing degrees — or even the many-one degrees — there would be many other ways to define  $\emptyset^{(\omega)}$ , for example by taking  $\bigoplus_n \emptyset^{(2^n)}$ . As one advances in the transfinite iterations, there is quickly no longer a canonical choice to select countably many elements allowing to iterate the jump at these limit steps. We then use for that a coding system imagined by Kleene and which will allow us to proceed properly.

### 1.1. Kleene's $\mathcal{O}$

In order to formalize Turing jumps transfinite construction, we will start by naming (or coding) the steps, by assigning them a natural numbers. If one proceeds naively and associates the integer 0 for the initial step, and the integer  $n+1$  for the successor step of step  $n$ , all the natural numbers will be used to name the finite steps, and there will be none left to name the limit ones. We will therefore resort to a more elaborate coding based on the fundamental theorem of arithmetic, which states the uniqueness of the prime factors decomposition of each natural number.

**Definition 1.1 (Kleene [113]).** We inductively define a set of notations  $\mathcal{O} \subseteq \mathbb{N}$  with a partial order  $<_o$  on its elements:

- (1)  $1 \in \mathcal{O}$ .

- (2) If  $a \in \mathcal{O}$  then  $2^a \in \mathcal{O}$ . We then have  $a <_o 2^a$ , and  $b <_o a$  implies  $b <_o 2^a$  for all  $b$ .
- (3) If  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  is a total computable function such that  $\forall n \Phi_e(n) \in \mathcal{O}$  with  $\Phi_e(n) <_o \Phi_e(n+1)$  for all  $n$ , then  $3 \cdot 5^e \in \mathcal{O}$ . Moreover for each  $b \in \mathcal{O}$  such that  $b <_o \Phi_e(n)$  for a some  $n$  we have  $b <_o 3 \cdot 5^e$ .  $\diamond$

Formally, the partial order  $<_o \subseteq \mathbb{N} \times \mathbb{N}$  can be defined as the smallest set containing  $1 <_o 2$  and closed under operations (2) and (3). The set  $\mathcal{O}$  is then the domain of this partial order.

Intuitively, case (1) defines the code for step 0, while cases (2) and (3) correspond respectively to the successor and limit steps. The details of the definition are mainly conventional, the main idea being to obtain a coding system for the successor and limit steps which does not “bite its own tail”: the integers 2, 3 and 5 used are all prime numbers, and the fundamental arithmetic theorem — the uniqueness of numbers prime factors decomposition — guarantees that an element of  $\mathcal{O}$  defined at a limit or successor step will never be equal to a previously constructed code. In particular, in case (3), the use of  $3 \cdot 5^e$  rather than  $3^e$  is simply there to avoid falling back to 1 in the case  $e = 0$ . We might as well have noted this step  $3^{e+1}$ .

Note that in the limit case — case (3) — only the limits of certain increasing sequences of steps admit a code, namely the computable ones. This restriction is necessary, insofar as there is only countably many integers, while there are uncountably many infinite sequences of them. Figure 1.2 illustrates the order  $<_o$  on  $\mathcal{O}$ .

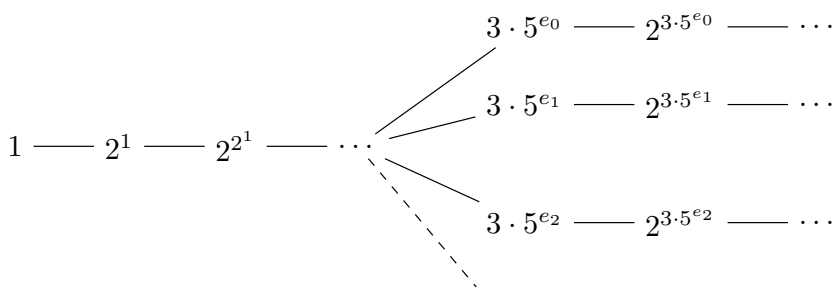


Figure 1.2: Partial order  $<_o$  on  $\mathcal{O}$

### Notation

We will sometimes write  $a \leq_o b$  for elements  $a, b \in \mathcal{O}$  to mean  $a <_o b$  or  $a = b$ . We will sometimes call “Kleene’s  $\mathcal{O}$ ” the set  $\mathcal{O}$  defined above.

Kleene's  $\mathcal{O}$  can now be used to properly conduct transfinite iterations of the jump, traditionally denoted by the letter  $H$ .

**Definition 1.3.** We define the transfinite iterations  $H_a$  of the jump for the elements  $a \in \mathcal{O}$ :

- (1)  $H_1 = \emptyset$
- (2)  $H_{2^a} = H'_a$
- (3)  $H_{3 \cdot 5^e} = \bigoplus_n H_{\Phi_e(n)}$

◇

Note the advantage of using codes: given  $a \in \mathcal{O}$  we can enumerate all the elements  $b \in \mathcal{O}$  such that  $b <_o a$ . Indeed if  $a$  is of the form  $2^b$  one enumerates  $b$  and recursively calls the enumeration on  $b$  and if  $a$  is of the form  $3 \cdot 5^e$  one enumerates  $\Phi_e(n)$  for all  $n$  and recursively calls step by step the enumeration on each  $\Phi_e(n)$ .

Therefore, given a set  $H_a$ , subject to the knowledge of the code  $a$ , we can then enumerate all the pairs  $(H_b, b)$  for  $b <_o a$ , by adding to our enumeration the corresponding set  $H_b$ , which we can find in a computable way from  $H_a$  and  $a$ . We have in particular  $H_b <_T H_a$  for  $b <_o a$ . We could in fact show with a little more work that we have  $H_b <_m H_a$  for  $b <_o a$ . This will be a consequence of Corollary 28-1.14.

### Remark

Any finite iteration of the Turing jump is coded in a unique way using Definition 1.3. Indeed, for all  $n$ ,  $\emptyset^{(n)} = H_a$  with

$$a = \underbrace{2^{2^{\cdot^{2^1}}}}_{n \text{ copies of } 2}$$

On the other hand, there exist several codes of  $\emptyset^{(\omega)}$  because the same computable function has infinitely many Turing codes. Thus, if  $\Phi_e = \Phi_i$  and  $3 \cdot 5^e \in \mathcal{O}$ , then  $3 \cdot 5^i \in \mathcal{O}$  and  $H_{3 \cdot 5^e} = H_{3 \cdot 5^i}$ . As explained previously, if we consider the iterations of Definition 1.3 within the Turing degrees, there are also functions  $\Phi_e$  and  $\Phi_i$  which are not equivalent, but such that  $H_{3 \cdot 5^e} \equiv_T H_{3 \cdot 5^i}$ . For example, if  $3 \cdot 5^e \in \mathcal{O}$  and  $\Phi_i(n) = \Phi_e(n+1)$  for all  $n$ , then  $3 \cdot 5^i \in \mathcal{O}$  and  $H_{3 \cdot 5^e} \equiv_T H_{3 \cdot 5^i}$ .

## 2. Ordinals

We will see that the elements of  $\mathcal{O}$  are used in a way as an extension of integers, which we call *ordinals* or *transfinite numbers*. The first codes of  $\mathcal{O}$

are in fact linearly ordered like the integers:  $1 < 2 < 2^2 < 2^{2^2} < \dots$ ; for this reason we will also say that  $1 \in \mathcal{O}$  is a code of  $\mathbf{0}$ , that  $2 \in \mathcal{O}$  is a code of  $\mathbf{1}$ , etc., and we will write  $|1| = \mathbf{0}$ ,  $|2| = \mathbf{1}$  and in a general way if  $|a| = \mathbf{n}$  then  $|2^a| = \mathbf{n} + 1$ . The notations  $\mathbf{0}, \mathbf{1}, \mathbf{2}, \dots$  and  $\mathbf{n}$  denote the *finite ordinals* and are used to avoid confusion with their corresponding codes.

Now consider a computable function code  $e$  such that  $\Phi_e(n)$  returns  $a_n \in \mathcal{O}$  for all  $n$  and where  $|a_n| = \mathbf{n}$ . Then,  $3 \cdot 5^e$  sort of encodes for a *transfinite number*, the smallest “number” greater than each integer. Let us call this number  $\omega$ . The elements of  $\mathcal{O}$  encoding  $\omega$  are not unique. Consider  $e_1, e_2$  two function codes such that  $|\Phi_{e_1}(n)| = 2\mathbf{n}$  and  $|\Phi_{e_2}(n)| = 2\mathbf{n} + 1$ . The elements  $3 \cdot 5^{e_1}$  and  $3 \cdot 5^{e_2}$  morally code for the same thing:  $\omega$ , the smallest transfinite number greater than each integer. As explained in Remark 1.1, we have  $H_{3 \cdot 5^{e_1}} \equiv_T H_{3 \cdot 5^{e_2}}$ .

We would like a way to be able to formally account for this equivalence. This is precisely what the *ordinals* allow to do. If the concept was introduced by Cantor in 1883, it is however not its definition that we will present immediately, but that of von Neumann imagined in 1923, which has the advantage of being more concrete.

## 2.1. Von Neumann’s approach: ordinals as sets

John von Neumann defines ordinals as sets. Essentially :

### The principle of von Neumann’s ordinal

An ordinal is given by the set of ordinals which precede it, the order on the ordinals then being established by their relation of membership.

An implementation of this principle can be done informally via the following construction.

- (1) The empty set  $\emptyset$  is an ordinal.
- (2) If  $\alpha$  is an ordinal then  $\alpha \cup \{\alpha\} = \{\beta : \beta = \alpha \text{ or } \beta \in \alpha\}$  is the *successor ordinal* of  $\alpha$ .
- (3) Let  $(\alpha_i)_{i \in I}$  be a collection of ordinals having no greater element for the membership relation. Then,  $\bigcup_{i \in I} \alpha_i = \{\beta : \exists i \in I \beta \in \alpha_i\}$  is a *limit ordinal*.

We easily verify that if  $\alpha$  is the set of ordinals preceding it, then  $\alpha \cup \{\alpha\}$  is a set containing exactly  $\alpha$  and all the ordinals preceding  $\alpha$ : it is the successor ordinal of  $\alpha$ . In the same way if each  $\alpha_i$  for  $i \in I$  is the set of ordinals preceding it, then  $\bigcup_{i \in I} \alpha_i$  is indeed a downward-closed set of

ordinals: it contains all the ordinals preceding  $\bigcup_{i \in I} \alpha_i$  in the sense of the membership relation.

Morally “the set” of all ordinals is the smallest set containing  $\emptyset$  and which is closed under (2) and (3) of our informal construction. However, we will quickly see that “the collection” of ordinals is not a set, and we will soon see a more rigorous way of defining them. However, the previous construction contains everything one needs to understand what ordinals are.

**Example 2.1.** The first ordinals are in direct correspondence with the natural numbers. For these reasons, they will be noted as follows.

- $0 = \emptyset$
- $1 = \{0\} = \{\emptyset\}$
- $2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$
- $3 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$
- ...

It is a common practice to identify natural numbers and their corresponding ordinals: it is an “encoding” of the same concept.

The first limit ordinal is given by  $\omega = \{0, 1, 2, \dots\}$ .

We now see the formal definition of ordinals, which will require a little work to come down to the intuition we have given and which we recall here: an ordinal is the set of ordinals preceding it.

**Definition 2.2 (von Neumann).** A set  $\alpha$  is an ordinal if:

- (1)  $\alpha$  is *transitive*:  $\forall \beta \in \alpha, \beta \subseteq \alpha$ .
- (2) The membership relation on  $\alpha$  is a strict total order.
- (3) Any non-empty subset of  $\alpha$  has a smallest element for the membership relation.

We will say that an ordinal  $\alpha$  is smaller than an ordinal  $\beta$ , and we will write  $\alpha < \beta$  if  $\alpha \in \beta$ . ◇

Note that as the membership relation coincides with the order relation, (1) is equivalent to saying that the set  $\alpha$  is downward-closed: for all  $\beta \in \alpha$ , for all  $\gamma < \beta$ ,  $\gamma \in \alpha$ . Let us now see some propositions allowing to clarify the structure of ordinals. The first proposition may seem curious, and is in fact of interest only in the absence of the axiom of foundation (we refer the reader to section Section 9-4 for more details on this subject).

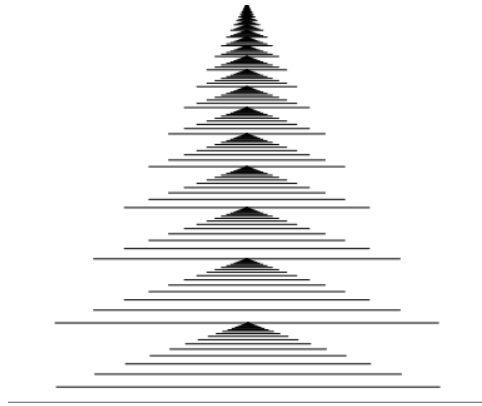


Figure 2.3: Representation of the first ordinals

**Proposition 2.4.** For any ordinal  $\alpha$  we have  $\alpha \notin \alpha$ . ★

PROOF. Suppose  $\alpha \in \alpha$ . Then,  $\alpha$  contains an element (itself,  $\alpha$ ) which belongs to itself. The membership relation is therefore not a strict order on  $\alpha$  since  $\alpha$  contains an element  $x$  such that  $x \in x$  (the order is therefore not strict). So  $\alpha$  is not an ordinal. ■

We now see a first proposition allowing us to return to our intuition.

**Proposition 2.5.** An ordinal  $\alpha$  is always the set of all ordinals smaller than itself. ★

PROOF. By definition if  $\beta < \alpha$  then  $\beta \in \alpha$ . It remains to show that  $\alpha$  only contains ordinals. Let  $x \in \alpha$ . Let  $z \in y \in x$ . Since  $\alpha$  is transitive then  $z, y \in \alpha$ . As the membership relation is an order relation on  $\alpha$  then  $z \in x$ . So  $x$  is transitive. Since  $x \subseteq \alpha$  and the membership relation on  $\alpha$  is a strict total order, then it is also restricted to  $x$ . Finally by transitivity, any non-empty subset of  $x$  is also a non-empty subset of  $\alpha$  and therefore contains a smallest element. So  $x$  is an ordinal. ■

We now show that the ordinals are totally ordered among themselves.

**Theorem 2.6**

*Let  $\alpha_1, \alpha_2$  be two distinct ordinals. Then,  $\alpha_1 < \alpha_2$  or  $\alpha_2 < \alpha_1$ .*

PROOF. Let us first show the following. For all distinct ordinals  $\alpha, \beta$  we have  $\alpha \subseteq \beta$  implies  $\alpha < \beta$ . As  $\alpha \neq \beta$  then  $\beta \setminus \alpha$  is non-empty and has a smallest element  $\gamma$ . By minimality of  $\gamma$  all its elements are elements of  $\alpha$ . We therefore have  $\gamma \subseteq \alpha$ . Now let  $x \in \alpha \subseteq \beta$ . As the membership relation

is total on  $\beta$  we then have  $x \in \gamma$  or  $\gamma \in x$  or  $\gamma = x$ . If  $\gamma = x$  then  $\gamma \in \alpha$  which contradicts the choice of  $\gamma$ . In the same way if  $\gamma \in x \in \alpha$  then by transitivity of  $\alpha$  we would have  $\gamma \in \alpha$  which is also a contradiction. So  $x \in \gamma$ , which implies  $\alpha \subseteq \gamma$  and therefore  $\alpha = \gamma$ , which gives  $\alpha < \beta$ .

Let's move on to the proof. Let  $A = \alpha_1 \setminus \alpha_2$ . If  $A$  is empty then  $\alpha_1 \subseteq \alpha_2$  and therefore  $\alpha_1 < \alpha_2$ . Otherwise  $A$  has a smallest element  $\beta$  such that  $\beta \notin \alpha_2$ . In particular all the elements of  $\beta$  belong to  $\alpha_2$  and therefore  $\beta \subseteq \alpha_2$ . We cannot have by definition  $\beta < \alpha_2$ . So  $\beta = \alpha_2$  and  $\alpha_2 < \alpha_1$ . ■

The following theorem allows us to completely come back to our initial intuition.

**Theorem 2.7**

*A set  $\alpha$  is an ordinal iff it is a downward-closed set of ordinals.*

PROOF. We have seen with Proposition 2.5 that an ordinal  $\alpha$  is always the set of all ordinals smaller than itself. Let  $\gamma \in \beta \in \alpha$ . By transitivity  $\gamma \in \alpha$  therefore  $\alpha$  is downward-closed.

Suppose now that  $\alpha$  is a downward-closed set of ordinals. Let  $\gamma \in \beta \in \alpha$ . Since  $\alpha$  is downward-closed then  $\gamma \in \alpha$ . So  $\alpha$  is transitive. According to Proposition 2.4 the membership relation on  $\alpha$  is anti-reflexive. According to Theorem 2.6 it is total. Since  $\alpha$  only contains ordinals which are transitive sets, the membership relation is also transitive. It is therefore necessarily also anti-symmetrical because if  $x \in y$  and  $y \in x$  then by transitivity  $x \in x$  which contradicts the anti-reflexivity. So the membership relation on  $\alpha$  is a strict total order. Finally, let  $A \subseteq \alpha$  be a non-empty set. Consider  $\beta \in A$ . Either  $\beta$  be the smallest element of  $A$  in which case there is nothing to check, or no, in which case  $A \cap \beta \subseteq \beta$  is non-empty, since  $\beta$  is an ordinal then  $A \cap \beta$  has a smallest element, which is also the smallest element of  $A$ . ■

**Corollary 2.8**

(1) *Let  $\alpha$  be an ordinal. Then,  $\alpha \cup \{\alpha\}$  is an ordinal.*

(2) *Let  $\{\alpha_i\}_{i \in I}$  be a collection of ordinals. Then,  $\bigcup_{i \in I} \alpha_i$  is an ordinal.*

PROOF. If  $\alpha$  is an ordinal then  $\alpha \cup \{\alpha\}$  is a downward-closed set of ordinals. Likewise if  $\{\alpha_i\}_{i \in I}$  a collection of ordinals then  $\bigcup_{i \in I} \alpha_i$  is a downward-closed set of ordinals. ■

### Notation

For an ordinal  $\alpha$  we will write  $\text{succ}(\alpha)$  for  $\alpha \cup \{\alpha\}$ , the successor ordinal of  $\alpha$ . For a collection of ordinals  $(\alpha_i)_{i \in I}$  we will write  $\sup_{i \in I} \alpha_i$  for the ordinal  $\bigcup_{i \in I} \alpha_i$  (which is a limit ordinal if  $(\alpha_i)_{i \in I}$  has no greater element).

Note that any ordinal  $\alpha$  is either  $\emptyset$ , or a successor ordinal, or a limit ordinal. Indeed, if  $\alpha \neq \emptyset$ , then  $\alpha$  has at least one element. If  $\alpha$  has a maximal element  $\beta$ , then  $\alpha = \text{succ}(\beta)$ . Otherwise,  $\alpha = \sup_{\beta \in \alpha} \beta$ . We can therefore reason on the ordinals by case analysis on this trichotomy.

### Notation

We denote by **Ord** “the set” of all ordinals.

The term “set” in the preceding notation has been quoted. The reason is the following:

**Proposition 2.9.** **Ord** is not a set. ★

PROOF. It is easy to see that the class **Ord** of ordinals, equipped with the membership relation  $\in$ , satisfies the properties (1) to (3) of Definition 2.2. We deduce that **Ord** is an ordinal and therefore that  $\text{Ord} \in \text{Ord}$ , which contradicts Proposition 2.4. This contradiction is resolved as follows: **Ord** is not a set and is therefore not subject to Definition 2.2. ■

If **Ord** is not a set, then what is its nature? It is easy to construct a first-order formula  $F(x)$  of Set Theory such that  $F(x)$  is true iff  $x$  is an ordinal. However, we cannot deduce from this that the set of elements which satisfy this formula exists: in order to avoid Russell’s paradox, the axiom of comprehension requires a pre-existing set: for any set  $a$  and any formula  $G(x)$ , the set  $\{x \in a : G(x)\}$  exists, but nothing guarantees the existence of the set  $\{x : G(x)\}$ , and in the case of the formula  $F(x)$  defining the ordinals, the previous proposition shows that the set  $\{x : F(x)\}$  does not exist, under penalty of arriving at a contradiction. In a way, **Ord** is “too big” to be a set. We will then speak of classes (not to be confused with our main use of the term which denotes throughout this book the subsets of  $2^{\mathbb{N}}$ ):

**Definition 2.10.** In Set Theory, a *class* is a collection of elements which satisfy a formula of Set Theory. ◇

We immediately see an example of the use of the notion of class, which will be reused from time to time as part of our manipulation of ordinals.

**Proposition 2.11.** Any non-empty class of ordinals has a smallest element. ★

PROOF. Let  $F(x)$  be a formula of Set Theory. Suppose that for at least one ordinal  $\alpha$  we have  $F(\alpha)$ . Then the set  $A$  of ordinals  $\beta \in \text{succ}(\alpha)$  such that  $F(\beta)$  is true, is non-empty. Since  $\text{succ}(\alpha)$  is an ordinal then  $A$  has a smallest element, which is also the smallest element  $\beta$  such that  $F(\beta)$ . ■

## 2.2. Cantor's approach: well-orders

von Neumann's definition of ordinals became standard. The concept, however, was not invented by von Neumann, but by Cantor [27] who introduced it in 1883 as a continuation of his work on infinity and cardinality. Von Neumann transfinite numbers are convenient in that they are "concrete" objects. We are now going to strip them of this concrete representation in order to reveal their essence. This rise in abstraction was the starting point of Cantor's definition: what characterizes an ordinal is its *order-type*, that is to say the order it represents, up to isomorphism.

**Definition 2.12.** A strict order  $<_R \subseteq A \times A$  on a set  $A$  is *well-founded* if there is no infinite sequence  $(a_n)_{n \in \mathbb{N}}$  such that  $a_{n+1} <_R a_n$ . If moreover  $<_R$  is total we will say that  $<_R$  is a *well-order*. ◇

Let us now see a small lemma which allows us to reconcile the definition of well-order presented above, with the definition of von Neumann ordinals.

**Lemma 2.13.** Let  $<_R \subseteq A \times A$  be a strict order. The following two statements are equivalent:

- (1) There is no infinite sequence  $(x_n)_{n \in \mathbb{N}}$  of  $A$  such that  $x_{n+1} <_R x_n$  for all  $n$ .
- (2) Every non-empty subset  $B \subseteq A$  has a minimal element. ★

PROOF. Let us show (1)  $\rightarrow$  (2) by contraposition. Suppose that there exists a non-empty subset  $B \subseteq A$  having no minimal element, i.e., for all  $a \in B$  there exists  $b \in B$  such that  $b <_R a$ . We can then easily construct an infinite sequence  $(x_n)_{n \in \mathbb{N}}$  of  $B \subseteq A$  such that  $x_{n+1} <_R x_n$ .

Let us show (2)  $\rightarrow$  (1) by contraposition. Suppose that there exists an infinite sequence  $(x_n)_{n \in \mathbb{N}}$  of  $A$  such that  $x_{n+1} <_R x_n$ . Then, this sequence constitutes a subset  $B \subseteq A$  which does not have any minimal element. ■

Via our new vocabulary, we can define von Neumann ordinals as follows:

### von Neumann Ordinals

A von Neumann ordinal is a transitive set on which the membership relation is a well-order.

Cantor defines ordinals as being the equivalence classes of well-orders, via the isomorphism equivalence relation between orders.

**Definition 2.14.** Two orders  $<_1 \subseteq A_1 \times A_1$  and  $<_2 \subseteq A_2 \times A_2$  on  $A_1$  and  $A_2$  respectively are *isomorphic* if there is a bijection  $f : A_1 \rightarrow A_2$  such that  $x <_1 y$  iff  $f(x) <_2 f(y)$ . ◇

We will now show that von Neumann's definition constitutes in a way a canonical representative of these equivalence classes. If an ordinal according to von Neumann is not strictly speaking an order relation, it induces one via the membership relation on the elements it contains, and this order relation is a well-order.

#### Theorem 2.15

*Let  $<_R \subseteq A \times A$  be a well-order on a set  $A$ . Then, there exists a von Neumann ordinal  $\alpha$  and an isomorphism  $f : A \rightarrow \alpha$ , for which we therefore have  $x <_R y$  iff  $f(x) < f(y)$ .*

**PROOF.** We define the function  $f$  on  $A$  by  $f(y) = \{f(x) : x <_R y\}$ . Such a function  $f$  is defined by transfinite induction on the well-order  $<_R$ . We will see the validity of such a definition via Corollary 3.10 in the next section. Suppose by contradiction that there exists  $x \in A$  such that  $f(x)$  is not an ordinal. Since  $<_R$  is a well-order, there is a minimal such  $x$ . In particular for any  $y <_R x$  the set  $f(y)$  is an ordinal. So  $f(x)$  is a set of ordinals. Let us show that  $f(x)$  is downward-closed. Let  $\alpha \in f(y) \in f(x)$ . By definition of  $f$ ,  $\alpha = f(z)$  for  $z <_R y$ . By transitivity of  $<_R$  we have  $z <_R x$  and therefore  $f(z) \in f(x)$ . So  $f(x)$  is downward-closed, and is therefore an ordinal, contradiction. So for all  $x \in A$  the set  $f(x)$  is an ordinal. We show similarly that the set  $\{f(x) : x \in A\}$  is downward-closed and therefore that it is an ordinal that we will call  $\alpha$ .

It is clear by definition that  $x <_R y$  implies  $f(x) \in f(y)$ . Conversely if one does not have  $x <_R y$  then  $x = y$  or  $y <_R x$  (because well-orders are total) in which case  $f(x) = f(y)$  or  $f(y) \in f(x)$ . We therefore do not have  $f(x) \in f(y)$ . Since  $x <_R y$  iff  $f(x) < f(y)$  and  $x \neq y$  implies  $x <_R y$  or  $y <_R x$  then  $f$  is injective. So  $f$  is an isomorphism from  $A$  to its image. ■

### Notation

Given a well-order  $< \subseteq A \times A$  on  $A$  we write  $|\cdot|$  to denote the ordinal  $\alpha$  whose elements order is isomorphic to  $<$ .

Here are some examples on the representation of ordinals by orders on  $\mathbb{N}$ .

#### Example 2.16.

- The ordinal  $\mathfrak{n} \in \omega$  corresponds to the well order  $0 < 1 < \dots < n - 1$ .
- The ordinal  $\omega$  corresponds to the usual order of all integers. To continue we must then change this usual order.
- The ordinal  $\text{succ}(\omega)$  corresponds to the well order  $1 < 2 < 3 < \dots < 0$ .
- The ordinal  $\sup\{\text{succ}(\omega), \text{succ}(\text{succ}(\omega)), \text{succ}(\text{succ}(\text{succ}(\omega))), \dots\}$  corresponds to the well order  $0 < 2 < 4 < 6 < \dots < 1 < 3 < 5 < 7 < \dots$

Note that just as there is not only one code of  $\mathcal{O}$  for the same ordinal, there is also not only one well-order for the same ordinal. For example, the ordinal  $\omega$  also matches the well order  $1 < 0 < 2 < 3 < 4 < \dots$ .

## 3. Induction and transfinite recurrence

Ordinals have several interests. We will see for example in Section 4.1 that they can, using the axiom of choice, serve as a basis for a unified approach to study sets cardinality. They also make it possible to formalize the transfinite iterations. It is this last aspect that we develop in this section.

### 3.1. Example with the Borel hierarchy

Let us remember for example Definition 17-3.1 on the  $\Sigma_n^0$  and  $\Pi_n^0$  classes of the Borel hierarchy.

This definition is in fact incomplete because it does not capture all the Borel classes, of which we give the complete definition here:

**Definition 3.1 (Borel classes).** The collection of Borel classes of  $2^{\mathbb{N}}$  is the smallest collection containing the open classes, and which is closed under complement and countable union. ◇

According to this definition, a countable union  $\bigcup_n \mathcal{B}_n$  where each  $\mathcal{B}_n$  is a  $\Pi_n^0$  class is indeed a Borel class. However if each  $\mathcal{B}_n$  is *properly*  $\Pi_n^0$ , that is

to say not  $\Sigma_n^0$  and in particular not  $\Pi_{n-1}^0$ , then  $\bigcup_n \mathcal{B}_n$  is itself not  $\Sigma_m^0$  for any  $m \in \mathbb{N}$ : it is in fact  $\Sigma_\omega^0$  and it is necessary to use ordinals to capture the possible complexities of Borel classes.

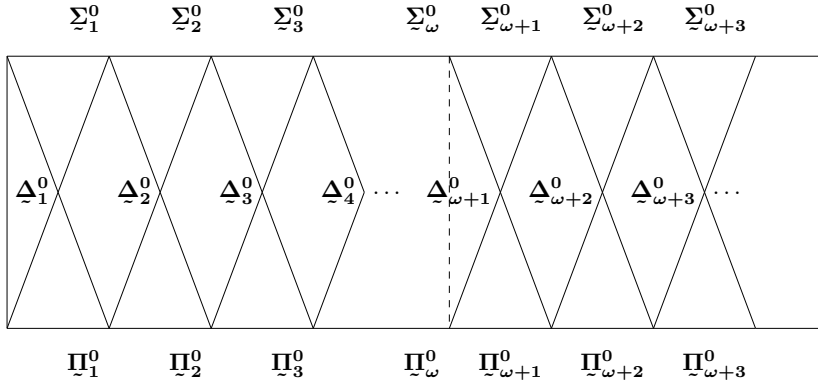


Figure 3.2: Transfinite Borel Hierarchy. Here  $\omega + n$  denotes the  $n$ -th successor of  $\omega$  (see forthcoming Definition 3.5).

**Definition 3.3 (Borel hierarchy).** The Borel hierarchy is defined by induction on the ordinals:

- A  $\Sigma_1^0$  class is an open class of  $2^{\mathbb{N}}$ .
- Let  $\alpha$  be an ordinal. A  $\Pi_\alpha^0$  class is the complement of a  $\Sigma_\alpha^0$  class.
- Let  $\alpha = \text{succ}(\beta)$  be a successor ordinal. A  $\Sigma_\alpha^0$  class is a countable union of  $\Pi_\beta^0$  classes.
- Let  $\alpha$  be a limit ordinal. A  $\Sigma_\alpha^0$  class is a countable union  $\bigcup_n \mathcal{B}_n$  where each  $\mathcal{B}_n$  is a  $\Pi_{\beta_n}^0$  class such that  $\sup_n \beta_n = \alpha$ .  $\diamond$

Note that in the absence of possible ambiguity and in order to remain consistent with the previous notations, we write  $\Sigma_1^0$  and not  $\Sigma_1^0$ . It is possible to shorten the previous definition a bit by gathering the limit and successor cases into a single case, which will sometimes be done in the different hierarchies to come:

**Definition 3.4 (Borel hierarchy (alternative def.)).** The Borel hierarchy is defined by induction on the ordinals:

- A  $\Sigma_1^0$  class is an open class of  $2^{\mathbb{N}}$ .

- Let  $\alpha$  be an ordinal. A  $\Pi_\alpha^0$  class is the complement of a  $\Sigma_\alpha^0$  class.
- Let  $\alpha$  be an ordinal. A  $\Sigma_\alpha^0$  class is a countable union  $\bigcup_n \mathcal{B}_n$  where each  $\mathcal{B}_n$  is a  $\Pi_{\beta_n}^0$  class such that  $\sup_n(\text{succ}(\beta_n)) = \alpha$ .  $\diamond$

Note that the Borel hierarchy is defined above on all ordinals. In practice, we generally assume *the axiom of countable choice* which implies that only countable ordinals are used for the creation of Borel classes. We will see this in more detail in Section 4.

### 3.2. Example with the ordinals arithmetic

We now see another example of definition by induction on ordinals, with the extensions of the usual arithmetic operations. Addition, multiplication and exponentiation are defined in a standard way by induction on integers, by treating the base case, and the successor case. For their extension to ordinals, we add a rule for the limit case. As we saw in Example 2.1, the first ordinals can be identified with natural integers, equipped with their standard order. The following definitions show that ordinals can be seen as an extension of natural numbers. Cases 1 and 2 of the following definitions correspond to the definitions by induction of addition, multiplication and exponentiation on integers. For example,  $n \times 0 = 0$  and  $n \times (m + 1) = (n \times m) + n$  inductively defines multiplication on  $\mathbb{N}$  from addition. Case 3, namely the limit case, allows to “overcome” the barrier of  $\omega$ , and to continue the iteration over arbitrary ordinals.

#### Definition 3.5.

- |  |  |
|--|--|
| 1. $\alpha + 0 = \alpha$   | 1. $\alpha \times 0 = 0$   |
| 2. $\alpha + \text{succ}(\beta) = \text{succ}(\alpha + \beta)$                   | 2. $\alpha \times \text{succ}(\beta) = (\alpha \times \beta) + \alpha$                     |
| 3. $\alpha + \beta = \sup_{\gamma \in \beta} (\alpha + \gamma)$ if $\beta$ limit | 3. $\alpha \times \beta = \sup_{\gamma \in \beta} (\alpha \times \gamma)$ if $\beta$ limit |
| 1. $\alpha^0 = 1$  |  |
| 2. $\alpha^{\text{succ}(\beta)} = (\alpha^\beta) \times \alpha$                  |  |
| 3. $\alpha^\beta = \sup_{\gamma \in \beta} (\alpha^\gamma)$ if $\beta$ limit     | $\diamond$   |

The preceding definitions are not necessarily obvious to understand and the reader can consult Figure 3.6 for a graphical representation of the first ordinals as well as the use of the multiplication and the ordinal power.

Note that addition and multiplication are no longer commutative in ordinals:  $\omega + 2 = \text{succ}(\text{succ}(\omega))$  and  $2 + \omega = \sup_{n \in \omega} (2 + n) = \omega$ .

The key reason that the previous definitions are valid is the fact that the ordinals are well-ordered, and the additional difficulty compared to induction on integers, is the understanding of the limit steps. If we want to perform the operation  $\alpha + \text{succ}(\beta)$  where  $\beta$  is limited, according to the previous definition, we need to know  $\alpha + \beta$ , and to know  $\alpha + \beta$  then we need know  $\alpha + \gamma$  for all  $\gamma < \beta$ , and so on for each of these additions  $\alpha + \gamma$ . Unlike induction on integers, the definition of  $\alpha + \text{succ}(\beta)$  depends here on a *infinitely many* “previous steps”. The addition is nevertheless well defined, because the order on the ordinals is well-founded: if the value of  $\alpha + \beta_1$  needs the value of  $\alpha + \beta_2$  which needs the value of  $\alpha + \beta_3$  and so on for  $\beta_1 > \beta_2 > \beta_3 > \dots$ , we will necessarily arrive at  $\beta_n = 0$  for a certain integer  $n$ .

We will see in the next section a formal justification for definitions by induction on ordinals. Before we get down to it, here is a little exercise to learn how to manipulate the transfinite a bit.

**Exercise 3.7. (★★)** Borks —stupid and wicked creatures— stand in front of you in a single transfinite line indexed by all successor ordinals. There are only  $\omega$  Borks at the moment. You have at your disposal a war hammer with which you must crush each Bork one after the other. The problem is, as soon as you crush one Bork, another one appears at the back of the line. Thus, after crushing Bork number 0, another appears in position  $\omega + 1$ . After crushing Bork number 1, another appears in position  $\omega + 2$ , and so on.

- (1) At what transfinite stage will you have finally wiped out all the Borks?
- (2) Same question, but this time as soon as you crush a Bork,  $\omega$  Borks appear at the back of the line!
- (3) Let’s put your hammer aside and suppose now that at each step a lightning strikes a Bork *at random*. At this point, if a Bork is struck down (which is the case if there is at least one Bork left)  $\omega$  Borks appear at the back of the line. Can we be sure that at a certain transfinite stage, there will be no more Borks? or does it depend on fate? (This last question is difficult.) ◇

### 3.3. Formal justification of ordinal recurrence

Let us start with a generalization of arithmetic induction to ordinals: if a property  $P$  is true on 0 and if the fact that it is true on an integer  $n$  implies that it is true on the integer  $n + 1$ , then it is true on all  $n \in \mathbb{N}$ . The following proposition extends this principle to ordinals, and comes simply from the well-foundedness of their order.

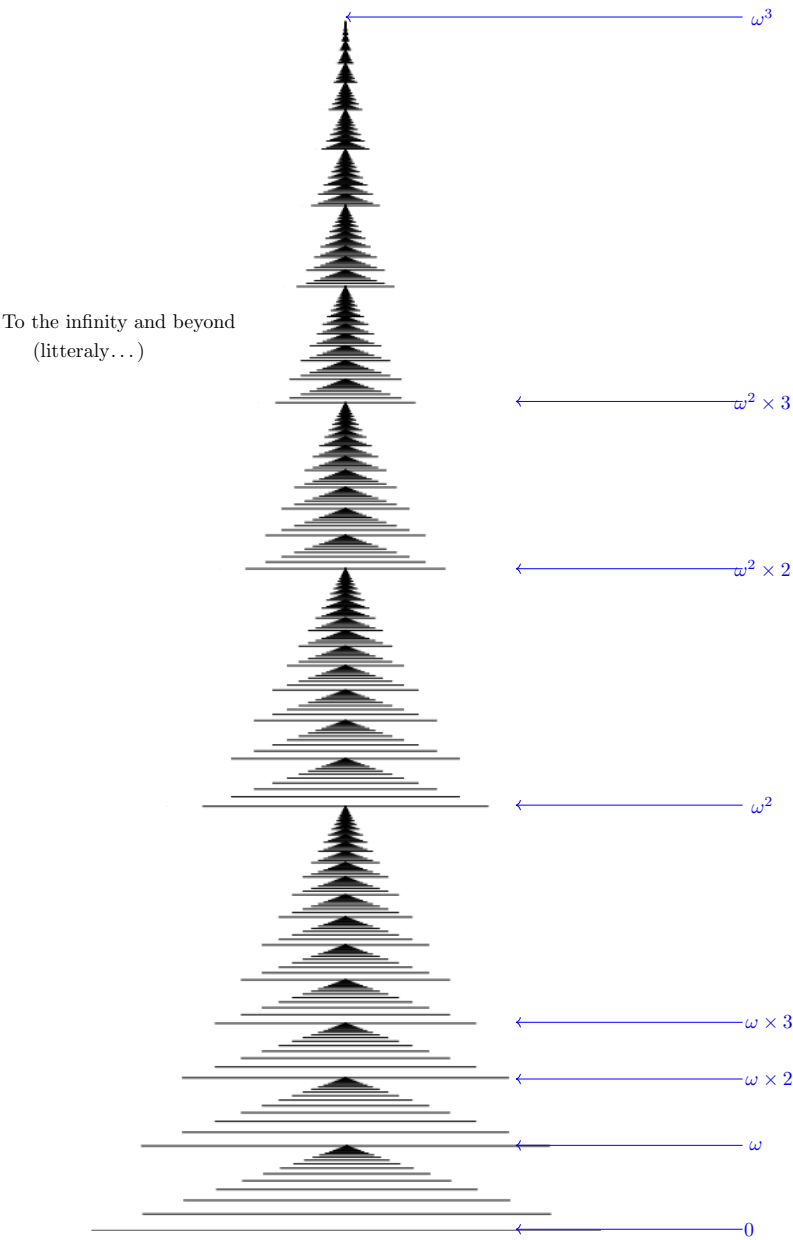


Figure 3.6: Representation of the first ordinals. Each bar represents an ordinal.

**Proposition 3.8 (Transfinite induction over ordinals).** Let  $F(x)$  be a first-order formula, such that for any ordinal  $\alpha$ , if  $F(\beta)$  is true for all  $\beta < \alpha$  then  $F(\alpha)$  is true. Then,  $F(\alpha)$  is true for any ordinal  $\alpha$ . ★

PROOF. Suppose by contradiction that  $F$  is not true of all ordinals  $\alpha$ . According to Proposition 2.11 there exists a smallest ordinal  $\alpha$  such that  $F(\alpha)$  is false. In particular by minimality of  $\alpha$  we have  $F(\beta)$  for all  $\beta < \alpha$ . So  $F(\alpha)$  is true which is a contradiction. ■

We have seen with Definition 3.5 functions defined by recursion on ordinals. As in the case of induction, it is formally an extension of the induction on integers: it is possible to define a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  by reusing for the definition of  $f(n)$  values  $f(0), \dots, f(n-1)$ . Often, only a few preceding values of  $f$  are used to define  $f(n+1)$ , as in the definition of the Fibonacci sequence:  $f(0) = 0, f(1) = 1, f(n+2) = f(n) + f(n+1)$ , but this is not always the case. For example recursive Definition 25-4.22 of Catalan numbers,  $C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$ , uses all previous values. This kind of definition is valid and the general theorem we could formulate is that for any function  $g : \mathbb{N}^{<\mathbb{N}} \rightarrow \mathbb{N}$  there exists a unique function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(n) = g(f \upharpoonright_n)$  for all  $n$ . For example such a function  $g : \mathbb{N}^{<\mathbb{N}} \rightarrow \mathbb{N}$  corresponding to the Fibonacci sequence is given by  $g(\sigma) = 0$  if  $|\sigma| = 0$ ,  $g(\sigma) = 1$  if  $|\sigma| = 1$ , and  $g(\sigma) = \sigma(|\sigma| - 2) + \sigma(|\sigma| - 1)$  otherwise. We see an extension of this theorem, which requires a little work to be proved in all generality on the class of ordinals. To begin with we need to manipulate objects a little more general than functions in order to deal uniformly with induction on all ordinals, and for that we introduce the concept of functional relation:

#### Notation

A functional relation  $G : \mathcal{V} \rightarrow \mathcal{V}$  where  $\mathcal{V}$  is the class of all sets, is simply a first-order formula such that for all  $x$  there exists a unique set  $r$  for which  $G(x, r)$  is true. We will then write  $G(x) = r$ .

Via this new notation, the formal meaning for example of the addition of Definition 3.5 in Set Theory is as follows: for all  $\alpha$  we have a functional relation  $G_\alpha$  such that if  $f$  is a function defined on some ordinal  $\gamma$ , then  $G_\alpha(f) = \alpha$  in case  $\gamma = 0$ ,  $G_\alpha(f) = \text{succ}(f(\gamma'))$  in case  $\gamma = \text{succ}(\gamma')$  and  $G_\alpha(f) = \sup_{\gamma' < \gamma} f(\gamma')$  in case  $\gamma$  is limit. Of course the objective is to use  $G_\alpha$  not on any function  $f$ , but on the function  $f$  which already consists of the addition of  $\alpha$  to  $\gamma'$  for  $\gamma' < \gamma$ . The following theorem guarantees the existence and the uniqueness of this function  $f$  at each step, which gives the validity of the definitions by induction on the class of ordinals.

**Theorem 3.9 (Transfinite recursion over ordinals)**

Let  $\mathcal{V}$  be the class of all sets. Let  $G : \mathcal{V} \rightarrow \mathcal{V}$  be a functional relation. Then, there exists a unique functional relation  $F : \text{Ord} \rightarrow \mathcal{V}$  such that  $F(\alpha) = G(F \upharpoonright_\alpha)$  for every ordinal  $\alpha$ .

PROOF. We define the predicate  $H(\alpha, f)$  as being: “ $\alpha$  is an ordinal,  $f$  is function of domain  $\alpha$  such that for all  $\beta \in \alpha$  we have  $G(f \upharpoonright_\beta) = f(\beta)$ ”.

The relation  $F(\alpha) = r$  is then defined as being: “there exists a function  $f$  such that  $H(\alpha, f)$  and  $G(f) = r$ .”

Let us show first that for any ordinal  $\alpha$ , for any function  $f_1, f_2$  defined on  $\alpha$ , if  $H(\alpha, f_1)$  and  $H(\alpha, f_2)$  then  $f_1 = f_2$ . According to Proposition 3.8 it suffices to show that if this is the case for all  $\beta < \alpha$  then it is the case for  $\alpha$ . Suppose indeed that this is the case for all  $\beta < \alpha$ . Let  $f_1, f_2$  be such that  $H(\alpha, f_1)$  and  $H(\alpha, f_2)$ . By definition of  $H$ , for all  $\beta < \alpha$  we have  $H(\beta, f_1 \upharpoonright_\beta)$  and  $H(\beta, f_2 \upharpoonright_\beta)$  (where  $f \upharpoonright_\beta$  denotes the restriction of  $f$  to the elements of  $\beta$ ). By hypothesis we therefore have  $f_1 \upharpoonright_\beta = f_2 \upharpoonright_\beta$  for all  $\beta < \alpha$ . If  $\alpha$  is limit or equal to  $\emptyset$  we therefore have  $f_1 = f_2$ . If  $\alpha = \text{succ}(\beta)$  then  $f_1(\beta) = G(f_1 \upharpoonright_\beta) = G(f_2 \upharpoonright_\beta) = f_2(\beta)$  and therefore  $f_1 = f_2$ . So for any ordinal  $\alpha$  we have  $H(\alpha, f_1)$  and  $H(\alpha, f_2)$  implies  $f_1 = f_2$ .

Let us show that for any ordinal  $\alpha$  there exists a unique set  $r$  (equal to  $G(F \upharpoonright_\alpha)$ ) such that  $F(\alpha) = r$ . According to Proposition 3.8 it suffices to show that if this is the case for all  $\beta < \alpha$  then it is the case for  $\alpha$ . Suppose indeed that this is the case for all  $\beta < \alpha$ . If  $\alpha = \text{succ}(\beta)$  then there exists a function  $f_\beta$  such that  $H(\beta, f_\beta)$ . Let  $r = G(f_\beta)$ . The function  $f$  defined on  $\alpha$  by  $f(\beta) = r$  and by  $f(\gamma) = f_\beta(\gamma)$  for  $\gamma < \beta$  exists and satisfies  $H(\alpha, f)$ . We have in particular  $F(\alpha) = r$  and this value  $r$  is unique. Now if  $\alpha$  is limit or equal to  $\emptyset$  then for all  $\beta < \alpha$  there exists a unique function  $f_\beta$  such that  $H(\beta, f_\beta)$ . Using the replacement axiom, the set  $\{f : \exists \beta < \alpha H(\beta, f)\}$  exists and according to the previous paragraph the function  $f = \bigcup_{\beta < \alpha} f_\beta$  is well defined, is unique and satisfies  $H(\alpha, f)$ . We then have  $F(\alpha) = r$  for  $r = G(f)$ . ■

From the correspondence between ordinals and well-orders, we can therefore make definitions by transfinite induction on any well-order. In the following corollary the notation  $Y^{<A}$  denote the set of functions defined from initial segments of  $A$  to  $Y$  (initial segments for the well-order relation on  $A$ ).

**Corollary 3.10 (Transfinite recursion)**

Let  $<_R$  be a well-order on a set  $A$ . Let  $Y$  be a set and  $g : Y^{<A} \rightarrow Y$  a function. Then, there exists a unique function  $f : A \rightarrow Y$  such that for

$\text{all } a \in A, f(a) = g(f \upharpoonright_a).$

## 4. Countable and uncountable ordinals

By definition, an ordinal  $\alpha$  is countable if there is a bijection  $f : \alpha \rightarrow \mathbb{N}$ . Note that this is a bijection and not an isomorphism, that is to say that the order is not required to be respected. In an equivalent way an ordinal is countable if it can be represented by a well-order on  $\mathbb{N}$ .

**Proposition 4.1.** Let  $\alpha$  be an ordinal. Then,  $\alpha$  is countable iff there exists  $X \in 2^{\mathbb{N}}$  such that the relation  $<_X \subseteq \mathbb{N} \times \mathbb{N}$  defined by  $a <_X b$  iff  $\langle a, b \rangle \in X$  is a well-order on  $\mathbb{N}$  for which  $|<_X| = \alpha$ . ★

PROOF. Suppose  $\alpha$  countable. Let  $f : \mathbb{N} \rightarrow \alpha$  be a bijection. We define  $X$  by  $\langle a, b \rangle \in X$  iff  $f(a) \in f(b)$ . The order  $<_X$  is by definition isomorphic to that of the elements of  $\alpha$  for the membership relation. We therefore have  $|<_X| = \alpha$ .

Conversely let  $X \in 2^{\mathbb{N}}$  be such that the relation  $<_X$  is a well-order on  $\mathbb{N}$  for which  $\alpha = |<_X|$ . According to Theorem 2.15 we have an isomorphism  $f : \mathbb{N} \rightarrow \beta$  between  $<_X$  and a certain ordinal  $\beta$  which is therefore such that  $|<_X| = \beta$ . By hypothesis  $|<_X| = \alpha$  and therefore  $\alpha = \beta$ . We therefore have a bijection between  $\mathbb{N}$  and  $\alpha$ . ■

We now introduce the smallest uncountable ordinal:

### Notation

Let  $\omega_1 = \{\alpha : \alpha \text{ is a countable ordinal}\}.$

By Theorem 2.7,  $\omega_1$  is an ordinal. On the other hand, it cannot be countable because we would then have  $\omega_1 \in \omega_1$ . We therefore conclude that the set of countable ordinals is itself not countable. Moreover, if  $\alpha$  is another uncountable ordinal we necessarily have  $\omega_1 < \alpha$  since by definition of  $\omega_1$  we do not have  $\alpha \in \omega_1$ . Thus,  $\omega_1$  is the smallest uncountable ordinal. In practice we will work here only with countable ordinals, and therefore representable by a well-order on  $\mathbb{N}$ .

### 4.1. Ordinals and the axiom of choice

This section uses Set Theory notions developed in Section 9-4. The axiom of choice allows us to show that any set can be put in bijection with an ordinal, which has the following consequences:

- Any set is well-orderable: given any set  $A$ , we can build a well-order  $< \subseteq A \times A$  on its elements.

- The cardinality of any set is comparable: this comes directly from the fact that for all ordinals  $\alpha, \beta$ , either  $\alpha \subseteq \beta$  or  $\beta \subseteq \alpha$ .

In fact the axiom of choice is equivalent to the fact that any set is well-orderable. It is not an equivalence of which the father of ordinals and set cardinality was fully aware. Cantor wrote for example (see [12, p. 143]):

*“That it is always possible to put any well-defined set in the form of a well-ordered set is, it seems to me, a law of thought, fundamental, rich in consequences and particularly remarkable for its universality, a law to which I promise to return in a later work.”*

This excerpt from Cantor’s writings illustrates well the important part of intuition that there was in his work. It is finally Zermelo and not Cantor who will come back to “this fundamental law of thought” in a later work, and who will understand that the possibility of being able to well order any set requires an axiom, equivalent to the axiom of choice :

**Theorem 4.2 (Zermelo)**

*The axiom of choice is equivalent to the fact that every set is well-orderable.*

PROOF. Recall for this proof that  $\mathcal{P}(A)$  denotes, for a set  $A$ , the set of its subsets. Suppose we have the axiom of choice. Let  $A$  be a set that we want to order. There exists in particular a choice function  $f$  defined on  $\mathcal{P}(A) \setminus \{\emptyset\}$  and which to each non-empty part  $B \subseteq A$  associates an element of  $B$ . We define by induction on the ordinals  $g(0) = f(A)$  then for any ordinal  $\alpha$  such that  $A \setminus \bigcup_{\beta < \alpha} \{g(\beta)\}$  is not empty we define  $g(\alpha) = f(A \setminus \bigcup_{\beta < \alpha} \{g(\beta)\})$ . By the axiom of comprehension let  $B \subseteq A$  be the subset of the elements of  $A$  equal to  $g(\alpha)$  for a certain ordinal  $\alpha$ . By the replacement axiom let  $\alpha$  be the ordinal supremum of  $g^{(-1)}(a)$  for  $a \in B$ . Then, we necessarily have  $A \setminus B$  empty, otherwise  $g(\text{succ}(\alpha))$  would be sent to an element of  $A \setminus B$  which contradicts the definition of  $B$ . So  $g$  is a bijection from an ordinal  $\alpha$  to  $A$ , through which  $A$  is well-ordered.

Conversely, let us suppose any set can be well-ordered. Let  $(A_i)_{i \in I}$  be a collection of non-empty sets, then we put a well-order on  $\bigcup_{i \in I} A_i$ , and we define the choice function  $f$  by assigning to an element  $i \in I$  the smallest element of  $x \in \bigcup_{i \in I} A_i$  such that  $x \in A_i$ . ■

## 4.2. Countable axiom of choice and Borel classes

Zermelo showed that the axiom of choice was used intuitively and instinctively by many mathematicians. Thus, for example the fact that any vector

space of infinite dimension has a basis, requires the axiom of choice. Regarding the study of sets cardinality, the axiom of choice is necessary to show that two sets always have a comparable cardinality, or for example to show that a countable union of countable sets is countable. This last property is of particular interest to us, and requires only a weak version of the axiom of choice.

It is common today, for set theorists, to work with the axiom of choice restricted to countable collections of sets (which themselves are not necessarily countable), without necessarily admitting the axiom of choice in its globality: “For any collection of sets  $(\mathcal{A}_n)_{n \in \mathbb{N}}$  there exists a choice function from  $n$  to  $\mathcal{A}_n$ ”. The countable version of the axiom of choice makes it possible to show that a countable union of countable sets is countable. In particular, it has the following consequence: if  $(\alpha_n)_{n \in \mathbb{N}}$  is a countable sequence of countable ordinals, then  $\sup_n \alpha_n$  is a countable ordinal. This implies in particular that any Borel class is always  $\Sigma_\alpha^0$  for some countable ordinal  $\alpha$ . This is due to the fact that Borel classes are constructed by countable union of Borel classes of lower rank. So it is impossible to have a sequence of countable ordinals  $(\beta_n)_{n \in \mathbb{N}}$  such that  $\sup_n \beta_n = \omega_1$ . There is therefore no new Borel class at step  $\omega_1$ , and therefore neither at the following steps. Regardless of the axioms used, we will always consider in this book that Borel classes are constructed along countable ordinals, and if by some incongruous chance a sequence of countable ordinals  $(\alpha_n)_{n \in \mathbb{N}}$  with  $\sup_n \alpha_n = \omega_1$  were to take advantage of the absence of this axiom to make its appearance, then a class  $\bigcup_n \mathcal{B}_n$  where each  $\mathcal{B}_n$  would be strictly  $\Sigma_{\alpha_n}^0$ , would not be considered as Borel class.

We now know that the countable version of the axiom of choice does not lead to the counter-intuitive situations that arise with its uncountable version: the axiom of countable choice is, for example, compatible with the fact that any set is measurable (see section 17-4) or has the Baire property (see definition 10-4.9). The subject is beyond the scope of this book, and the reader can consult Patrick Dehornoy’s excellent book on Set Theory [43] for more details.

### 4.3. Ordinals and continuum hypothesis

The ordinal  $\omega_1$  represents in a way “the smallest non-countable well-order”. Does this infinity coincide with that of real numbers? In other words, is there a bijection between  $\omega_1$  and  $2^{\mathbb{N}}$ ? This question can be seen as a formulation of the continuum hypothesis. Cantor’s original question was about subsets of real numbers (if  $\mathcal{A} \subseteq \mathbb{R}$  is uncountable, is there an injection from  $2^{\mathbb{N}}$  into  $\mathcal{A}$ ?). If we accept the axiom of choice, Cantor’s question is then equivalent to whether there exists an injection from  $2^{\mathbb{N}}$  to  $\omega_1$ . Note

that without the axiom of choice it is not clear either that an injection exists from  $\omega_1$  to  $2^{\mathbb{N}}$ :

**Proposition 4.3.** “Morally”, there exists an injection from  $\omega_1$  to  $2^{\mathbb{N}}$ , that is to say there is a function  $f$  which to  $\alpha < \omega_1$  associates a non-empty class of reals  $A_\alpha$  such that  $\beta_1 \neq \beta_2$  implies  $A_{\beta_1} \cap A_{\beta_2} = \emptyset$ . ★

PROOF. By Proposition 4.1 for any countable  $\alpha$  there exists  $X$  such that  $|\langle_X| = \alpha$  where  $\langle_X$  is defined by  $a \langle_X b$  iff  $\langle a, b \rangle \in X$ . We associate to each countable  $\alpha$  the class of reals  $X$  such that  $|\langle_X| = \alpha$ . ■

The preceding proposition is obviously unsatisfactory. We have an injection which associates  $\alpha < \omega_1$  to pairwise disjoint non-empty sets  $A_\alpha$  of real numbers. But how to choose a unique real in each set  $A_\alpha$ ? This is something impossible to do without the axiom of choice, with which, on the other hand, we easily construct an injection from  $\omega_1$  into  $2^{\mathbb{N}}$ .

Recall that a set  $X$  is subpotent (equipotent) to a set  $Y$  if there is an injection (bijection) from  $X$  to  $Y$  (see Chapter 2). We can iterate the definition of  $\omega_1$  as follows.

**Definition 4.4.** Suppose  $\omega_n$  defined for  $n \in \mathbb{N}$ . Let

$$\omega_{n+1} = \{\beta : \beta \text{ is an ordinal (strictly or not) subpotent to } \omega_n\}.$$

We easily show by induction that  $\omega_n$  is not equipotent to  $\omega_{n+1}$ : it is a hierarchy of increasingly large infinities.

Gödel built a model of ZFC — his famous constructible universe — in which  $\omega_1$  is equipotent to  $2^{\mathbb{N}}$ . Cohen showed how, for all  $n$ , to construct a model of ZFC in which  $2^{\mathbb{N}}$  is equipotent to  $\omega_n$ .

Regardless of the axioms used, we will soon see that for a large number of classes of sets, the continuum hypothesis is verified in Cantor’s sense: either the class is countable, or the class contains a perfect closed set and therefore a continuous injection from  $2^{\mathbb{N}}$  to itself. This will be the case in particular for any Borel class.

## 5. Effective ordinals

As there are uncountably many ordinals, they obviously cannot all be described by an algorithm. In this section, we will study two definitions of effective ordinals, based respectively on the von Neumann and Cantor’s approach, and show their equivalence. The class of effective ordinals will serve as a basis for the development of higher computability.

### 5.1. Constructive ordinals

The first approach is from Kleene, and follows von Neumann's idea, defining ordinals inductively in terms of successors and limit ordinals. As we saw in Section 1.1, Kleene defined a set  $\mathcal{O}$  of codes of ordinals based on the uniqueness of prime factorization. We recall here the definition of  $\mathcal{O}$ :

**Definition 5.1 (Kleene [113]).** We inductively define a set of notations  $\mathcal{O} \subseteq \mathbb{N}$  with a partial order  $<_o$  on its elements:

- (1)  $1 \in \mathcal{O}$
- (2) If  $a \in \mathcal{O}$  then  $2^a \in \mathcal{O}$ . We then have  $a <_o 2^a$ , and  $b <_o a$  implies  $b <_o 2^a$  for all  $b$ .
- (3) If  $\Phi_e : \mathbb{N} \rightarrow \mathbb{N}$  is a total computable function such that  $\forall n \Phi_e(n) \in \mathcal{O}$  with  $\Phi_e(n) <_o \Phi_e(n+1)$  for all  $n$ , then  $3 \cdot 5^e \in \mathcal{O}$ . Moreover for each  $b \in \mathcal{O}$  such that  $b <_o \Phi_e(n)$  for a certain  $n$  we have  $b <_o 3 \cdot 5^e$ .  
 $\diamond$

**Definition 5.2.** We naturally assign an ordinal to each code of  $\mathcal{O}$  as follows.

1.  $|1| = \emptyset$
2.  $|2^a| = \text{succ}(|a|)$
3.  $|3 \cdot 5^e| = \sup_n |\Phi_e(n)|$

An ordinal  $\alpha$  is *constructive* if there is a code  $a \in \mathcal{O}$  code such that  $\alpha = |a|$ .  
 $\diamond$

#### Notation

Let  $\alpha$  be an ordinal. We will write  $\mathcal{O}_{<\alpha}$  (resp.  $\mathcal{O}_{=\alpha}$ ) for the set of elements  $a \in \mathcal{O}$  such that  $|a| < \alpha$  (resp.  $|a| = \alpha$ ).

The ordinals are of course not all constructive. As  $\mathcal{O}$  is countable, and as there exists uncountably many countable ordinals, then there exists a smallest non-constructive countable ordinal.

#### Notation

We denote by  $\omega_1^{ck}$  the smallest non-constructive ordinal.

The exponent “ck” of the ordinal  $\omega_1^{ck}$  is the acronym of “Church Kleene” for Alonzo Church and Stephen Cole Kleene who introduced [35] the concept.

Note that no ordinal greater than  $\omega_1^{ck}$  is constructive:

**Exercise 5.3.** Show that constructive ordinals are downward-closed.  $\diamond$

We will gradually see many remarkable properties of the ordinal  $\omega_1^{ck}$ . We will see in particular with Theorem 29-6.1 that it is the smallest undefinable ordinal, for very powerful definability notions.

Constructive ordinals make it possible to define some computable functions by induction on the ordinals. In particular, it is possible to compute an addition function on the codes of constructive ordinals.

**Example 5.4.** We define the total computable addition function  $+_o : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , such that for all  $a, b \in \mathcal{O}$  the function returns  $(a +_o b) \in \mathcal{O}$  with  $|a| + |b| = |a +_o b|$  and  $a \leq_o (a +_o b)$ :

$$\begin{aligned} a +_o 1 &= a \\ a +_o 2^b &= 2^{a+_o b} \\ a +_o 3 \cdot 5^e &= 3 \cdot 5^{f(e,a)} \quad \text{where } f(e,a) \text{ is the code of the function} \\ &\quad \text{such that } \Phi_{f(e,a)}(n) = a +_o \Phi_e(n) \\ a +_o b &= 1 \quad \text{if } b \text{ is not of the form } 1, 2^c \text{ or } 3 \cdot 5^e \end{aligned}$$

Note that the definition of  $+_o$  uses the fixed point theorem in order to reuse the code of  $+_o$  in the function  $f(e, a)$ .

**Exercise 5.5. ( $\star$ )** Show that for all  $a, b \in \mathcal{O}$ ,  $|a| + |b| = |a +_o b|$ .  $\diamond$

## 5.2. Computable ordinals

Kleene's definition of constructive ordinals is in the spirit of von Neumann's: an ordinal is an object which "contains" the ordinals preceding it. This definition being equivalent to that of Cantor in the general case, it is natural to wonder if a definition *à la Cantor* of effective ordinals would not also be useful. We therefore introduce the following definition.

**Definition 5.6.** An ordinal  $\alpha$  is *computable* if it is finite or if  $\alpha = |<|$  for a computable well-order  $< \subseteq \mathbb{N} \times \mathbb{N}$  on  $\mathbb{N}$ .  $\diamond$

A first easy result is the inclusion of constructive ordinals in the computable ones.

**Proposition 5.7.** Constructive ordinals are computable. We can in fact uniformly transform a code  $a \in \mathcal{O}$  into a code  $e$  computing a well-order  $<_e$  such that  $|a| = |<_e|$ .  $\star$

The proof lies in the following two lemmas:

**Lemma 5.8.** Let  $<_R \subseteq \mathbb{N} \times \mathbb{N}$  be a c.e. well-order on  $A \subseteq \mathbb{N}$  with  $|A|$  infinite. Then, there exists a c.e. well-order  $<_S \subseteq \mathbb{N} \times \mathbb{N}$  on  $\mathbb{N}$  such that  $|<_R| = |<_S|$ . ★

PROOF. If  $<_R$  is a computably enumerable relation on  $A$ , then  $A$  is of course also computable enumerable: when  $\langle a, b \rangle$  is enumerated in  $<_R$  we enumerate  $a$  and  $b$  in  $A$ . Note that an enumeration of  $A$  with  $|A|$  infinite naturally induces a computable bijection  $f : \mathbb{N} \rightarrow A$  by defining  $f(n)$  as being the  $n$ -th element enumerated in  $A$  (not taking into account redundancies). We then define  $<_S$  as being  $\langle f^{-1}(a), f^{-1}(b) \rangle$  for each  $\langle a, b \rangle$  enumerated in  $<_R$ . ■

**Lemma 5.9.** Let  $<_R \subseteq \mathbb{N} \times \mathbb{N}$  be a c.e. well-order on  $\mathbb{N}$ . Then,  $<_R$  is computable. ★

PROOF. If  $<_R$  is a total order on  $\mathbb{N}$  we have for any integer  $a, b$  with  $a \neq b$ , either  $a <_R b$  or  $b <_R a$ . So  $\langle b, a \rangle \notin <_R$  iff  $a = b$  or  $\langle a, b \rangle \in <_R$ . The complement of  $<_R$  is therefore also computably enumerable. So  $<_R$  is computable. ■

PROOF OF PROPOSITION 5.7. Suppose that  $a \in \mathcal{O}$  codes for an infinite ordinal. We can enumerate the relation  $<_a \subseteq \mathbb{N} \times \mathbb{N}$  defined as being the relation  $<_o$  restricted to the elements  $b, c <_o a$ , which gives us a c.e. well-order on a subset  $A \subseteq \mathbb{N}$  with  $A$  infinite and such that  $|<_a| = |a|$ . We then apply Lemma 5.8 and Lemma 5.9. ■

The advantages of constructive ordinals over the computable ones are clear: given a code  $e \in \mathcal{O}$  we immediately see if  $e$  codes for a successor ordinal (in which case we can obtain a code for the predecessor ordinal) or if  $e$  codes for a limit ordinal (in which case we can obtain a computable sequence of codes for the ordinals constituting the limit). This makes it possible to make definitions of computable functions by induction on constructive ordinals as with Example 5.4. It is not possible to do the same with computable ordinals. The double jump is for example necessary to know if a computable ordinal is limit or successor. Yet the two concepts coincide.

**Theorem 5.10 (Kleene, Markwald)**

*Computable ordinals are constructive. We can uniformly transform the code  $e$  of a well-founded computable total order  $<_e$  into a code  $a \in \mathcal{O}$  such that  $|<_e| \leq |a|$ .*

The proof of the theorem uses the following lemma.

**Lemma 5.11.** There exists a total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that if  $W_e \subseteq \mathcal{O}$  then  $f(e) \in \mathcal{O}$  with  $\sup_{a \in W_e} |a| \leq |f(e)|$  ★

PROOF. We can consider without loss of generality that  $W_e$  is infinite. In order to be sure of this, we can, for example, list in addition to what is already there all the constructive codes of finite ordinals. We define  $f(e)$  as being  $3 \cdot 5^a$  where  $a$  is such that  $\Phi_a(n)$  returns the finite sum of the  $n$  first distinct codes different from 1 (the code of  $\emptyset$ ), from the ordinals enumerated in  $W_e$ . The finite sum is made via the function  $+_o$  of Example 5.4. Note that we have  $\Phi_a(n) <_o \Phi_a(n+1)$  and therefore  $3 \cdot 5^a \in \mathcal{O}$ . ■

PROOF OF THEOREM 5.10. Let  $<_R \subseteq \mathbb{N} \times \mathbb{N}$  be a well-order computable on  $\mathbb{N}$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the function of the previous lemma. We define a function  $g : \mathbb{N} \rightarrow \mathbb{N}$  which on  $a$  computes the code  $e_a$  of the set  $W_{e_a} = \{g(b) : b <_R a\}$  and returns  $f(e_a)$ . Note that  $g$  uses the fixed point theorem.

Let's see what the function  $g$  gives on the first element of  $<_R$ . Let  $a$  be the smallest element of  $<_R$ . Then,  $W_{e_a} = \emptyset$  and we have  $f(e_a) \in \mathcal{O}$  according to the previous lemma and therefore  $g(a) \in \mathcal{O}$ . Starting from this initial case we easily show by induction that for any  $a \in \mathbb{N}$  we have  $g(a) \in \mathcal{O}$  with  $|<_R \upharpoonright_{X_a}| \leq |g(a)|$  where  $X_a$  is the set of elements smaller than  $a$  via  $<_R$ .

We then easily compute the code  $e$  of the function which enumerates  $g(a)$  for all  $a \in \mathbb{N}$  and we return  $f(e)$ . We then have  $|<_R| \leq |f(e)|$ . As constructive ordinals are downward-closed, the ordinal  $|<_R|$  is therefore constructive. ■

Note that we have no effective conversion transforming the code of a computable ordinal  $<_R$  into a code  $a$  of a constructive ordinal such that  $|<_R| = |a|$ . We have instead only  $|<_R| \leq |a|$  and it is possible to show that the equality cannot be uniformly obtained.

**Exercise 5.12. (★)** Show the existence of a total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(n)$  always codes for a computable ordinal via a relation  $<_{f(n)}$ , such that  $n \in \emptyset' \rightarrow |<_{f(n)}| = \omega + 1$  and such that  $n \notin \emptyset' \rightarrow |<_{f(n)}| = \omega$ . Deduce from this the impossibility of uniformly transforming a computable ordinal code into a constructive code for the same ordinal. ◇

### 5.3. Representation of ordinals by trees

There is another usual representation of countable ordinals, easy to handle and very useful in many cases: via trees in the Baire space  $\mathbb{N}^{\mathbb{N}}$ , that was introduced in Section 8-7.

We recall that the manipulation of trees and strings in the Baire space is analogous to that of the Cantor space, for which we use the same notations. Two new ones are introduced, which will be used from time to time in what follows:

### Notation

Let  $T$  be a tree with  $\sigma \in T$ .

- (1)  $T \upharpoonright_\sigma$  denotes the tree of nodes of  $T$  comparable with  $\sigma$ .
- (2)  $T \downharpoonright_\sigma$  denotes the tree  $\{\tau \in \mathbb{N}^{<\mathbb{N}} : \sigma\tau \in T\}$ , i.e., the tree  $T \upharpoonright_\sigma$  “trimmed” so as to have  $\sigma$  as the root.

**Definition 5.13.** A tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  is *well-founded* if it does not contain infinite paths, that is, if  $[T]$  is empty. For a well-founded tree  $T$  and  $\sigma \in T$  we define by induction  $|\sigma| = \sup\{|\sigma n| + 1 : n \in \mathbb{N}, \sigma n \in T\}$ . Finally, we define  $|T| = |\epsilon|$  where  $\epsilon$  is the root of  $T$ . We will sometimes say that  $|T|$  is the *height* of  $T$ . ◇

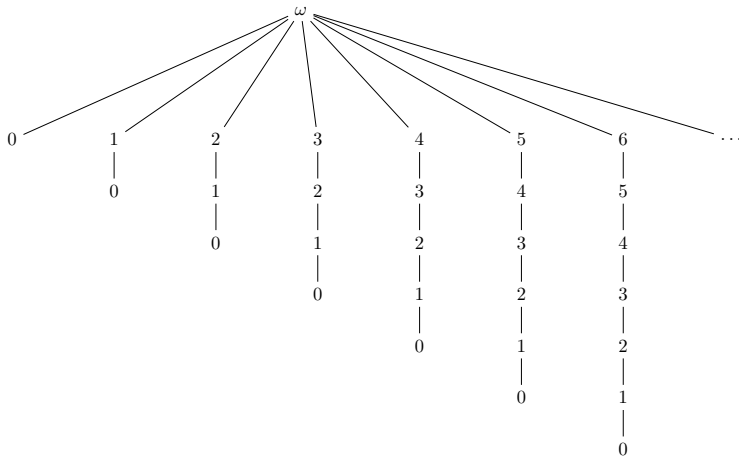


Figure 5.14: Representation of the ordinal  $\omega$  by a tree  $T$ . We wrote for each node  $\sigma$  its corresponding ordinal, that is,  $|T \upharpoonright_\sigma|$ .

Note that the value  $|\sigma|$  of the node  $\sigma$  of a well-founded tree  $T$  depends on  $T$ . When there is a possible ambiguity we will use instead  $|T \upharpoonright_\sigma|$  which equals  $|\sigma|$  and explicitly mentions  $T$ .

For an ill-founded tree  $T$ , the value  $|T|$  is not defined, but we will sometimes use it by abuse of notation, considering it greater than all the ordinals.

**Notation**

We denote by  $\mathcal{T}$  the set of c.e. codes of well-founded trees. We denote by  $\mathcal{T}_{<\alpha}$  (resp.  $\mathcal{T}_{=\alpha}$ ) the set of c.e. codes of trees  $T \in \mathcal{T}$  such that  $|T| < \alpha$  (resp. such that  $|T| = \alpha$ ).

Note that the nodes of  $T$  form a well-founded order via their suffix relation:  $\sigma < \tau$  iff  $\tau \prec \sigma$ . Via this order  $<$  Definition 5.13 is equivalent to

$$|\sigma| = \sup\{|\tau| + 1 : \tau < \sigma, \tau \in T\}.$$

If this order is not total, it nevertheless makes it possible to represent ordinals. We will sometimes need to “make it total”. This can be done via the Kleene-Brouwer ordering. We define  $\sigma < \tau$  for  $\sigma, \tau \in T$  iff  $\tau \prec \sigma$  or when  $\sigma, \tau$  are not prefixes of one another, if  $\sigma$  is lexicographically smaller than  $\tau$ . We can easily check that the order we obtain this way is total on the nodes of  $T$ , and that it is a well-order when  $T$  is well-founded.

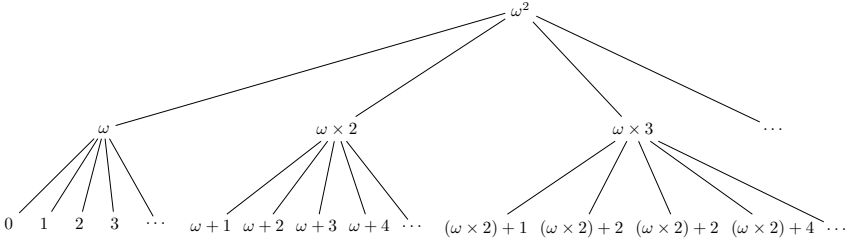


Figure 5.15: Representation of the ordinal  $\omega^2$  with the Kleene-Brouwer ordering on trees

Let us define formally the ordinal associated with the nodes of a tree via the Kleene-Brouwer ordering.

**Definition 5.16.** For a well-founded tree  $T$  the Kleene-Brouwer ordering  $<_{\text{KB}}$  is defined on the elements of  $T$  by  $\sigma <_{\text{KB}} \tau$  if  $\tau \prec \sigma$  or when  $\sigma, \tau$  are not prefixes of one another, if  $\sigma$  is lexicographically smaller than  $\tau$ . For a node  $\sigma \in T$  we define by induction:

$$|\sigma|_{\text{KB}} = \sup \{ (|\tau|_{\text{KB}} + 1) : \tau <_{\text{KB}} \sigma, \tau \in T \}.$$

We finally define  $|T|_{\text{KB}} = |\epsilon|_{\text{KB}}$  where  $\epsilon$  is the root of  $T$ . ◇

**Exercise 5.17. (★)** Show by induction that we always have  $|T| \leq |T|_{\text{KB}}$  for a well-founded tree  $T$ . ◇

Let us now show that the c.e. trees are representations of computable ordinals.

**Proposition 5.18.** An ordinal is computable iff it is equal to  $|T|$  for a c.e. tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ . We can uniformly transform the code of a well-order  $<_R$  into a code of a c.e. tree  $T$  such that  $|<_R| \leq |T|$ , and conversely, we can uniformly transform the code of a c.e. tree  $T$  into a code of well-order  $<_R$  such that  $|T| \leq |<_R|$ . ★

PROOF. Let  $<_R$  be a well-order computable on  $\mathbb{N}$ . We enumerate in  $T$  all the nodes  $n$  for  $n \in \mathbb{N}$ . Then inductively for each node of the form  $\sigma a$  enumerated in  $T$ , we enumerate as children of  $\sigma a$  all the nodes  $\sigma ab$  for  $b$  such that  $b <_R a$ . The tree  $T$  thus defined is well-founded because an infinite path of  $T$  would induce an infinite sequence  $\cdots <_R a_2 <_R a_1 <_R a_0$  which cannot happen because  $<_R$  is a well-order. We easily check by induction that  $|<_R| \leq |T|$ .

For the converse it suffices to consider the Kleene-Brouwer ordering on the nodes of  $T$ . By Exercise 5.17 we have  $|T| \leq |T|_{\text{KB}}$  and via an encoding of the elements of  $\mathbb{N}^{<\mathbb{N}}$  we enumerate a well-order  $<_R$  on a subset  $A \subseteq \mathbb{N}$  such that  $|T|_{\text{KB}} = |<_R|$ . Lemma 5.8 and Lemma 5.9 allow us to conclude. ■

Note that following this same principle, an ordinal is countable iff it is equal to  $|T|$  for a well-founded tree  $T$ , but not necessarily c.e.

## 6. Relativization

The notions of constructible and computable ordinals can be relativized to an oracle.

**Definition 6.1.** Let  $X \in 2^{\mathbb{N}}$ . We define the set  $\mathcal{O}^X$  as follows.

1.  $1 \in \mathcal{O}^X$  with  $|1| = \emptyset$
2. If  $a \in \mathcal{O}^X$  then  $2^a \in \mathcal{O}^X$  with  $|2^a| = \text{succ}(|a|)$ . Moreover  $a <_o 2^a$ , and  $b <_o a$  implies  $b <_o 2^a$  for all  $b$ .
3. If  $\Phi_e$  is a total functional over  $X$  with  $\Phi_e(X, n) \in \mathcal{O}^X$  and  $\Phi_e(X, n) <_o \Phi_e(X, n+1)$  for all  $n$  then  $3 \cdot 5^e \in \mathcal{O}^X$  with  $|3 \cdot 5^e| = \sup_n |\Phi_e(X, n)|$ . Moreover for all  $a$  if  $a <_o \Phi_e(X, n)$  for some  $n$  then  $a <_o 3 \cdot 5^e$ .

An ordinal  $\alpha$  is *X-constructive* if there exists  $a \in \mathcal{O}^X$  such that  $\alpha = |a|$ . ◇

Given  $X$  fixed, the order  $<_o$  on the elements of  $\mathcal{O}^X$  is of course not the same as the order  $<_o$  on the elements of  $\mathcal{O}$ . So we will sometimes write  $<_o^X$  to remove any ambiguity when necessary, just as we will sometimes write  $|a|^X$  instead of  $|a|$  for an ordinal  $a \in \mathcal{O}^X$ .

**Definition 6.2.** An ordinal  $\alpha$  is  $X$ -computable if it is finite or if there is an  $X$ -computable well-order  $<_R \subseteq \mathbb{N} \times \mathbb{N}$  on  $\mathbb{N}$  such that  $|<_R| = \alpha$ .  $\diamond$

**Notation**

We denote by  $\mathcal{T}^X$  the set of  $X$ -c.e. codes of well-founded trees in Baire space.

The different equivalences seen so far can easily be relative to an oracle  $X$ .

**Theorem 6.3**

*An ordinal is  $X$ -constructive iff it is  $X$ -computable iff it is equal to  $|T|$  for an  $X$ -c.e. tree  $T \in \mathcal{T}^X$ .*

Let us remember now the symbol  $\omega_1^{ck}$  denoting the smallest non-constructible — or equivalently non-computable — ordinal. Here again the notion is relativized.

**Notation**

We denote by  $\omega_1^X$  the smallest non  $X$ -computable ordinal.

Note that if  $\omega_1^X$  can be greater than  $\omega_1^{ck}$  it still remains countable. Conversely, any countable ordinal is computable for a certain  $X$ , by considering for example an oracle which codes for an order relation on  $\mathbb{N}$  representing the ordinal. This gives us:

**Proposition 6.4.** The ordinal  $\omega_1$  is the supremum over the oracles  $X$ , of the  $X$ -computable ordinals.  $\star$

# Chapter 28

## Hyperarithmetic sets

Just like one can extend  $\Sigma_n^0$  and  $\Pi_n^0$  classes to any countable ordinal  $\alpha$ , and obtain  $\Sigma_\alpha^0$  and  $\Pi_\alpha^0$  classes, in order to fully define the Borel hierarchy, one can also iterate the arithmetic hierarchy along the computable ordinals, producing a new class of sets called *hyperarithmetic sets*. We will see through this chapter and Chapter 29 that this class is extremely robust, in the sense that it admits many characterizations, using either Turing jump iterations, or  $\Pi_2^0$  singletons, or modulus.

This chapter will require the development of a coding system arsenal, for effective Borel classes and sets of integers, using natural numbers. The proofs will often call for somewhat laborious manipulations of these encodings, but the reader will be rewarded by the addition of hyperarithmetic sets to its toolbox. This development will culminate with two results: Corollary 1.14 which affirms that the sets  $H_a$  for  $a \in \mathcal{O}$  — in other words the  $\alpha$ -iterations of the Turing jump of  $\emptyset$  — are complete at the different levels of the Kleene hierarchy, and Theorem 4.4 which characterizes the  $\Sigma_\alpha^0$  classes of the effective Borel hierarchy in terms of the  $\alpha$ -iterations of the Turing jump.

### 1. Kleene hierarchy

We generalize Definition 5-1.1 of arithmetic sets as follows.

**Definition 1.1.** Kleene’s hyperarithmetic hierarchy is defined by induction on ordinals:

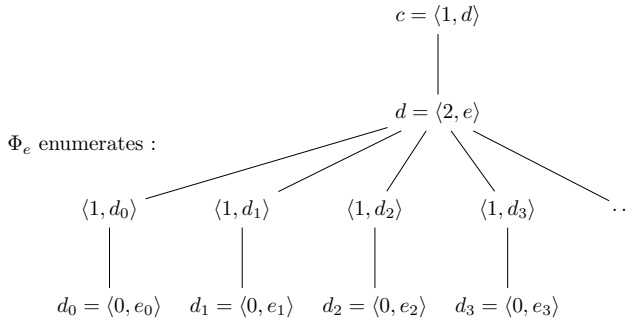


Figure 1.2: The unfolding along the tree corresponding to a  $\Pi_2^0$ -code  $c$ , which codes for the complement of the  $\Sigma_2^0$  set coded by  $\langle 2, e \rangle$  where  $\Phi_e$  enumerates the  $\Pi_1^0$ -codes corresponding themselves to complements of the  $\Sigma_1^0$  sets coded by each  $e_i$ .

- A  $\Sigma_1^0$ -code is given by a pair  $\langle 0, e \rangle$ . The corresponding set  $A \subseteq \mathbb{N}$  is given by  $A = W_e$ .
- A  $\Pi_\alpha^0$ -code is given by a pair  $\langle 1, e \rangle$  where  $e$  is a  $\Sigma_\alpha^0$ -code. The corresponding set  $A \subseteq \mathbb{N}$  is given by  $A = \mathbb{N} \setminus B$  where  $B$  is the set corresponding to the code  $e$ .
- A  $\Sigma_\alpha^0$ -code is given by a pair  $\langle 2, e \rangle$  where  $W_e$  is non-empty and enumerates  $\Pi_{\beta_n}^0$ -codes for  $\beta_n < \alpha$ , with  $\sup_n(\beta_n + 1) = \alpha$ . The corresponding set  $A \subseteq \mathbb{N}$  is given by  $\bigcup_n A_n$  where  $A_n$  is the set corresponding to the  $n$ -th code enumerated by  $W_e$ .  $\diamond$

We say that a set  $A$  is  $\Sigma_\alpha^0$  (resp.  $\Pi_\alpha^0$ ) if it corresponds to a  $\Sigma_\alpha^0$ -code  $e$  (resp. a  $\Pi_\alpha^0$ -code  $e$ ). We say that  $A$  is  $\Delta_\alpha^0$  if it is both  $\Sigma_\alpha^0$  and  $\Pi_\alpha^0$ . Finally, we say that a set  $A$  is *hyperarithmetical* if it is  $\Sigma_\alpha^0$  for some ordinal  $\alpha$ .

Let us remember the sets  $H_a$  of Definition 27-1.3, corresponding to the iterated versions of the jump for  $a \in \mathcal{O}$ . For any  $a \in \mathcal{O}$  such that  $|a| = n \in \mathbb{N}$ , the set  $H_a = \emptyset^{(n)}$  is  $\Sigma_n^0$ . As for the set  $H_a$  with  $a = 3 \cdot 5^e$  such that  $|a| = \omega$ , it is  $\Delta_\omega^0$ . Indeed we have  $\langle n, m \rangle \in H_a$  iff  $n \in H_{\Phi_e(m)}$  and therefore also  $\langle n, m \rangle \notin H_a$  iff  $n \notin H_{\Phi_e(m)}$ . The sets  $\{\langle n, m \rangle : n \in H_{\Phi_e(m)}\}$  and  $\{\langle n, m \rangle : n \notin H_{\Phi_e(m)}\}$  are  $\Delta_1^0(H_{\Phi_e(m)})$  uniformly in  $m$  and therefore  $\Pi_{|\Phi_e(m)|+1}^0$  uniformly in  $m$  according to Theorem 5-5.5. This allows us to make a  $\Sigma_\omega^0$  definition of  $H_a$  as well as of its complement. As for the set  $H_{2^a}$ , it is  $\Sigma_\omega^0$ , and this keeps iterating smoothly on the elements of  $\mathcal{O}$ . To be completely rigorous, however, we will need some manipulation lemmas to deal with  $\Sigma_\alpha^0$  and  $\Pi_\alpha^0$  codes, starting with a generalization of Lemmas 5-1.7 and 5-1.6.

**Lemma 1.3.** The  $\Sigma_\alpha^0$  and  $\Pi_\alpha^0$  sets are closed under unions and finite intersections. ★

PROOF. We can recursively use the following equalities for any sets  $(A_n)_{n \in \mathbb{N}}$ ,  $(B_n)_{n \in \mathbb{N}}$  and  $A, B$ :

$$\begin{aligned} (\bigcup_{n \in \mathbb{N}} A_n) \cap (\bigcup_{n \in \mathbb{N}} B_n) &= \bigcup_{n_1, n_2 \in \mathbb{N}} (A_{n_1} \cap A_{n_2}) \\ (\bigcup_{n \in \mathbb{N}} A_n) \cup (\bigcup_{n \in \mathbb{N}} B_n) &= \bigcup_{n \in \mathbb{N}} C_n \text{ avec } C_{2n} = A_n \text{ and } C_{2n+1} = B_n \\ A \cap (\bigcup_{n \in \mathbb{N}} B_n) &= \bigcup_{n \in \mathbb{N}} A \cap B_n \\ A \cup (\bigcup_{n \in \mathbb{N}} B_n) &= \bigcup_{n \in \mathbb{N}} C_n \text{ avec } C_0 = A \text{ and } C_{n+1} = B_n \\ A^c \cap B^c &= (A \cup B)^c \\ A^c \cup B^c &= (A \cap B)^c \end{aligned}$$

Using the fixed point theorem, we create the computable functions of union and intersection of two codes, which propagate their application using the above relations. ■

**Lemma 1.4.** The  $\Sigma_\alpha^0$  sets are stable under uniform countable unions. ★

PROOF. Given the  $\Sigma_\alpha^0$ -codes  $(\langle 2, e_n \rangle)_{n \in \mathbb{N}}$  it suffices to create the code  $\langle 2, e \rangle$  where  $e$  enumerates  $\bigcup_n W_{e_n}$ . ■

We end with two code manipulation lemmas that will be useful from time to time.

**Lemma 1.5.** Let  $x \in \mathbb{N}$ . There is a computable function which on the  $\Sigma_\alpha^0$  (resp.  $\Pi_\alpha^0$ ) code of a set  $A$  returns the  $\Sigma_\alpha^0$  (resp.  $\Pi_\alpha^0$ ) code of a set  $B$  such that  $B = \mathbb{N}$  if  $x \in A$  and  $B = \emptyset$  otherwise. ★

PROOF. It suffices to “unfold” the code and replace each leaf corresponding to a set  $W_k$  by a c.e. set equal to  $\mathbb{N}$  if  $x \in W_k$  and equal to  $\emptyset$  otherwise.

The procedure works trivially for  $\Sigma_1^0$  codes. Suppose the procedure works for  $\Sigma_\alpha^0$  codes. Let  $a = \langle 1, b \rangle$  be a  $\Pi_\alpha^0$  code of a set  $A = B^c$  where  $b$  is a code of  $B$ . Then, the procedure applied to  $b$  returns  $\mathbb{N}$  if  $x \in B$  and  $\emptyset$  otherwise, which applied to  $a$  goes to the complement and gives as desired  $\mathbb{N}$  if  $x \in A$  and  $\emptyset$  otherwise. We similarly check by induction that we have the expected result with the codes of the form  $\langle 2, b \rangle$  corresponding to unions. ■

**Lemma 1.6.** Let  $A$  be a  $\Sigma_\alpha^0$  set and  $f$  a computable function. Then,  $f(A)$  is a  $\Sigma_\alpha^0$  set. ★

PROOF. According to Lemma 1.5 we can create for any  $x$  the  $\Sigma_\alpha^0$  code of a set  $B_x$  equal to  $\mathbb{N}$  if  $x \in A$  and equal to  $\emptyset$  otherwise.

We have  $f(A) = \bigcup_{x \in \mathbb{N}} \{f(x)\} \cap B_x$ . According to Lemma 1.3 the set  $\{f(x)\} \cap B_x$  is  $\Sigma_\alpha^0$  uniformly in  $x$ . According to Lemma 1.4 the set  $f(A)$  is therefore also  $\Sigma_\alpha^0$ . ■

We now have all the tools necessary to attack the complexity of the sets resulting from the Turing jump transfinite iterations.

**Proposition 1.7.** For any ordinal  $\alpha \geq \omega$  and any  $a \in \mathcal{O}_{=\alpha}$ , the set  $H_{2^a}$  is  $\Sigma_\alpha^0$  uniformly in  $a$ . ★

PROOF. We prove the proposition by induction on the elements of  $\mathcal{O}$ . Induction begins with the sets  $H_a$  for finite  $|a|$  which are all  $\Sigma_{|a|}^0$  according to Proposition 5-5.3. We will see in the rest of the proof (as it was sketched in the paragraph following Definition 1.1) that this leads to have  $H_{2^a}$  sets to be  $\Sigma_{|a|}^0$  for  $a \in \mathcal{O}_{=\omega}$ . Suppose the proposition is true for all  $b <_o a$  and let us show that  $H_a$  is  $\Delta_{|a|}^0$ .

If  $a = 2^b$  codes for a successor ordinal then by induction hypothesis the set  $H_a$  is  $\Sigma_{|b|}^0$ . So it and its complement are  $\Delta_{|a|}^0$ . If  $a = 3 \cdot 5^e$  codes for a limit ordinal, we have  $\langle n, m \rangle \in H_a$  iff  $n \in H_{\Phi_e(m)}$  as well as  $\langle n, m \rangle \notin H_a$  iff  $n \notin H_{\Phi_e(m)}$ . The set  $H_{\Phi_e(m)}$  being by induction uniformly  $\Sigma_{|\Phi_e(m)|}^0$  we have according to Lemma 1.6 applied to each  $H_{\Phi_e(m)}$ , a  $\Sigma_{|a|}^0$ -code of  $H_a$  as well as of its complement. So  $H_a$  is a  $\Delta_{|a|}^0$  set.

In both cases  $H_a$  is a  $\Delta_{|a|}^0$  set. Moreover we have  $n \in H_{2^a}$  iff  $\Phi_n(H_a, n) \downarrow$ , which is a  $\Sigma_1^0(H_a)$  condition. As the set  $H_a$  and its complement are  $\Delta_{|a|}^0$  then the set  $H_{2^a}$  is  $\Sigma_{|a|}^0$  via the predicate  $n \in H_{2^a}$  iff

$$\exists \sigma \exists t \Phi_n(\sigma, n)[t] \downarrow \wedge \forall i < |\sigma| ((\sigma(i) = 0 \wedge i \notin H_a) \vee (\sigma(i) = 1 \wedge i \in H_a)).$$

How to actually get a  $\Sigma_{|a|}^0$ -code for  $H_{2^a}$  from the above formula is not necessarily clear and we therefore detail how to do this. Given a string  $\sigma$  and an integer  $i < |\sigma|$  we can uniformly compute the code  $e_{\sigma, i}$  of a  $\Sigma_{|a|}^0$  set equal to  $\mathbb{N}$  if  $\sigma(i) = 0$  and  $i \notin H_a$  or if  $\sigma(i) = 1$  and  $i \in H_a$  and equal to  $\emptyset$  otherwise: it suffices to apply Lemma 1.5 to  $i$  and  $\mathbb{N} - H_a$  if  $\sigma(i) = 0$  and to  $i$  and  $H_a$  if  $\sigma(i) = 1$ . Let  $B_{e_{\sigma, i}}$  be the  $\Sigma_{|a|}^0$  set thus described. Then,

$$H_{2^a} = \bigcup_{\sigma} \left( \{n : \Phi_n(\sigma, n) \downarrow\} \cap \bigcap_{i < |\sigma|} B_{e_{\sigma, i}} \right).$$

According to Lemma 1.3 and Lemma 1.4 it is indeed a  $\Sigma_{|a|}^0$  set. ■

### Quantifiers alternation

Note that in the previous proof, from the fact that  $H_{2^a}$  is a  $\Sigma_1^0(X)$  set with  $X$  a  $\Delta_{|a|}^0$  set, we obtain a  $\Sigma_{|a|+1}^0$  description of  $H_{2^a}$ . Thus, the equivalence between  $\exists$  quantification and union, as well as between  $\forall$  quantification and intersection, “morally” continues in the transfinite, simply the first-order language no longer allows us to formally account for it (we would need sentences of infinite length).

### Complexity of jump iterations

Note that for  $a \in \mathcal{O}_{<\omega}$  the set  $H_a$  is  $\Sigma_{|a|}^0$  (i.e.  $\emptyset^{(n)}$  is  $\Sigma_n^0$ ) while for  $a \in \mathcal{O}_{\geq\omega}$  it is the set  $H_{2^a}$  which is  $\Sigma_{|a|}^0$ . This phenomenon is due to the limit stages.

We have seen with Proposition 5-5.3 that for  $a \in \mathcal{O}_{<\omega}$ ,  $H_a$  is  $\Sigma_{|a|}^0$ -complete. We are now going to prove that each set  $H_{2^a}$  is  $\Sigma_{|a|}^0$ -complete, i.e., each  $\Sigma_{|a|}^0$  set is many-one reducible to  $H_{2^a}$ . However, this requires a bit of work, we start with an effective transformation of the  $\Sigma_\alpha^0$ -codes into well-founded trees of height  $\alpha$ .

**Proposition 1.8.** There exists a total computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $e$  is a  $\Sigma_\alpha^0$ -code of the effective Kleene hierarchy iff  $f(e) \in \mathcal{T}_{=\alpha}$ . ★

The proof of the preceding proposition is not difficult and we are simply giving the general idea here, leaving the details to the reader. The  $f$  function returns the code of the tree corresponding to the unfolding of the  $\Sigma_\alpha^0$ -code by simply omitting the passage from the complement of  $\Pi_\alpha^0$  to  $\Sigma_\alpha^0$ . If at some point we find that  $e$  is not a valid code, then the function returns the code of an ill-founded tree. Note that this implies having  $\omega_1^{ck}$  as a bound on the  $\Sigma_\alpha^0$  sets:

### Corollary 1.9

*If a subset of  $\mathbb{N}$  is  $\Sigma_\alpha^0$  then  $\alpha < \omega_1^{ck}$ .*

Let us now see the key theorem in the proof that every set  $H_{2^a}$  is  $\Sigma_{|a|}^0$ -complete.

### Theorem 1.10 (Spector [213])

*For any ordinal  $\alpha < \omega_1^{ck}$  and any  $a \in \mathcal{O}_{=\alpha}$ , we have*

1.  $\mathcal{O}_{<\alpha} \leq_T H_{2^a}$  uniformly in  $a$ .

2.  $\mathcal{T}_{<\alpha} \leq_T H_{2^a}$  uniformly in  $a$ .

PROOF. We want two functionals  $\Psi_1, \Psi_2$  such that for all  $a \in \mathcal{O}_{=\alpha}$  and  $x \in \mathbb{N}$  we have:

- $x \in \mathcal{O}_{<\alpha} \rightarrow \Psi_1(H_{2^a}, a, x) = 1$  and  $x \notin \mathcal{O}_{<\alpha} \rightarrow \Psi_1(H_{2^a}, a, x) = 0$
- $x \in \mathcal{T}_{<\alpha} \rightarrow \Psi_2(H_{2^a}, a, x) = 1$  and  $x \notin \mathcal{T}_{<\alpha} \rightarrow \Psi_2(H_{2^a}, a, x) = 0$

The function  $\Psi_1(H_{2^a}, a, x)$  does the following computation:

1. If  $a = 1$  then the computation returns 0, and indeed  $\mathcal{O}_{<|1|}$  is empty.
2. If  $a = 2^b$  then the computation returns 1 if the following condition is true (and returns 0 otherwise):
  - (a)  $\Psi_1(H_{2^b}, b, x) = 1$ , which corresponds to the fact that  $x \in \mathcal{O}_{<|b|}$ .  
It remains to cover the cases where  $|x| = |b|$ .
  - (b) or  $x = 2^y$ ,  $|b|$  is successor and  $\Psi_1(H_{2^b}, b, y) = 1$  which corresponds to the fact that  $x \in \mathcal{O}_{=|b|}$  with  $b$  successor.
  - (c) or  $x = 3 \cdot 5^e$ ,  $b$  is limit and the following  $\Pi_2^0(H_b)$  statement is true: for all  $n \in \mathbb{N}$  we have  $\Phi_e(n) \downarrow_{<_o} \Phi_e(n+1) \downarrow$  and for all  $n$  there exists  $c <_o b$  such that  $\Psi_1(H_{2^c}, c, \Phi_e(n)) = 1$ . This case corresponds to the fact that  $x \in \mathcal{O}_{=|b|}$  with  $b$  limit. Note that the  $\Pi_2^0(H_b)$  statement is also  $\Pi_1^0(H'_b)$ . As  $H_{2^a} = H'_b$  this is a question that can be answered using our oracle.
3. If  $a = 3 \cdot 5^e$ , then the computation returns 1 if the following  $\Sigma_1^0(H_a)$  condition is true (and returns 0 otherwise): there exists  $b <_o a$  such that  $\Psi_1(H_{2^b}, b, x) = 1$ . This case corresponds to the fact that  $x \in \mathcal{O}_{<|a|}$  with  $a$  limit. As  $H_{2^a} = H'_a$  this is a question that can be answered using our oracle.

One easily checks by induction that the computation does what is expected. The function  $\Psi_2(H_{2^a}, a, x)$  does the following computation:

1. If  $a = 1$  then  $\Psi_2(H_{2^a}, a, x) = 0$ , and indeed  $\mathcal{T}_{<|1|}$  is empty.
2. If  $a = 2^b$ , let  $T$  be the tree encoded by  $x$ . Then,  $\Psi_2(H_{2^a}, a, x) = 1$  iff the following  $\Pi_1^0(H_{2^b})$  statement is true: for all  $n \in T$  — in other words, for any node of depth 1 in  $T$  — and all  $x_n$  where  $x_n$  is the code of  $T \upharpoonright_n$ , we have  $\Psi_2(H_{2^b}, b, x_n) = 1$ . This corresponds to the fact that for each subtree  $T \upharpoonright_n$  we have  $|T \upharpoonright_n| < |b|$  and therefore to the fact that  $|T| < |a|$ .

3. If  $a = 3 \cdot 5^e$ , then  $\Psi_2(H_{2^a}, a, x) = 1$  iff the following  $\Sigma_1^0(H_a)$  statement is true: there exists  $b <_o a$  such that  $\Psi_2(H_{2^b}, b, x) = 1$ . This case corresponds to the fact that  $x \in \mathcal{T}_{<|a|}$  with  $a$  limit. ■

The previous theorem cannot be improved. For many ordinals  $\alpha$  the set  $\mathcal{O}_{<\alpha}$  will in fact be computable by sets  $H_a$  for  $a \in \mathcal{O}_{<\alpha}$ , but for ordinals  $\alpha$  which are limits of limits, it is possible to show that the set  $\mathcal{T}_{<\alpha}$  is  $\Sigma_\alpha^0$  but not  $\Pi_\alpha^0$ , and that the set  $\mathcal{T}_{<\alpha+1}$  is  $\Pi_{\alpha+1}^0$  but not  $\Sigma_{\alpha+1}^0$ . The precise limits are given in the following exercise.

**Exercise 1.11. (★★)** Show that for any  $\alpha = 0$  or limit, and for any  $k, p \in \omega$ :

1. The set  $\mathcal{T}_{<\omega(\alpha+k)}$  is  $\Sigma_{\alpha+2k}^0$ -complete.
2. The set  $\mathcal{T}_{\leq \omega(\alpha+k)+p}$  is  $\Pi_{\alpha+2k+1}^0$ -complete. ◇

Let us now show that sets of the form  $H_a$  are complete for their complexity classes.

**Theorem 1.12**

Let  $\alpha < \omega_1^{ck}$ , let  $a \in \mathcal{O}_{=\alpha}$  and let  $A$  be a  $\Sigma_\alpha^0$  set. Then,  $A$  is many-one reducible to  $H_{2^a}$ , uniformly in  $a$  and in a  $\Sigma_\alpha^0$ -code of  $A$ .

We will need the following lemma.

**Lemma 1.13.** Let  $e$  be a  $\Sigma_\alpha^0$ -code. Let  $b \in \mathcal{O}$  be such that  $\alpha + 2 \leq |b|$ . Then,  $H_b$  can find uniformly in  $b$  and in  $e$  a code  $a <_o b$  such that  $|a| = \alpha$ .★

PROOF. We start by transforming  $e$  into a code  $f(e) \in \mathcal{T}_{=\alpha}$  via Proposition 1.8. Given  $b$ , we enumerate all the codes  $a$  such that  $a + 1 <_o b$  while looking for the smallest such that  $f(e) \in \mathcal{T}_{=|a|}$ , that is to say the unique  $a$  such that :

- (1)  $f(e) \in \mathcal{T}_{<\text{succ}(|a|)}$
- (2)  $\forall c <_o a \ f(e) \notin \mathcal{T}_{<\text{succ}(|c|)}$

For (1), according to Theorem 1.10 we have  $\mathcal{T}_{<\text{succ}(|a|)} \leq_T H_{2^{2^a}}$  which is at worst equals  $H_b$ . For (2) the question requires —always using Theorem 1.10—  $H_{2^a}$  which in the worst case is equals  $H_b$ . The oracle  $H_b$  is therefore sufficient to identify  $a$ . ■

PROOF OF THEOREM 1.12. We seek to define a total computable function  $f : \mathbb{N}^3 \rightarrow \mathbb{N}$  such that for all  $\alpha < \omega_1^{ck}$ , for all  $\Sigma_\alpha^0$ -code  $e$  of a set  $A$  we have  $x \in A$  iff  $f(a, e, x) \in H_{2^a}$ .

The function  $f(a, e, x)$  returns the code of the functional which we suppose is using the oracle  $H_a$  and which handles the following cases:

1. If  $\alpha = 1$  — the initial case corresponding to the  $\Sigma_1^0$  sets — then the functional halts on its own input iff  $x$  belongs to the  $\Sigma_1^0$  set of code  $e$ .
2. Otherwise if  $\alpha$  is limit, the functional enumerates the  $\Sigma_{\beta_n}^0$ -codes  $e_n$  of sets  $A_n$  such that  $A = \bigcup_n (\mathbb{N} \setminus A_n)$  and with  $\sup_n (\beta_n + 1) = \alpha$ . Note that we have  $\beta_n + 2 \leq \alpha$  for each  $n$ . We can therefore apply Lemma 1.13 with the oracle  $H_a$  to find for each  $e_n$  a code  $b_n <_o a$  such that  $|b_n| = \beta_n$ . The function halts on its own input iff the following  $\Sigma_1^0(H_a)$  condition is true:  $\exists n \ f(b_n, e_n, x) \notin H_{b_n}$ .
3. Otherwise if  $\alpha = \text{succ}(\beta)$  with  $a = 2^b$ , the functional enumerates the  $\Sigma_{\beta_{n,m}}^0$ -codes  $e_{n,m}$  of sets  $A_{n,m}$ , with  $\sup_m (\beta_{n,m} + 1) = \alpha_n \leq \beta$ ,  $\sup_n (\alpha_n + 1) = \alpha$  and  $A = \bigcup_n \bigcap_m A_{n,m}$ . Note that we have  $\beta_{n,m} + 2 \leq \alpha$  for each  $n, m$ . We can therefore apply Lemma 1.13 with the oracle  $H_a$  to find for each  $e_{n,m}$  a code  $b_{n,m} <_o a$  such that  $|b_{n,m}| = \beta_{n,m}$ . The functional halts its own input iff the following  $\Sigma_2^0(H_b)$  condition is true:  $\exists n \forall m \ f(b_{n,m}, e_{n,m}, x) \in H_{b_{n,m}}$ . ■

Our efforts are now rewarded with three corollaries which somewhat structure our knowledge of hyperarithmetical sets. The first of them is simply the juxtaposition of Theorem 1.12 and Proposition 1.7.

**Corollary 1.14**

*For any  $\alpha < \omega_1^{ck}$  such that  $\alpha \geq \omega$ , and any  $a \in \mathcal{O}_{=\alpha}$ ,  $H_{2^a}$  is  $\Sigma_\alpha^0$ -complete uniformly in  $a$ .*

The second corollary tells us that no level is superfluous in the Kleene hierarchy: for all  $\alpha < \omega_1^{ck}$  there exists a  $\Sigma_\alpha^0$  set which is not  $\Pi_\alpha^0$ .

**Corollary 1.15**

*Kleene's hierarchy is strict.*

PROOF. For a computable ordinal  $\alpha \geq \omega$ , for  $a \in \mathcal{O}_{=\alpha}$ , each set  $H_{2^a}$  is  $\Sigma_\alpha^0$ . Suppose by contradiction that one of them is also  $\Pi_\alpha^0$ . Then, its complement is  $\Sigma_\alpha^0$ . We therefore have from Theorem 1.12 a total computable function  $f$  such that  $e \notin H_{2^a}$  iff  $f(e) \in H_{2^a}$ , that is to say  $\Phi_e(H_a, e) \uparrow$  iff  $\Phi_{f(e)}(H_a, f(e)) \downarrow$ . Note that we also have a total computable function  $g$  such that  $\Phi_e(H_a, e) \downarrow$  implies  $\forall n \ \Phi_{g(e)}(H_a, n) \downarrow$  and  $\Phi_e(H_a, e) \uparrow$  implies  $\forall n \ \Phi_{g(e)}(H_a, n) \uparrow$ . So in particular  $\Phi_e(H_a, e) \uparrow$  iff  $\Phi_{g(f(e))}(H_a, e) \downarrow$ . According to the fixed point theorem there exists  $e$  such that  $\Phi_e(H_a, e) = \Phi_{g(f(e))}(H_a, e)$ , which is a contradiction. ■

The last corollary will finally allow us to abstract ourselves a bit from our notation system for ordinals:

**Corollary 1.16**

Let  $\alpha < \omega_1^{ck}$  and  $a, b \in \mathcal{O}_{=\alpha}$ . Then,  $H_{2^a} \equiv_m H_{2^b}$ .

PROOF. It suffices to note that each  $H_{2^a}$  is  $\Sigma_{|a|}^0$ -complete. ■

Note the many-one equivalences only work for successor cases. What about sets  $H_a$  for a limit ordinal  $a$ ? These sets are  $\Delta_{|a|}^0$  and if we always have  $H_a \equiv_T H_b$  for  $a, b \in \mathcal{O}_{=\alpha}$  with  $\alpha$  limit, we will not necessarily have  $H_a \equiv_m H_b$ . Moschovakis [161] showed that the equivalence still holds when  $\alpha$  is of the form  $\beta + \omega$ , but that the structure of the many-one degrees within the Turing degree of an  $H_a$  is very chaotic in case  $|a|$  is a limit of limit ordinal.

We will use the previous corollary to consider the  $\alpha$ -th iterations of the jump up to the many-one degrees (or up to the Turing degrees in limit cases):

**Notation**

For  $\alpha < \omega_1^{ck}$  we will sometimes write  $\emptyset^{(\alpha)}$  to denote the set  $H_a$  for some  $a \in \mathcal{O}_{=\alpha}$  without further specifications.

In particular each set  $\emptyset^{(\alpha+1)}$  is well defined in the many-one degrees, as the  $\Sigma_\alpha^0$ -complete set, and for  $\alpha$  limit the set  $\emptyset^{(\alpha)}$  is well defined in the Turing degrees.

## 2. $\Pi_2^0$ singletons

We saw in Section 26-1 that the  $\Pi_2^0$  singletons in Cantor space exceeded the arithmetic hierarchy, and in particular that the Turing  $\omega$ -jump of  $\emptyset$  was a  $\Pi_2^0$  singleton. We will now see that we can find  $\Pi_2^0$  singletons at every level of the Kleene hierarchy, and therefore that any hyperarithmetic set is computable by a  $\Pi_2^0$  singleton. We will see later in Section 29-5 that the converse is true.

**Theorem 2.1**

There is a  $\Pi_2^0$  class  $\mathcal{P} \subseteq \mathbb{N} \times 2^\mathbb{N}$  such that for all  $a \in \mathcal{O}$  the class

$$\{X : (a, X) \in \mathcal{P}\}$$

is the singleton  $H_a$ .

PROOF. Let us remember  $\mathcal{S}$  from Example 17-3.7: the  $\Pi_2^0$  class containing exactly the sets which are Turing jumps of other sets. Let us also denote by  $\Psi$  the functional such that  $\Psi(X') = X$  for all  $X$ .

Using the fixed point theorem, we then define the class  $\mathcal{P}$  as being:

$$\mathcal{P} = \left\{ (a, X) : \begin{array}{l} a = 1 \text{ and } X = \emptyset \text{ or} \\ a = 2^b \text{ and } X \in \mathcal{S} \text{ and } (b, \Psi(X)) \in \mathcal{P} \text{ or} \\ a = 3 \cdot 5^b \text{ and } X = \bigoplus_n X_n \text{ with } \forall n (\Phi_b(n), X_n) \in \mathcal{P} \end{array} \right\}$$

Let us see a little more formally how the class  $\mathcal{P}$  is defined. For any  $n \in \mathbb{N}$ , let  $S_n \subseteq 2^{<\mathbb{N}}$  be the  $\Sigma_1^0$  set such that  $\mathcal{S} = \bigcap_n [S_n]$ . We can assume without loss of generality that each  $S_n$  is closed under suffix: if  $\sigma \in S_n$  then any extension of  $\sigma$  belongs to  $S_n$ . In order to use the fixed point theorem, we define the function  $f : \mathbb{N} \rightarrow \mathbb{N}$  which on a code  $e$  returns the code  $f(e)$  such that:

$$\begin{aligned} \Phi_{f(e)}(a, n, \sigma) \downarrow \quad &\leftrightarrow \quad \vee \quad \begin{array}{l} (a = 1 \wedge \sigma \prec 0^\infty) \\ (a = 2^b \wedge \sigma \in S_n \wedge \Phi_e(b, n, \Psi(\sigma)) \downarrow) \\ (a = 3 \cdot 5^b \wedge \Phi_e(\Phi_b(n), n, \sigma_n) \text{ where } \sigma = \bigoplus_n \sigma_n) \end{array} \end{aligned}$$

According to the fixed point theorem there is  $e$  such that  $\Phi_{f(e)} = \Phi_e$ . We then show by induction that for any  $a \in \mathcal{O}$ , the  $\Pi_2^0$  class given by  $\bigcap_n [\{\sigma : \Phi_e(a, n, \sigma) \downarrow\}]$  contains the set  $H_a$  and only it. ■

### Corollary 2.2

Any hyperarithmetical set is computable by a  $\Pi_2^0$  singleton.

PROOF. According to Theorem 1.12 and Theorem 2.1. ■

What does  $\{X : (a, X) \in \mathcal{P}\}$  look like for an element  $a \notin \mathcal{O}$ ? for most of these elements the class in question will be empty, but we will see with Theorem 31-3.3 that this is not always the case: there are elements  $a \notin \mathcal{O}$ , but which look like elements of  $\mathcal{O}$ , in the sense that one can enumerate elements  $b < a$  the same way one would do for elements of  $\mathcal{O}$ , without ever finding any inconsistency: simply this enumeration will contain an infinite sequence of elements  $\dots < b_4 < b_3 < b_2 < b_1 < a$ . This phenomenon with fascinating consequences will be studied in sections 31-3 and 31-4.

Let us now see a notion introduced by Groszek and Slaman, and which, as we will see with Theorem 29-5.4, exactly characterizes hyperarithmetical sets. The following definition can be seen as an extension of the definition of modulus for  $\Delta_2^0$  sets (see Definition 4-7.7).

**Definition 2.3 (Groszek and Slaman [78]).** A set  $X$  admits a modulus if there exists a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for any function  $g \geq f$  we have  $g \geq_T X$ .  $\diamond$

**Theorem 2.4**

*Any hyperarithmetical set admits a modulus.*

PROOF. Let  $\bigcap_n \mathcal{U}_n$  be a  $\Pi_2^0$  class containing a single element  $X$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be a function such that  $X \in \mathcal{U}_n[f(n)]$  for all  $n$ . For all  $g \geq f$  the class  $\bigcap_n \mathcal{U}_n[g(n)]$  is  $\Pi_1^0(g)$  and contains only the point  $X$ . According to Proposition 8-3.6, for all  $g \geq f$  the set  $X$  is  $g$ -computable. So every  $\Pi_2^0$  singleton admits a modulus. Corollary 2.2 allows us to conclude.  $\blacksquare$

We will see the converse of the previous theorem with Theorem 29-5.4, which will provide us with an elegant characterization of hyperarithmetical sets.

### 3. Relativization

The iteration of the jump can be relativized to any oracle  $X$ .

**Definition 3.1.** Let  $X \in 2^{\mathbb{N}}$ . We define the transfinite iterations  $H_a^X$  of the jump for the elements  $b \in \mathcal{O}^X$ :

1.  $H_1^X = X$
2.  $H_{2^a}^X = (H_a^X)'$
3.  $H_{3 \cdot 5^e}^X = \bigoplus_n H_{\Phi_e(X, n)}^X$

$\diamond$

Kleene's hierarchy is also easily relativized to an oracle  $X$ , along ordinals  $\alpha < \omega_1^X$ , as the various theorems seen so far, in particular:

**Theorem 3.2**

*For all  $X \in 2^{\mathbb{N}}$ , all ordinal  $\alpha < \omega_1^X$  and all  $a \in \mathcal{O}_{=\alpha}^X$ , we have*

1.  $\mathcal{O}_{<\alpha}^X \leq_T H_{2^a}^X$  uniformly in  $a$ .
2.  $\mathcal{T}_{<\alpha}^X \leq_T H_{2^a}^X$  uniformly in  $a$ .

**Theorem 3.3**

Let  $X \in 2^{\mathbb{N}}$ . Let  $\alpha < \omega_1^X$  and  $a \in \mathcal{O}_{=\alpha}^X$ . The set  $H_{2^a}^X$  is  $\Sigma_\alpha^0(X)$ -complete uniformly in  $X$  and  $a$ .

**Theorem 3.4**

There exists  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  a  $\Pi_2^0$  class such that for all  $X$ , for all  $a \in \mathcal{O}^X$ , the class  $\{Y : (a, X, Y) \in \mathcal{P}\}$  is the singleton  $H_a^X$ .

Note in the previous theorems that we must also relativize the smallest non-computable ordinal  $\omega_1^{ck}$ , and use instead  $\omega_1^X$ , the smallest non  $X$ -computable ordinal.

## 4. Effective Borel hierarchy

By following the Kleene hierarchy example for sets of integers, it is possible to give an effective version of the Borel hierarchy, the full version of which was introduced with Definition 27-3.3. We give here directly the relativized version to any oracle  $X$ .

**Definition 4.1.** The effective Borel hierarchy is defined by induction on ordinals, relative to an oracle  $X \in 2^{\mathbb{N}}$ :

- A  $\Sigma_1^0(X)$ -code is given by a pair  $\langle 0, e \rangle$ . The corresponding class  $\mathcal{U}$  is given by  $\mathcal{U} = \bigcup_{\sigma \in W_e^X} [\sigma]$ .
- A  $\Pi_\alpha^0(X)$ -code is given by a pair  $\langle 1, e \rangle$  where  $e$  is a  $\Sigma_\alpha^0(X)$ -code. The corresponding class  $\mathcal{B}$  is given by  $\mathcal{B} = 2^{\mathbb{N}} \setminus \mathcal{A}$  where  $\mathcal{A}$  is the set corresponding to the code  $e$ .
- A  $\Sigma_\alpha^0(X)$ -code is given by a pair  $\langle 2, e \rangle$  where  $W_e^X$  is non-empty and enumerates on oracle  $X$  some  $\Pi_{\beta_n}^0(X)$ -codes for  $\beta_n < \alpha$ , with  $\sup_n(\beta_n + 1) = \alpha$ . The corresponding class  $\mathcal{B}$  is given by  $\bigcup_n \mathcal{A}_n$  where  $\mathcal{A}_n$  is the class corresponding to the  $n$ -th code enumerated by  $W_e^X$ .

We say that a class  $\mathcal{B} \subseteq 2^{\mathbb{N}}$  is  $\Sigma_\alpha^0(X)$  (resp.  $\Pi_\alpha^0(X)$ ) if it corresponds to a  $\Sigma_\alpha^0(X)$ -code (resp. a  $\Pi_\alpha^0(X)$ -code). We say that the class  $\mathcal{B}$  is  $\Delta_\alpha^0(X)$  if it is both  $\Sigma_\alpha^0(X)$  and  $\Pi_\alpha^0(X)$ .  $\diamond$

Remember the definition of non-effective Borel classes through ordinals. We insisted on the use of the countable choice axiom in order to guarantee that any Borelian is indeed  $\Sigma_\alpha^0$  for a certain ordinal  $\alpha$  countable. The main interest is then to have any Borel class as being effective relative to some

oracle  $X$ : as  $\alpha$  is countable it can be encoded by an oracle, which can then encode the well-founded tree corresponding to the unfolding of the Borel code.

We thus obtain that any Borel class is  $\Sigma_\alpha^0(X)$  for some oracle  $X$ .

**Proposition 4.2.** If a class is  $\Sigma_\alpha^0(X)$  then  $\alpha < \omega_1^X$ . ★

PROOF. It suffices to repeat the proof of Proposition 1.8 to transform a  $\Sigma_\alpha^0(X)$ -code into an element of  $\mathcal{T}^X$ . We deduce that  $\alpha < \omega_1^X$ . ■

Just as Kleene's hierarchy is strict, it is possible to show that the Borel hierarchy is also strict, in a strong sense:

**Theorem 4.3**

*Let  $X \in 2^\mathbb{N}$ . For all  $\alpha < \omega_1^X$  there exists a  $\Sigma_\alpha^0(X)$  class which is not  $\Pi_\alpha^0$ .*

PROOF. The detailed proof is left in exercise. We suppose to simplify the presentation  $X = \emptyset$ , the relativization not posing any particular problem. The general idea is as follows: we define a functional  $\Psi$  such that for all  $a \in \mathcal{O}$  we have:

- (1) For any  $Y \in 2^\mathbb{N}$  the value  $\Psi(Y, a)$  is a  $\Sigma_{|a|}^0(Y)$ -code.
- (2) For any  $\Sigma_{|a|}^0$  class  $\mathcal{A}$  there exists an oracle  $Y$  such that  $\Psi(Y, a)$  is a  $\Sigma_{|a|}^0(Y)$ -code of  $\mathcal{A}$ .
- (3) For all  $Y \in 2^\mathbb{N}$  the class

$$\{Y \in 2^\mathbb{N} : Y \text{ belongs to the class of code } \Psi(Y, a)\}$$

is  $\Sigma_\alpha^0$ .

The functional  $\Psi$  simply performs a Borelian coding system via its oracle, using  $a \in \mathcal{O}$  to control the height of the produced code. We then show that the class

$$\mathcal{C} = \{Y \in 2^\mathbb{N} : Y \text{ does not belong to the class of code } \Psi(Y, a)\}$$

cannot be  $\Sigma_{|a|}^0$ . Indeed if this was the case there would be an oracle  $Y$  such that  $\Psi(Y, a)$  returns a  $\Sigma_{|a|}^0(Y)$ -code for this Borel class. We would then have  $Y \notin \mathcal{C}$  iff  $Y \in \mathcal{C}$  which is a contradiction. ■

We now make the link between Kleene's hierarchy and the effective Borel hierarchy.

**Theorem 4.4**

A class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is  $\Sigma_{\alpha}^0$  iff for  $a \in \mathcal{O}_{=\alpha}$  there exists an integer  $n$  such that

$$\mathcal{A} = \{X : n \in H_{2a}^X\}.$$

PROOF. Let us show that for  $n \in \mathbb{N}$  and  $a \in \mathcal{O}_{=\alpha}$  the class  $\{X : n \in H_{2a}^X\}$  is  $\Sigma_{\alpha}^0$  uniformly in  $n$  and  $a$ . According to Theorem 3.3 the set  $H_{2a}^X$  is  $\Sigma_{\alpha}^0(X)$  uniformly in  $X$  and  $a$ . We therefore have a computable function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that  $g(a)$  is a  $\Sigma_{\alpha}^0(X)$ -code of the set  $H_{2a}^X$  for all  $X$ .

Let us now define a computable function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that:

- (1)  $f(a, n)$  is the  $\Sigma_{\alpha}^0$ -code of a Borel class for all  $a \in \mathcal{O}_{=\alpha}$  and for all  $n$ .
- (2)  $n \in H_{2a}^X$  iff  $X$  belongs to the class described by  $\Sigma_{\alpha}^0$ -code  $f(a, n)$ .

The function  $f(a, n)$  computes the code  $g(a)$  which is for all  $X$  a  $\Sigma_{\alpha}^0(X)$ -code of the set  $H_{2a}^X$ . We then use a function  $h_n : \mathbb{N} \rightarrow \mathbb{N}$  defined inductively on the nodes of the tree described by the code  $g(a)$  (depending on an oracle  $X$ ). On a code  $\langle 2, e \rangle$  the function  $h_n$  returns a code corresponding to the union of  $[\sigma] \cap \mathcal{B}_{\sigma}$  for each  $\sigma$  such that  $W_e^{\sigma}$  enumerates a code  $e_{\sigma}$ , and where  $\mathcal{B}_{\sigma}$  is the class encoded by  $h_n(e_{\sigma})$ . On a code  $\langle 1, e \rangle$  the function  $h_n$  returns  $\langle 1, h_n(e) \rangle$ . On a code  $\langle 0, e \rangle$  the function  $h_n$  returns the code  $\langle 0, d \rangle$  where  $d$  is the code of the  $\Sigma_1^0$  class which enumerates the strings  $\sigma$  such that  $n \in W_e^{\sigma}$ . The function  $f(a, n)$  finally returns  $h_n(g(a))$ . It suffices to show by induction that  $f$  satisfies (1) and (2) above.

Let us now show that for a  $\Sigma_{\alpha}^0$  class  $\mathcal{A}$ , we can uniformly find an integer  $e$  such that  $\mathcal{A} = \{X : e \in H_{2a}^X\}$ . To do this, we first show how to find an integer  $d$  which is for all  $X$  the  $\Sigma_{\alpha}^0(X)$ -code of a set containing 0 iff  $X \in \mathcal{A}$ . For this, we transform uniformly in  $X$  the tree corresponding to a  $\Sigma_{\alpha}^0$ -code of  $\mathcal{A}$  into a  $\Sigma_{\alpha}^0(X)$ -code of the same tree, except that each leaf corresponding to a  $\Sigma_1^0$  class  $\mathcal{U} \subseteq 2^{\mathbb{N}}$ , is replaced by a leaf corresponding to a  $\Sigma_1^0(X)$  set  $U \subseteq \mathbb{N}$  such that  $0 \in U$  iff a prefix  $\sigma$  of  $X$  is such that  $[\sigma] \subseteq \mathcal{U}$ . We easily show by induction that 0 belongs to the  $\Sigma_{\alpha}^0(X)$ -code  $d$  iff  $X \in \mathcal{A}$ . We now use Theorem 3.3 to find a computable function  $h$  such that 0 belongs to our  $\Sigma_{\alpha}^0(X)$  set iff  $h(0) \in H_{2a}^X$ . We then have  $\mathcal{A} = \{X : h(0) \in H_{2a}^X\}$ . ■

We see finally another version of the above theorem, but for the classes  $\{Y : n \in \mathcal{O}_{<\alpha}^Y\}$ .

**Theorem 4.5**

Let  $Y \in 2^{\mathbb{N}}$ . For all  $n \in \mathbb{N}$  and  $\alpha < \omega_1^Y$  the class  $\{X : n \in \mathcal{O}_{<\alpha}^X\}$  is a  $\Sigma_{\alpha+1}^0(Y)$  class uniformly in  $Y$  and in  $n$ .

PROOF. We can make the same construction as in the above theorem, of a computable function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that:

- (1)  $f(a, n)$  is the  $\Sigma_{\alpha+1}^0(Y)$ -code of a Borel class for all  $a \in \mathcal{O}_{=\alpha}^Y$  and for all  $n$ .
- (2)  $n \in \mathcal{O}_{<\alpha}^X$  iff  $X$  belongs to the class described by the  $\Sigma_{\alpha+1}^0(Y)$ -code  $f(a, n)$ .

According to Theorem 3.2, for all  $X, Y$  and  $\alpha < \omega_1^Y$  we have  $\mathcal{O}_{<\alpha}^{X \oplus Y} \leq_T H_{2^a}^{X \oplus Y}$  uniformly in  $X, Y$  and in  $a \in \mathcal{O}_{=|\alpha|}^Y$ . We clearly have  $\mathcal{O}_{<\alpha}^X \leq_m \mathcal{O}_{<\alpha}^{X \oplus Y}$  because to know if  $e \in \mathcal{O}_{<\alpha}^X$  it suffices to check whether  $e' \in \mathcal{O}_{<\alpha}^{X \oplus Y}$  where  $e'$  does the same thing as  $e$  but using “half” of its oracle corresponding to  $X$ . According to Theorem 3.3 the set  $H_{2^a}^{X \oplus Y}$  is  $\Sigma_\alpha^0(X \oplus Y)$  uniformly in  $X, Y$  and  $a$ . The set  $\mathcal{O}_{<\alpha}^X$  is therefore  $\Delta_{\alpha+1}^0(X \oplus Y)$  uniformly in  $X, Y$  and in  $\alpha < \omega_1^Y$ , and it is therefore in particular  $\Sigma_{\alpha+1}^0(X \oplus Y)$ .

We can then proceed in a similar way as in the previous proof to create  $f$ . The details are left to the reader. ■

**Exercise 4.6. (★)** Show that we can consider without loss of generality that the  $\Sigma_\alpha^0$  classes (of the form  $\bigcup_n \mathcal{B}_n$ ) are increasing, that is to say with  $(\mathcal{B}_n \subseteq \mathcal{B}_{n+1})$ . ◇

**Exercise 4.7. (★)** Show that Proposition 17-4.3 continues in the transfinite: for a  $\Sigma_\alpha^0$  class  $\mathcal{A}$ , the set  $\{q \in \mathbb{Q} : \lambda(\mathcal{A}) > q\}$  is  $\Sigma_\alpha^0$  uniformly in a code of  $\mathcal{A}$ . ◇



# Chapter 29

## Beyond hyperarithmetical

What is beyond hyperarithmetical sets? According to Theorem 28-1.10,  $\mathcal{O}_{<\alpha} \leq_T H_{2^a}$  for all  $a \in \mathcal{O}_{=\alpha}$ . Using a technique similar to that of Exercise 28-1.11, one can show that fragments of Kleene's  $\mathcal{O}$  are arbitrarily complex in the hyperarithmetical hierarchy. It follows that  $\mathcal{O}$  cannot itself be hyperarithmetical.

In this chapter, we will introduce the  $\Sigma_1^1$  and  $\Pi_1^1$  classes and sets defined using second-order quantifications. These notions will allow us to give a characterization of hyperarithmetical sets purely in terms of definability by second-order formulas, and to deduce new characterizations in terms of modulus or  $\Pi_2^0$  singletons.

The  $\Pi_1^1$  sets play a central role, in particular in the correspondence between Classical Computability Theory and Higher Computability Theory. We will see to what extent the  $\Pi_1^1$  sets can be perceived as higher computably enumerable sets. We will see in particular that Kleene's  $\mathcal{O}$  plays for hyperarithmetical sets a role analogous to that played by  $\emptyset'$  for computable sets. This correspondence between computability and higher computability will be pushed further in Chapter 30 where we will see that the  $\Sigma_1^1$  classes can be seen as the higher counterpart of the  $\Pi_1^0$  classes.

### 1. A little history: the Moscow school

In the 19th century the writers Nicolas Gogol and Fyodor Dostoyevsky theorize “the Russian soul”, a vague concept still today, often perceived as a mixture of mysticism, irrationality, excess and abatement.

One cannot help but see the manifestation of this famous Russian soul — whether fantasized or real — when one immerses himself in the history of the Muscovite Set Theory at the beginning of the 20th century. The mathematicians Jean-Michel Kantor and Loren Graham have carried out an in-depth historical investigation in which they study the impact of cultural differences between Russia and Western Europe, on the reception and perception of the new mathematics that then constitutes Set Theory. They write in particular [106]:

*“Russian mathematicians of the late 19th century and early 20th century believed that their work was closely related to philosophy, religion and ideology in general. From this point of view, they stood out from most of their French colleagues.”*

### Egorov and Luzin

We have spoken in Section 17-1 of the famous French trio: Borel, Baire and Lebesgue, whose work had hatched at the beginning of the 20th century on a set of results of great richness following the work of Cantor on infinity. The concepts of Borel set then gradually make their way into the mathematical community, which perceives its aesthetics and interest. It is in line with their work that, under the leadership of Dimitri Egorov and Nicolai Luzin, the *Moscow mathematical school*, one of the most impressive that has ever existed, was born. Egorov and Luzin studied mathematics under Nikolai Vasilievich Bugaev (1837-1903) who was very interested in the theory of discontinuous functions which had for him a philosophical interest that went well beyond the logical considerations of the French and German mathematicians [106, p.89]:

*“Discontinuity is an affirmation of independent and autonomous individuality. Discontinuity also occurs where questions of final causes and aesthetic and ethical issues arise.”*

We see the striking contrast with the statements of Poincaré (see Section 17-1.1) for whom discontinuity is a logical aberration unrelated to the real world, where Bugaev sees in discontinuous mathematical objects an abstract, almost mystical interest. Bugaev was the mathematics professor of Egorov, who during several stays in France and Germany familiarized himself with the most recent developments in Set Theory. This theory fascinates him and Egorov decides to teach it when he returns to Moscow. Luzin is one of his favorite students, and they will set up together what will first be a mathematics seminar called *Lusitanie* — we do not really know today if this name was chosen in honor of Luzin or not — which will constitute the premises of what will later become the Moscow school of

mathematics. Lusitania's history is also inscribed within the Great History and the deep upheavals which will agitate Russia at the beginning of the 20th century. Let us mention on this subject an anecdote about Luzin's youth. The latter was deeply marked by the failed 1905 revolution. He lost his interest in mathematics and went through a deep crisis that lasted three years, as evidenced by the letters he wrote to Egorov [106, p.106] :

*“When you met me in college, I was just an ignorant child. I do not know what happened, but I cannot be satisfied today with the analytical functions and the Taylor series ... The misery of the people, the torments of life ... are unbearable visions ... I can no longer live on science alone ... I have nothing, no vision of the world and no education.”*

Luzin will come out of his depression with the help of his friend and spiritual guide Pavel Florensky — priest and mathematician — notably after reading Florensky's thesis *“On religious truth”*. Luzin then resumed his mathematical work. All his life he was mainly interested in Set Theory, which he studied with all the scientific rigor of a mathematician, but also with a conviction that stems from a certain form of religious faith. We can find for example in his notes:

*“We think that integers exist objectively. We believe that the totality of second-class transfinite numbers objectively exists. We want the following: having assumed their obviousness, we associate with each of the transfinite numbers a definition, a name, and that for all the transfinite numbers that we consider.”*

## Lusitania

Egorov and Luzin launch a seminar shortly before the First World War for a small group of motivated students: Lusitania, which will experience unexpected success and sustainability.

In the first decade of the 20th century, there were hardly any world-renowned mathematicians in Russia. This will change with the creation of Lusitania. In barely 10 years, many very young mathematicians from this school will find a place on the international scene, including Pavel Alexandrov who will develop an important part of modern topology, Pavel Urysohn, another renowned topologist, or Nina Bari, known for her work on trigonometric series. Andrei Kolmogorov, certainly the most famous Russian mathematician, will also go through Lusitania, but a little later. In 1930, Moscow will have become one of the main mathematical capitals of the world.

The seminar of Egorov and Luzin takes quickly: between the pedagogy and the passionate investment of the two professors, the motivation and the

exceptional scientific quality of the students, the alchemy operates. Quickly, a very close-knit, happy and hard-working group is formed, and discovers with admiration the latest developments of these “new mathematics” that constitutes Set Theory, through which arise the feeling of taking part in something important, to an intellectual adventure greater than oneself.

The history of Lusitania is all the more remarkable in that the many leading mathematical developments which occurred there took place under catastrophic material conditions: the First World War contributed to the famine and generalized shortages which plagued the country. The October 1917 revolution plunged Russia into a violent civil war. The students of Lusin and Egorov suffer from cold and hunger. The temperature in classrooms sometimes drops below 0 degrees [106, p.137]. Whatever, the students come all the same.

### Descriptive Set Theory

The first important achievement of Lusitania is that which concerns the subject of the next chapter: the birth of Descriptive Set Theory. One of the central questions at the time was that of Cantor’s continuum hypothesis. Alexandrov, then barely 18 years old, shows that the continuum hypothesis is true for all Borel classes. We will see a modern and effective form of this theorem with Corollary 30-3.3. As we know today, the continuum hypothesis will require much more substantial efforts and developments, but for the time, it was all the same an important step towards its resolution. Could we extend the proof of Alexandrov to all classes? At the time, the existence of non-Borel classes was not yet completely clear, and specific examples of these classes were even less so. A year later, another young student from Lusitania, Mikhaïl Souslin, spots an error in a Lebesgue’s proof dating from 1905, which will remain famous as “Lebesgue’s error”, marking the starting point of a new field of research: *Descriptive Set Theory*. Lebesgue’s proof concerns the following statement: the image of a Borel class by a continuous function is also a Borel class. Souslin will show that this is not necessarily true: the image of some Borel classes by some continuous functions are *strictly more complex* than the Borel classes. It is the discovery by Souslin of the so-called *analytical* classes or as we call them in this book, the  $\Sigma_1^1$  classes. Souslin [218] will show hereafter that the Borel classes are exactly the  $\Sigma_1^1$  classes whose complement is also  $\Sigma_1^1$ . Later, Kleene [115] will give — relying on results of Spector [213] — an effective version of Souslin’s theorem, and which applies not only to classes, but also to sets of integers. It is this result that we will present in Section 5.

### The end

If the legacy of Lusitania continues today, the country's political troubles will get the better of Lusitania itself, which will end around the 1920s. The new generation of mathematicians formed by Luzin and Egorov is adapting to the new order of things dictated by the power in place. But the old generation does not get used to it. Egorov does not hide his religious convictions, nor his animosity towards the new regime. He was arrested in 1930, accused of mixing mathematics and religion. In prison Egorov went on hunger strike after which he had to be transferred to a hospital where he died shortly after.

Luzin meanwhile keeps a low profile. The arrest of his colleague and friend terrifies him, and he remains as discreet as possible. He hides his religious beliefs, and tries to show his loyalty to the Soviet state. But his past catches up with him. He became the target of repeated attacks until 1936 when he was accused in a trial of being, in substance, an enemy of the Soviet party. Several of his former students testify against him, accusing him of plagiarism and nepotism. This is a terrible blow for Luzin, who for a reason still unclear today, and despite an official conviction, will ultimately neither be arrested nor expelled from the academy of sciences.

Despite this painful episode in the history of mathematics, The school founded by Egorov and Luzin endured, and through it, Russia made a deep and lasting mark on mathematics, perhaps bringing some of this mysterious Russian soul, through a certain style, and intensity in mathematical commitment. We end our historical interlude with a quote from Jean Michel Kantor and Loren Graham's book, which clearly shows the way in which the Russian school left its mark upon the mathematician's mind:

*"We kept citing with a mix of fear and admiration, the legendary mathematical seminars of Moscow, where lectures, started in the afternoon, often continued late into the evening; with seminar organizers and assistants submitting the speakers to a long series of questions in order to understand. Anyone attending the seminar could be sent to the board unexpectedly, making the Western habit of attending a seminar as a distant and distracted spectator, impossible."*

The authors of this book, who have had the good fortune to work with their Muscovite colleagues, can attest that the legend does contain a substance of truth . . .

## 2. Second-order quantifications

We saw with Kleene's hierarchy that the arithmetic hierarchy could be extended in a natural way along computable ordinals, to give the hyper-

arithmetic sets. According to the remark following Proposition 28-1.7, this extension can be thought of in terms quantifiers alternation within “infinite” formulas.

There is another natural approach to extend the arithmetic hierarchy, consisting in allowing second-order quantifiers, in other words, quantifiers on sets of integers. We then obtain a hierarchy whose first level already contains all the hyperarithmetic sets, as we will see later.

**Definition 2.1.** Let  $A \subseteq \mathbb{N}$  be a set.

1.  $A$  is  $\Sigma_1^1$  if there exists an arithmetic class  $\mathcal{B} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  such that

$$A = \{n \in \mathbb{N} : \exists X \in 2^{\mathbb{N}} (X, n) \in \mathcal{B}\}.$$

2.  $A$  is  $\Pi_1^1$  if there exists an arithmetic class  $\mathcal{B} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  such that

$$A = \{n \in \mathbb{N} : \forall X \in 2^{\mathbb{N}} (X, n) \in \mathcal{B}\}.$$

3.  $A$  is  $\Delta_1^1$  iff  $A$  is  $\Pi_1^1$  and  $\Sigma_1^1$ .

The  $\Sigma_1^1$  and  $\Pi_1^1$  sets are also called respectively *effective analytical* and *effective co-analytical* sets.  $\diamond$

Note that the  $\Pi_1^1$  sets are the complements of the  $\Sigma_1^1$  sets and vice versa. A  $\Sigma_1^1$  (resp.  $\Pi_1^1$ ) set has the right to use existential (resp. universal) quantifications, not on integers, but on sets of integers. This gives much more expressive power, and we will see that the hyperarithmetic sets are all  $\Pi_1^1$  and  $\Sigma_1^1$ . We will in fact see with Theorem 5.2 that the hyperarithmetic sets are exactly the  $\Delta_1^1$  sets.

We define in the same way the notions of  $\Sigma_1^1$  and  $\Pi_1^1$  classes.

**Definition 2.2.** Let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  be a class.

1.  $\mathcal{A}$  is  $\Sigma_1^1$  if there exists an arithmetic class  $\mathcal{B} \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  such that  $\mathcal{A} = \{X \in 2^{\mathbb{N}} : \exists Y \in 2^{\mathbb{N}} (Y, X) \in \mathcal{B}\}$
2.  $\mathcal{A}$  is  $\Pi_1^1$  if there exists an arithmetic class  $\mathcal{B} \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  such that  $\mathcal{A} = \{X \in 2^{\mathbb{N}} : \forall Y \in 2^{\mathbb{N}} (Y, X) \in \mathcal{B}\}$
3.  $\mathcal{A}$  is  $\Delta_1^1$  iff  $\mathcal{A}$  is  $\Pi_1^1$  and  $\Sigma_1^1$ .

The  $\Sigma_1^1$  and  $\Pi_1^1$  classes are also called respectively *effective analytical* and *effective co-analytical* classes.  $\diamond$

Note that the definitions of  $\Sigma_1^1$  and  $\Pi_1^1$  classes and sets naturally extend to the Baire space as well as to the product of different spaces over which the definition is valid.

**Example 2.3.** A class  $\mathcal{A} \subseteq \mathbb{N}^{\mathbb{N}} \times 2^{\mathbb{N}} \times \mathbb{N}$  is  $\Sigma_1^1$  if there exists an arithmetic set  $\mathcal{B} \subseteq 2^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}} \times 2^{\mathbb{N}} \times \mathbb{N}$  such that  $A = \{(f, X, n) : \exists Y \in 2^{\mathbb{N}} (Y, f, X, n) \in \mathcal{B}\}$ .

## 2.1. Closing properties

We will see that in an equivalent way, the quantifications of the  $\Sigma_1^1$  and  $\Pi_1^1$  classes and sets can be done on functions rather than on sets of integers. In general, these classes will be easier to handle via quantifications on functions. We introduce for this some convenient notations.

### Notation

- Let  $X$  be an infinite set. Then,  $f_X$  denotes the function from  $\mathbb{N}$  to  $\mathbb{N}$  encoded by  $X$ , i.e., the function such that  $f_X(n)$  returns the  $n$ -th element of  $X$ .
- The notation  $\exists X \oplus Y \mathcal{B}(X, Y)$  is a shorthand meaning  $\exists Z \mathcal{B}(Z^{[0]}, Z^{[1]})$  where  $Z^{[i]} = \{2n + i : n \in Z\}$ .
- The same goes for the notation  $\exists(\bigoplus_m X_m) \forall m \mathcal{B}(X_m, m)$

The following proposition implies in particular that the  $\Sigma_1^1$  classes or sets can be defined with an arbitrary number of existential quantifiers on the sets, essentially by encoding two quantifications into one using the effective join.

### Proposition 2.4.

- (1) The  $\Sigma_1^1$  sets or classes are closed under effective union indexed by sets of integers or indexed by functions from  $\mathbb{N}$  into  $\mathbb{N}$ .
- (2) The  $\Pi_1^1$  sets or classes are closed under effective intersection indexed by sets of integers or by functions from  $\mathbb{N}$  into  $\mathbb{N}$ . ★

PROOF. Note that (1) is equivalent to (2) by passing to the complement. We thus show only (1), and we do it at the same time for classes and sets by considering classes in the product space  $2^{\mathbb{N}} \times \mathbb{N}$ .

Let  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}} \times \mathbb{N}$  be the  $\Sigma_1^1$  class equal to  $\{(Y, X, n) : \exists Z (Z, Y, X, n) \in \mathcal{B}\}$  for an arithmetic set  $\mathcal{B}$ . Let us show that the class

$$\{(X, n) : \exists Y (Y, X, n) \in \mathcal{A}\}$$

is also  $\Sigma_1^1$ . We have  $\exists Y (Y, X, n) \in \mathcal{A}$  iff  $\exists Z \exists Y (Z, Y, X, n) \in \mathcal{B}$  iff  $\exists Z \oplus Y (Z, Y, X, n) \in \mathcal{B}$ . This gives closure for effective unions indexed by sets of integers.

Let  $\mathcal{A} \subseteq \mathbb{N}^{\mathbb{N}} \times 2^{\mathbb{N}} \times \mathbb{N}$  be the  $\Sigma_1^1$  class equal to  $\{(f, X, n) : \exists Y (Y, f, X, n) \in \mathcal{B}\}$  for an arithmetic set  $\mathcal{B}$ . Let us show that the class

$$\{(X, n) : \exists f (f, X, n) \in \mathcal{A}\}$$

is also  $\Sigma_1^1$ . We have  $\exists f (f, X, n) \in \mathcal{A}$  iff  $\exists Y \exists f (Y, f, X, n) \in \mathcal{B}$  iff

$$\exists Y \exists Z \text{ such that } (Z \text{ is infinite and } (Y, f_Z, X, n) \in \mathcal{B}).$$

To be infinite is indeed an arithmetic predicate. By the closure property of the previous paragraph, the class is indeed  $\Sigma_1^1$ . ■

### Quantification over $\mathbb{N}^{\mathbb{N}}$ vs $2^{\mathbb{N}}$

The preceding proposition has an important implication: the  $\Sigma_1^1$  sets or classes can be described in an equivalent way via second-order quantifications on  $\mathbb{N}^{\mathbb{N}}$  or on  $2^{\mathbb{N}}$ . This is a property that we will use for the next normal form theorem.

In order to simplify the proof of the following proposition, we will use the axiom of countable choice. This axiom is however not absolutely necessary: we will see later with Proposition 30-1.2 that it is automatically verified for the  $\Sigma_1^1$  classes (which are those on which it is used in the following proposition).

**Proposition 2.5.**  $\Sigma_1^1/\Pi_1^1$  sets or classes are closed under effective union or intersection indexed by integers. ★

PROOF. Here again we show the proposition for both classes and sets. Let  $\mathcal{A} \subseteq \mathbb{N} \times (2^{\mathbb{N}} \times \mathbb{N})$  be the  $\Sigma_1^1$  class equal to  $\{(m, X, n) : \exists f (f, m, X, n) \in \mathcal{B}\}$  for an arithmetic set  $\mathcal{B}$ . Let us show that the class

$$\{(X, n) : \exists m (m, X, n) \in \mathcal{A}\}$$

is also  $\Sigma_1^1$ . We have  $\exists m (m, X, n) \in \mathcal{A}$  iff  $\exists m \exists f (f, m, X, n) \in \mathcal{B}$  iff  $\exists f \exists m (f, m, X, n) \in \mathcal{B}$ . The predicate  $\exists m (f, m, X, n) \in \mathcal{B}$  being arithmetic, the class is indeed  $\Sigma_1^1$ .

Let us show that the class  $\{(X, n) : \forall m (m, X, n) \in \mathcal{A}\}$  is also  $\Sigma_1^1$ . We have  $\forall m (m, X, n) \in \mathcal{A}$  iff  $\forall m \exists f (f, m, X, n) \in \mathcal{B}$ . In particular for all  $m$  there exists a function  $f$  such that  $(f, m, X, n) \in \mathcal{B}$ . By using the axiom of choice, it suffices to choose for each  $m$  a function  $f_m$  such that  $(f_m, m, X, n) \in \mathcal{B}$ . By considering the function  $\bigoplus_m f_m$  we then have

$$\forall m \exists f (f, m, X, n) \in \mathcal{B} \text{ iff } \exists \left( \bigoplus_m f_m \right) \forall m (f_m, m, X, n) \in \mathcal{B}.$$

The predicate  $\forall m (f_m, m, X, n) \in \mathcal{B}$  being arithmetic, the class is indeed  $\Sigma_1^1$ .

By passing to the complement, the  $\Pi_1^1$  sets or classes are closed under union or effective intersection indexed by the integers. ■

We will see with Proposition 30-1.2 that we can build choice functions on the countable sequence of  $\Sigma_1^1$  classes — of course without using the axiom of choice. In the second part of the previous proposition, the classes  $\mathcal{B}_m = \{f \in \mathbb{N}^{\mathbb{N}} : (f, m, X, n) \in \mathcal{B}\}$  are arithmetic (with the oracle  $X$ ) and therefore in particular  $\Sigma_1^1(X)$  (we will shortly introduce relativization of these classes). We can therefore choose in each  $\mathcal{B}_m$  a function  $f_m$  without using the axiom of choice. It is simply necessary to check that the developments leading to the construction of this choice function do not use the preceding proposition. This is the case, the developments in question are mainly due to Kleene's normal form theorem of the next section, which does not need the closure of  $\Sigma_1^1$  classes by effective union or intersection.

## 2.2. Normal forms

The advantage of using quantifications on functions rather than sets of integers, is the simplicity of the following normal forms, for the  $\Sigma_1^1$  and  $\Pi_1^1$  predicates.

### **Theorem 2.6 (Kleene [114])**

Let  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  be a class.

- (1)  $\mathcal{A}$  is  $\Sigma_1^1$  iff there exists a functional  $\Phi_e$  such that  $(X, n) \in \mathcal{A}$  iff  $\exists f \Phi_e(f, X, n) \uparrow$ .
- (2)  $\mathcal{A}$  is  $\Pi_1^1$  iff there exists a functional  $\Phi_e$  such that  $(X, n) \in \mathcal{A}$  iff  $\forall f \Phi_e(f, X, n) \downarrow$ .

PROOF. Note that (1) is equivalent to (2) by complement. We therefore simply show (1). Without loss of generality, a  $\Sigma_1^1$  class in  $2^{\mathbb{N}} \times \mathbb{N}$  is of the form  $\{(X, n) : \exists f \mathcal{B}(f, X, n)\}$ . Moreover  $\mathcal{B}$  is of the form  $\exists x_1 \forall x_2 \dots \mathcal{R}(x_1, x_2, \dots, x_m, f)$  for some  $m$  and some computable predicate  $\mathcal{R}$ . We show by induction that we can “merge” the quantifications of  $\mathcal{B}$  with the existential quantification on functions:

Let  $\mathcal{A} = \{(X, n) : \exists f \exists y (f, y, X, n) \in \mathcal{B}\}$  for an arithmetic predicate  $\mathcal{B}$ . Then, we can merge  $y$  in the quantification of  $f$  because we also have  $\mathcal{A} = \{(X, n) : \exists f (f \upharpoonright_{\mathbb{N}^*}, f(0), X, n) \in \mathcal{B}\}$  where  $f \upharpoonright_{\mathbb{N}^*}$  is the function such that  $f \upharpoonright_{\mathbb{N}^*}(n) = f(n+1)$  for all  $n$ .

Let  $\mathcal{A} = \{(X, n) : \exists f \forall y \exists z (f, y, z, X, n) \in \mathcal{B}\}$  for an arithmetic predicate  $\mathcal{B}$ . Then, we also have  $\mathcal{A} = \{(X, n) : \exists f \oplus g \forall y (f, y, g(y), X, n) \in \mathcal{B}\}$ .

We can thus keep merging the arithmetic quantifiers until reaching a predicate of the type  $\mathcal{A} = \{(X, n) : \exists f \forall y (f, y, X, n) \in \mathcal{R}\}$  where  $\mathcal{R}$  is a computable predicate, which one easily transforms into a functional  $\Phi_e$  such that  $\mathcal{A} = \{(X, n) : \exists f \Phi_e(f, X, n) \uparrow\}$ . ■

### 2.3. Relativization

As for hyperarithmetical hierarchies, the notions of  $\Sigma_1^1/\Pi_1^1$  sets/classes can be relativized to an oracle. The normal form theorem is also straightforward to relativize.

#### Theorem 2.7 (Relativized normal form)

Let  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  be a class. Let  $Z \in 2^{\mathbb{N}}$ .

1.  $\mathcal{A}$  is  $\Sigma_1^1(Z)$  iff there is a functional  $\Phi_e$  such that  $(X, n) \in \mathcal{A}$  iff  $\exists f \Phi_e(f, Z, X, n) \uparrow$ .
2.  $\mathcal{A}$  is  $\Pi_1^1(Z)$  iff there exists a functional  $\Phi_e$  such that  $(X, n) \in \mathcal{A}$  iff  $\forall f \Phi_e(f, Z, X, n) \downarrow$ .

Just like for Borel classes, the  $\Sigma_1^1/\Pi_1^1$  classes also have a non-effective version. It is these non-effective versions that were first discovered and studied by Mikhail Souslin [218] in 1917.

**Definition 2.8.** A class  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  is  $\Sigma_1^1$  if  $\mathcal{A}$  is  $\Sigma_1^1(Z)$  for some  $Z \in 2^{\mathbb{N}}$ . Likewise  $\mathcal{A}$  is  $\Pi_1^1$  if  $\mathcal{A}$  is  $\Pi_1^1(Z)$  for some  $Z \in 2^{\mathbb{N}}$ . ◇

The definition we just gave is not the original one: the  $\Sigma_1^1$  or *analytic* classes, studied by Souslin were the images of Borel classes by continuous functions. We will see in Section 30-1 that the  $\Sigma_1^1(X)$  classes indeed admit a canonical representation which makes them the images of  $\mathbb{N}^{\mathbb{N}}$  by functions computable in  $\mathcal{O}^X$ . The  $\Pi_1^1$  classes are also qualified as *co-analytic*.

**Exercise 2.9.** Show that if  $X$  is  $\Sigma_1^1(Y)$  and  $Y$  is  $\Delta_1^1(Z)$  then  $X$  is  $\Sigma_1^1(Z)$ .

◇

### 2.4. Separations

We finally show that the  $\Sigma_1^1/\Pi_1^1$  sets/classes are distinct. These results of separations are fundamental, in particular in the light of Theorem 5.2 and of forthcoming Corollary 5.15, which will then provide us with a separation between the Borel classes and the  $\Sigma_1^1/\Pi_1^1$  classes.

**Theorem 2.10**

*There are  $\Pi_1^1$  sets which are not  $\Sigma_1^1$  and vice versa.*

PROOF. Let  $A \subseteq \mathbb{N}$  be the following  $\Pi_1^1$  class:  $n \in A$  iff  $\forall f \Phi_n(f, n) \downarrow$ . Suppose by contradiction that the complement of  $A$  is  $\Pi_1^1$ . Then, according to Kleene's normal form there exists a code  $e$  such that  $n \in \mathbb{N} \setminus A$  iff  $\forall f \Phi_e(f, n) \downarrow$ . We therefore have  $e \in \mathbb{N} \setminus A$  iff  $\forall f \Phi_e(f, e) \downarrow$  iff  $e \in A$  which is a contradiction. So  $A$  is  $\Pi_1^1$  but not  $\Sigma_1^1$ . Its complement is therefore  $\Sigma_1^1$  but not  $\Pi_1^1$ . ■

The proof of the previous theorem is very similar to that showing  $\emptyset'$  is a  $\Sigma_1^0$  set which is not  $\Pi_1^0$ . This will be reinforced in Section 4 with the analogy between  $\Pi_1^1$  sets and c.e. sets.

**Theorem 2.11**

*There are  $\Pi_1^1$  classes which are not  $\underline{\Sigma}_1^1$  and  $\Sigma_1^1$  classes which are not  $\underline{\Pi}_1^1$ .*

PROOF. Let  $\mathcal{A}$  be the  $\Pi_1^1$  class equal to  $\{1^n 0X : \forall f \Phi_n(f, X, 1^n 0X) \downarrow\}$ . Suppose by contradiction that  $\mathcal{A}$  is  $\underline{\Sigma}_1^1$  that is to say that  $2^{\mathbb{N}} \setminus \mathcal{A}$  is  $\underline{\Pi}_1^1$ . Then, there exists  $Y$  such that  $2^{\mathbb{N}} \setminus \mathcal{A}$  is  $\Pi_1^1(Y)$  and therefore there exists  $e$  such that  $2^{\mathbb{N}} \setminus \mathcal{A} = \{X : \forall f \Phi_e(f, Y, X) \downarrow\}$ . Now we have  $1^e 0Y \in \mathcal{A}$  iff  $\forall f \Phi_e(f, Y, 1^e 0Y) \downarrow$  iff  $1^e 0Y \in 2^{\mathbb{N}} \setminus \mathcal{A}$  which is a contradiction. So  $\mathcal{A}$  is not  $\underline{\Sigma}_1^1$  and by complement  $2^{\mathbb{N}} \setminus \mathcal{A}$  is a  $\Sigma_1^1$  class which is not  $\underline{\Pi}_1^1$ . ■

### 3. The $\Pi_1^1$ sets and the well-orders

We see in this section the close link between  $\Pi_1^1$  sets and the notion of well-order.

**Theorem 3.1**

*The set  $\mathcal{O}^X$  is  $\Pi_1^1(X)$  uniformly in  $X$ .*

PROOF. We show the theorem for  $\mathcal{O}$ . The proof is straightforward to uniformly relativize to an oracle  $X$ . For this proof, we extend the order  $<_o$  to as many integers as we can. For any integer  $a$  we write  $a <_o 2^a$ . For any integer  $e$ , if  $\Phi_e(n) \downarrow$  we will write  $\Phi_e(n) <_o 3 \cdot 5^e$ . We then close the relation  $<_o$  transitively (note that cycles can appear and that  $<_o$  is no longer an order relation on certain subsets of integers).

We define a first  $\Pi_2^0$  condition for an element  $a$  to belong to  $\mathcal{O}$ : let  $A$  the set containing  $a$  as well as the elements  $b <_o a$ . The condition:

- (1) The relation  $<_o$  is a total order once restricted to elements of  $A$ , which are all equal to 1, or to  $2^b$  for some  $b$ , or  $3 \cdot 5^e$  for some code  $e$  such that  $\Phi_e$  is total with  $\Phi_e(n) <_o \Phi_e(n+1)$  for all  $n$ .

This condition is not sufficient: it could be that a code  $a$  satisfies it but is such that the enumeration of  $A$  contains an infinite sequence of smaller and smaller elements:  $\dots <_o a_3 <_o a_2 <_o a_1 <_o a$ . It is then necessary in addition to satisfy the following  $\Pi_1^1$  condition.

- (2)  $\forall B \subseteq A$  if  $B$  is non-empty then  $B$  contains a smallest element for  $<_o$ .

It is clear that any element of  $\mathcal{O}$  satisfies (1) and (2). Conversely, let us show that if  $a$  satisfies (1) and (2), then  $a \in \mathcal{O}$ . Let us reason by contradiction and suppose that  $a \notin \mathcal{O}$ . As  $a \in A$ ,  $A \setminus \mathcal{O} \neq \emptyset$ , so by (2),  $A \setminus \mathcal{O}$  has a smallest element, which we denote by  $d$ . By (1), either  $d = 1$ , or  $d$  is of the form  $2^b$ , or of the form  $3 \cdot 5^e$  with  $\Phi_e(n) <_o \Phi_e(n+1)$  for all  $n$ . As  $1 \in \mathcal{O}$ ,  $d$  necessarily follows one of the other two forms.

- If  $d = 2^b$ . As  $b <_o 2^b$  by definition of  $<_o$ , then  $b \in A$  by downward-closure of  $A$ . By minimality of  $d$ ,  $b \in \mathcal{O}$ . But then  $2^b \in \mathcal{O}$ , which contradicts  $d \notin \mathcal{O}$ .
- If  $d = 3 \cdot 5^e$ , then as for all  $n$ ,  $\Phi_e(n) <_o 3 \cdot 5^e$  by definition of  $<_o$ ,  $\Phi_e(n) \in A$  by downward-closure of  $A$ . By minimality of  $d$ ,  $\Phi_e(n) \in \mathcal{O}$  for all  $n$ . Then,  $3 \cdot 5^e \in \mathcal{O}$ , which contradicts  $d \notin \mathcal{O}$ .

In both cases, we get a contradiction. Thus, for any  $a$  which satisfies (1) and (2),  $a \in \mathcal{O}$ . ■

### Theorem 3.2

Let  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$ . Then,  $\mathcal{A}$  is  $\Pi_1^1$  iff there exists  $h : \mathbb{N} \rightarrow \mathbb{N}$  total computable such that  $(X, n) \in \mathcal{A}$  iff  $h(n) \in \mathcal{O}^X$ .

PROOF. Suppose  $(X, n) \in \mathcal{A}$  iff  $h(n) \in \mathcal{O}^X$ . Using Theorem 3.1 we easily give a  $\Pi_1^1$  description of the class  $\mathcal{A}$ . Let us now show the converse. Let  $\mathcal{A}$  be a  $\Pi_1^1$  set. According to the normal form theorem, there exists a code  $e$  such that  $(X, n) \in \mathcal{A}$  iff  $\forall f \Phi_e(f, X, n) \downarrow$ . Uniformly in  $n, X$  we define a computable tree  $T_n^X$  such that  $\sigma \in T_n^X$  iff  $\Phi_e(\sigma, X, n)[|\sigma|] \uparrow$ . The tree  $T_n^X$  is well-founded iff  $\forall f \forall t [T_n^X] \text{ iff } \forall f \exists t \Phi_e(f \upharpoonright t, X, n)[t] \downarrow$  iff  $(X, n) \in \mathcal{A}$ .

The function  $h$  simply transforms the tree  $T_n^X$  into a computable ordinal code using the Kleene-Brouwer order as used in the proof of Proposition 27-5.18, then transforms the code of a computable ordinal in code of a constructive ordinal using the transformation of computable ordinals into constructive ordinals as performed in the proof of Theorem 27-5.10. Note that

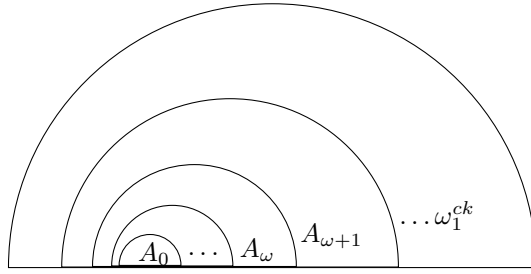


Figure 3.5: Illustration of the fact that a  $\Pi_1^1$  set is an increasing union along  $\omega_1^{ck}$  of  $\Sigma_{\alpha+1}^0$  sets.

if the tree  $T_n^X$  is ill-founded, the transformations performed in each of these proofs remain valid, but simply result in an integer not belonging to  $\mathcal{O}^X$ . ■

### Remark

The previous theorem generalizes the two special cases that interest us:

- A set  $A \subseteq \mathbb{N}$  is  $\Pi_1^1$  iff there exists  $h : \mathbb{N} \rightarrow \mathbb{N}$  total computable such that  $n \in A$  iff  $h(n) \in \mathcal{O}$ .
- A class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is  $\Pi_1^1$  iff there is  $e \in \mathbb{N}$  such that  $X \in \mathcal{A}$  iff  $e \in \mathcal{O}^X$ .

### Corollary 3.3

For all  $X$  the set  $\mathcal{O}^X$  is many-one complete for the  $\Pi_1^1(X)$  sets (we also say  $\Pi_1^1(X)$ -complete).

PROOF. This is a simple reformulation of Theorem 3.2. ■

### Corollary 3.4

For all  $X$  the set  $\mathcal{O}^X$  is  $\Pi_1^1(X)$  but is not  $\Sigma_1^1(X)$ .

PROOF. We show the corollary for  $\mathcal{O}$ , the proof straightforwardly relativizes to  $\mathcal{O}^X$  for all  $X$ . We easily show that any many-one set reducible to a  $\Pi_1^1$  set is also  $\Pi_1^1$ , and that any set many-one reducible to a  $\Sigma_1^1$  set is also  $\Sigma_1^1$ . So if  $\mathcal{O}$  is  $\Sigma_1^1$ , then any  $\Pi_1^1$  set is also  $\Sigma_1^1$ , which contradicts Theorem 2.10. ■

We now see two other important corollaries:  $\Pi_1^1$  sets/classes are uniform unions of Borel classes indexed by countable ordinals.

**Corollary 3.6**

Let  $A \subseteq \mathbb{N}$  be a  $\Pi_1^1$  set. Then, for all  $\alpha < \omega_1^{ck}$ , uniformly in a code of  $\alpha$  we can find the code of a  $\Sigma_{\alpha+1}^0$  set  $A_\alpha$  such that  $A = \bigcup_{\alpha < \omega_1^{ck}} A_\alpha$ .

PROOF. From Theorem 3.2 we have a computable function  $h : \mathbb{N} \rightarrow \mathbb{N}$  such that  $n \in A$  iff  $h(n) \in \mathcal{O}$ . In particular  $A = \bigcup_{\alpha < \omega_1^{ck}} \{n : h(n) \in \mathcal{O}_{<\alpha}\}$ . According to Theorem 28-1.10 the set  $\mathcal{O}_{<\alpha}$  is uniformly Turing reducible to a  $\Sigma_\alpha^0$  set and is therefore  $\Delta_{\alpha+1}^0$ . The set  $\{n : h(n) \in \mathcal{O}_{<\alpha}\}$  is in particular  $\Sigma_{\alpha+1}^0$ . ■

The preceding corollary is remarkable: we have succeeded in transforming a universal quantification over the Baire space functions, into an existential quantification over the computable ordinals. In particular, we go from a quantification having the cardinality of the continuum to a countable quantification. The following corollary follows the same idea but for classes.

**Corollary 3.7**

Let  $\mathcal{A} \subseteq 2^\mathbb{N}$  be a  $\Pi_1^1$  class. Then, for all  $\alpha < \omega_1$ , uniformly in an oracle  $X$  encoding  $\alpha$  we can find the code of a  $\Sigma_{\alpha+1}^0(X)$  class  $\mathcal{A}_\alpha$  such that  $\mathcal{A} = \bigcup_{\alpha < \omega_1} \mathcal{A}_\alpha$ .

PROOF. According to Theorem 3.2 we have a code  $e$  such that  $Y \in \mathcal{A}$  iff  $e \in \mathcal{O}^Y$ . In particular  $\mathcal{A} = \bigcup_{\alpha < \omega_1} \{Y : e \in \mathcal{O}_{<\alpha}^Y\}$ . According to Theorem 28-4.5, for all  $\alpha < \omega_1$ , for all  $X$  such that  $\alpha < \omega_1^X$  and for all  $a \in \mathcal{O}_{=\alpha}^X$ , the set  $\{Y : e \in \mathcal{O}_{<\alpha}^Y\}$  is  $\Sigma_{\alpha+1}^0(X)$  uniformly in  $X$  and  $a$ . ■

**Exercise 3.9. (★★) (Lusin).** Lebesgue's measure, well defined on Borel classes, can be extended as follows: a class  $\mathcal{A} \subseteq 2^\mathbb{N}$  is *measurable* if it is equal to  $\mathcal{B} \cup \mathcal{C}$  where  $\mathcal{B}$  is a Borel class and  $\mathcal{C}$  is included in a Borel class of measure zero. The measure of  $\mathcal{A}$  is then defined as being that of  $\mathcal{B}$ . Show that any  $\Pi_1^1$  class — and therefore also any class  $\Sigma_1^1$  — is measurable.

Hint: Start by showing that there exists a countable ordinal  $\alpha$  such that the class of sets  $X$  computing a code for  $\alpha$  is of zero measure. ◇

## 4. Analogies between $\Pi_1^1$ sets and c.e. sets

There is an analogy between the  $\Pi_1^1$  sets and the c.e. sets. To begin with it should be fairly clear, via Corollary 3.3, that Kleene's  $\mathcal{O}$  is a version of the halting problem for  $\Pi_1^1$  predicates. The analogy goes further than

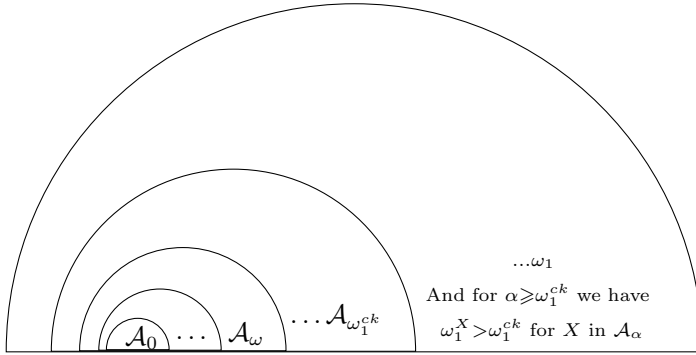


Figure 3.8: Illustration of the fact that a  $\Pi_1^1$  class is an increasing union along  $\omega_1$  of  $\Sigma_{\alpha+1}^0(Y)$  classes in any  $Y$  computing  $\alpha$ . Note that for  $X \in \mathcal{A} \setminus \mathcal{A}_{\omega_1^{ck}}$ , we necessarily have  $\omega_1^X > \omega_1^{ck}$ .

that: it is possible to consider the  $\Pi_1^1$  sets as actually being computably enumerable, but with computation times extending along the computable ordinals. This way of seeing things follows directly from Corollary 3.6: given  $A = \bigcup_{\alpha < \omega_1^{ck}} A_\alpha$  we can see the elements of  $A_{\alpha+1} \setminus A_\alpha$  as being the elements “enumerated” in  $A$  at step  $\alpha+1$ . This idea will be stated precisely with Theorem 4.3.

We start with a theorem characteristic of this analogy between  $\Pi_1^1$  sets and c.e. set : suppose that we have a sequence  $(A_n)_{n \in \mathbb{N}}$  of subsets of  $\mathbb{N}$  with each  $A_n$  being c.e. uniformly in  $n$ . Then, it is possible to define a partial computable function such that  $A_n \neq \emptyset$  implies  $f(n) \in A_n$ : it suffices for all  $n$  to define  $f(n)$  as being the first element enumerated in  $A_n$ . We can do exactly the same thing in the  $\Pi_1^1$  case, but this time considering the elements as being enumerated along ordinal computation times.

**Theorem 4.1 (Kreisel’s  $\Pi_1^1$  uniformisation)**

Let  $A \subseteq \mathbb{N} \times \mathbb{N}$  be a  $\Pi_1^1$  class. Then, there exists a partial  $\Pi_1^1$  function (that is to say of  $\Pi_1^1$  graph)  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that:

$$\forall n (\exists m (n, m) \in A \rightarrow (n, f(n)) \in A)$$

PROOF. Using Corollary 3.6, let  $A = \bigcup_{\alpha < \omega_1^{ck}} A_\alpha$  where each  $A_\alpha$  is a  $\Sigma_{\alpha+1}^0$  set uniformly in a code of  $\alpha$ .

We define  $f$  as follows:  $f(n) = x$  if  $x$  is the smallest element such that  $(n, x) \in A_\alpha$  for  $\alpha$  the smallest ordinal such that  $\{y : (n, y) \in A_\alpha\}$  is non-empty. Formally  $f(n) = x$  if there exists  $\alpha < \omega_1^{ck}$  such that  $(n, x) \in A_\alpha$ , such that  $(n, y) \notin A_\beta$  for all  $y$  and for all  $\beta < \alpha$ , and such that  $(n, y) \notin A_\alpha$  for all  $y < x$ .

To see that it is indeed a  $\Pi_1^1$  definition it is necessary to start by going through the encodings of the ordinals by elements of  $\mathcal{O}$ . We then anticipate a bit on Theorem 5.1 to come to take advantage of the fact that predicates of the form  $x \in A$  for a  $\Sigma_\alpha^0$  set  $A$  are uniformly  $\Delta_1^1$ : given a many-one reduction  $h$  from  $A$  to  $H_{2^a}$  for  $a \in \mathcal{O}_{=\alpha}$  we have  $x \in A$  iff  $\exists X \mathcal{P}(2^a, X) \wedge h(x) \in X$  iff  $\forall X \mathcal{P}(2^a, X) \rightarrow h(x) \in X$ . We use here the  $\Pi_2^0$  class  $\mathcal{P}$  of Theorem 28-2.1.

As it is the first example of this kind we give here the formal  $\Pi_1^1$  definition of our function  $f$ . In the following we will allow ourselves the  $\Pi_1^1$  descriptions of the form  $\exists \alpha < \omega_1^{ck} \dots$ . For that, let  $h$  be the computable function such that  $h(a, e, n, x) \in H_{2^a}$  iff  $(n, x)$  belongs to the  $\Sigma_{|a|}^0$  set of code  $e$ . Let  $g$  be the computable function such that  $g(a)$  is the  $\Sigma_{|a|}^0$ -code of  $A_{|a|}$  for all  $a \in \mathcal{O}$ . Finally, given a set  $X$  which we suppose to be equal to  $H_{2^a}$ , we denote by  $X_b$  for  $b <_o a$  the set obtained in a computable way from  $X$ , which is equal to  $H_{2^b}$  if  $X = H_{2^a}$ . We then have  $f(n) = x$  if:

$$\exists a \ a \in \mathcal{O} \wedge \forall X \left( \mathcal{P}(2^a, X) \Rightarrow \left( \begin{array}{l} h(a, g(a), n, x) \in X \\ \wedge \ \forall b <_o a \ \forall y \ h(b, g(b), n, y) \notin X_b \\ \wedge \ \forall y < x \ h(a, g(a), n, y) \notin X \end{array} \right) \right)$$

■

Note that this proof of the uniformization theorem also works for  $\Pi_1^1$  classes  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times \mathbb{N}$  and partial  $\Pi_1^1$  functionals  $f : 2^{\mathbb{N}} \rightarrow \mathbb{N}$  which on an oracle  $X$  such that  $\{n : (X, n) \in \mathcal{A}\}$  is not empty, return an integer such that  $(X, f(X)) \in \mathcal{A}$ . We will see with Corollary 30-5.3 — and it is much more difficult to show — that the  $\Pi_1^1$  uniformization theorem even works for  $\Pi_1^1$  classes  $\mathcal{A} \subseteq 2^{\mathbb{N}} \times 2^{\mathbb{N}}$ .

In the meantime, let's continue on the analogy between being  $\Pi_1^1$  and c.e. Let us take our example of a c.e. set  $(A_n)_{n \in \mathbb{N}}$  and of our partial computable function such that  $A_n \neq \emptyset$  implies  $f(n) \in A_n$ . Suppose for a moment that no set  $A_n$  is empty. Then, our function becomes a total computable function and therefore can be described in a  $\Delta_1^0$  way. In fact any total function of c.e. graph admits a computable graph, since to decide if  $f(n) = x$  it suffices to look for the unique  $y$  such that  $f(n) = y$  and to check whether  $x = y$ . The same goes for the graph of  $\Pi_1^1$  functions:

**Proposition 4.2.** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be a total  $\Pi_1^1$  function. Then,  $f$  is  $\Delta_1^1$ . ★

PROOF. It suffices to see that the relation  $f(n) \neq x$  is equivalent to  $\exists y \neq x \ f(n) = y$ . The relation  $f(n) \neq x$  is therefore also  $\Pi_1^1$ , which makes the function  $\Delta_1^1$ . ■

We will see in the next section that the  $\Delta_1^1$  sets coincide with the hyperarithmetical sets. In the meantime, here is the formal aspect of the fact that  $\Pi_1^1$  can be seen as an analogue of being c.e.

**Theorem 4.3**

*For any non-empty  $\Pi_1^1$  set  $A \subseteq \mathbb{N}$ , there exists a total  $\Pi_1^1$  function  $f : \mathcal{O} \rightarrow \mathbb{N}$  such that for any  $\mathcal{O}_1 \subseteq \mathcal{O}$  where each computable ordinal has exactly one code in  $\mathcal{O}_1$ , we have  $A = \{f(a) : a \in \mathcal{O}_1\}$ .*

PROOF. This is an improvement of the proof of the Uniformization Theorem. Let  $A = \bigcup_{\alpha < \omega_1^{ck}} A_\alpha$  and let  $y \in A$  be a fixed element. We define the function  $f : \mathcal{O} \rightarrow \mathbb{N}$  as follows. On an element  $a \in \mathcal{O}$  encoding an ordinal  $\alpha$  the function starts by computing the code of an ordinal  $\beta$  and an integer  $n$  such that  $\alpha = \beta \times \omega + n$ . Then the function returns the  $n$ -th element of  $A_\beta$  if it exists. Otherwise,  $f(a) = y$ .

It suffices to show that (1) for each ordinal  $\alpha$  there exists a unique ordinal  $\beta$  and a unique integer  $n$  such that  $\alpha = \beta \times \omega + n$ , and (2) for all  $\beta < \omega_1^{ck}$  the ordinal  $\beta \times \omega + n < \omega_1^{ck}$ . We leave this basic evidence to the reader.

The function  $f$  is well defined by the uniqueness of  $\beta$  and  $n$ , and is total because  $f(a) = y$  if  $A_\beta$  does not have  $n$ -th element. Moreover, for all  $a \in \mathcal{O}$ ,  $f(a) \in A$ .

Let  $\mathcal{O}_1 \subseteq \mathcal{O}$  be a set containing exactly one code of each computable ordinal. Let us show that  $A \subseteq \{f(a) : a \in \mathcal{O}_1\}$ . Let  $z \in A$  and let  $\beta < \omega_1^{ck}$  and  $n \in \mathbb{N}$  such that  $z$  is the  $n$ -th element of  $A_\beta$ . By (2),  $\beta \times \omega + n < \omega_1^{ck}$ , so there exists an  $a \in \mathcal{O}_1$  such that  $|a| = \beta \times \omega + n$ , so  $f(a) = z$ . Thus,  $A = \{f(a) : a \in \mathcal{O}_1\}$  ■

The analogy between being c.e. and being  $\Pi_1^1$  is not just formal. In 1989 Joel David Hamkins and Jeff Kidder imagine a Turing machine where the computing time can be any ordinal. Concretely, the Turing machine has the same behavior as usual at the successor computation steps. At limit steps, each cell of the machine takes as a value the lower limit of the values in the previous steps, the machine enters a special “limit” state and the head returns to the first cell. The founding article [81] on the study of these infinite-time Turing machines will only come later, and their authors show in particular that the real numbers enumerable by such machines, in ordinal time less than  $\omega_1^{ck}$ , are exactly the  $\Pi_1^1$  sets.

## 5. Kleene/Souslin equivalence theorem

We see in this section a fundamental theorem: the  $\Delta_1^1$  and hyperarithmetical sets/classes coincide. This result dates back to the work of Souslin at the beginning of the 20th century. The notion of computability was then in its infancy, also Souslin's result relates to the  $\Sigma_1^1$  classes, which he himself had discovered a little earlier, via the famous "Lebesgue's error" which we have already spoken of in the historical interlude at the beginning of the chapter.

### 5.1. The equivalence theorem for sets

We start by showing the simple direction: if a set is hyperarithmetical, then it is  $\Delta_1^1$ .

**Theorem 5.1 (Kleene [115])**  
*The hyperarithmetical sets are  $\Delta_1^1$ .*

PROOF. Let  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}}$  be the  $\Pi_2^0$  class of Theorem 28-2.1, such that for  $a \in \mathcal{O}$  the class  $\{X : \mathcal{P}(a, X)\}$  is the  $\Pi_2^0$  class containing only  $H_a$ . Let  $X$  be hyperarithmetical. By Theorem 28-1.12 let  $e \in \mathbb{N}$  and  $a \in \mathcal{O}$  such that  $\Phi_e(H_a) = X$ . So we have

- (1)  $n \in X$  iff  $\exists Y \mathcal{P}(a, Y)$  and  $\exists \sigma \prec Y \Phi_e(\sigma, n) \downarrow = 1$ .
- (2)  $n \notin X$  iff  $\exists Y \mathcal{P}(a, Y)$  and  $\exists \sigma \prec Y \Phi_e(\sigma, n) \downarrow = 0$ .

We have a  $\Sigma_1^1$  description of  $X$  and its complement. ■

Moreover, note that from the  $\Sigma_\alpha^0$ -code of a set  $A$  and an element  $a \in \mathcal{O}_{=\alpha}$ , we can uniformly obtain a  $\Delta_1^1$  code for  $A$ , that is, a pair of a  $\Sigma_1^1$  and a  $\Pi_1^1$  codes for  $A$ . For this, we use the uniform many-one reduction from the  $\Sigma_\alpha^0$  sets to the  $H_a$  sets for  $a \in \mathcal{O}_{\alpha+1}$ .

In particular one can obtain uniformly in  $a$  a  $\Delta_1^1$  code of the set  $\mathcal{O}_{<a}$ , which will be used for the next lemma.

**Theorem 5.2 (Kleene)**  
*A set  $X \subseteq \mathbb{N}$  is  $\Delta_1^1$  iff it is hyperarithmetical.*

We already have the direction  $X$  hyperarithmetical implies  $X \Delta_1^1$ . The other direction is more subtle, and calls upon Spector's  $\Sigma_1^1$  *boundedness lemma*.

**Lemma 5.3 ( $\Sigma_1^1$  boundedness, Spector [213]).** Let  $A \subseteq \mathcal{O}$  be a  $\Sigma_1^1$  set. Then,  $\sup_{a \in A} |a| < \omega_1^{ck}$ . ★

PROOF. Suppose instead that  $\sup_{a \in A} |a| = \omega_1^{ck}$ . Then, we can give the following  $\Sigma_1^1$  description of  $\mathcal{O}$ , in contradiction with Corollary 3.4:

$$a \in \mathcal{O} \text{ iff } \exists b \in A \text{ such that } a \in \mathcal{O}_{<b}.$$

As one can obtain a  $\Sigma_1^1$  code for each set  $\mathcal{O}_{<b}$  uniformly in  $b$ , the above description is indeed  $\Sigma_1^1$ , which contradicts the fact that  $\mathcal{O}$  does not admit a  $\Sigma_1^1$  description. So  $\sup_{a \in A} |a| < \omega_1^{ck}$ . ■

PROOF OF THEOREM 5.2. According to Theorem 5.1 the hyperarithmetic sets are  $\Delta_1^1$ . Now suppose that  $A \subseteq \mathbb{N}$  is  $\Delta_1^1$ . Since  $A$  is  $\Pi_1^1$ , there exists a computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $n \in A$  iff  $f(n) \in \mathcal{O}$ . Since  $A$  is  $\Sigma_1^1$ , the set  $\{f(n) : n \in A\}$  is  $\Sigma_1^1$ . According to the  $\Sigma_1^1$  boundedness lemma there exists an ordinal  $\alpha < \omega_1^{ck}$  such that  $n \in A$  iff  $f(n) \in \mathcal{O}_{<\alpha}$ . Since  $\mathcal{O}_{<\alpha}$  is hyperarithmetic then  $A$  is also hyperarithmetic. ■

## 5.2. First consequence

The Kleene/Souslin equivalence allows us to complete Theorem 28-2.4 which states that arithmetic sets all have a modulus. Recall that a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is a modulus of a set  $A \subseteq \mathbb{N}$  if any function dominating  $f$  computes  $A$ .

### Theorem 5.4 (Groszek and Slaman [78])

*A set  $X \subseteq \mathbb{N}$  is hyperarithmetic iff it admits a modulus.*

The first direction is Theorem 28-2.4. The other direction of the theorem requires the following lemma of independent interest.

**Lemma 5.5.** Let  $X$  be a set admitting a modulus. Then,  $X$  admits a uniform modulus: there exists a code  $e$  and a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that for any function  $g \geq f$  we have  $\Phi_e(g) = X$ . ★

PROOF. Let  $f$  be a function such that for any  $g \geq f$  we have  $g \geq_T X$ . Suppose by contradiction that  $X$  does not admit a uniform modulus, that is to say that for any  $e$  and any function  $g$  there exists a function  $h \geq g$  such that  $\Phi_e(h) \neq X$ . We then build by forcing a function  $g \geq f$  such that  $g \not\geq_T X$ , contradicting the fact that  $f$  is a modulus.

We introduce for the construction the following notation: given  $\sigma \in \mathbb{N}^{<\mathbb{N}}$  and  $g \in \mathbb{N}^{\mathbb{N}}$ , we denote by  $\sigma g$  the function resulting from the concatenation

of  $\sigma$  and  $g$ , that is to say the function which to  $n < |\sigma|$  associates  $\sigma(n)$  and which to  $n \geq |\sigma|$  associates  $g(n - |\sigma|)$ .

Our conditions  $\mathbb{P}$  are simply pairs  $\langle \sigma, g \rangle$  such that  $\sigma g \geq f$ . The extension of our forcing conditions is defined by  $\langle \tau, h \rangle \preceq \langle \sigma, g \rangle$  if  $\tau \succeq \sigma$  and  $\tau h \geq \sigma g$ . Moreover, given  $\langle \sigma, g \rangle \in \mathbb{P}$  and  $\tau \in \mathbb{N}^{<\mathbb{N}}$ , we write  $\tau \in [\sigma, g]$  if  $\tau \succeq \sigma$  and  $\tau(n) \geq (\sigma g)(n)$  for all  $n < |\tau|$ .

Given a condition  $\langle \sigma, g \rangle \in \mathbb{P}$  and a code  $e$  we affirm that one of the two following possibilities necessarily occurs:

- (1) There is  $m \in \mathbb{N}$  and there is  $\tau \in [\sigma, g]$  such that  $\Phi_e(\tau, m) \downarrow \neq X(m)$
- (2) There exists  $\langle \tau, h \rangle \preceq \langle \sigma, g \rangle$  and there is  $m \in \mathbb{N}$  such that for all  $\rho \in [\tau, h]$  we have  $\Phi_e(\rho, m) \uparrow$

Suppose that neither (1) nor (2) happens, then we have the conjunction of the following two statements:

- (3) For all  $m \in \mathbb{N}$  and for all  $\tau \in [\sigma, g]$  we have  $\Phi_e(\tau, m) \downarrow$  implies  $\Phi_e(\tau, m) = X(m)$
- (4) For all  $\langle \tau, h \rangle \preceq \langle \sigma, g \rangle$  and for all  $m \in \mathbb{N}$  there exists  $\rho \in [\tau, h]$  such that  $\Phi_e(\rho, m) \downarrow$

There then exists a unique functional  $\Phi$  such that for any function  $h \geq \sigma g$  we have  $\Phi(h) = X$ . Given such  $h$  and an integer  $m$ , the functional searches  $\rho \succeq \sigma$  with  $\rho(n) \geq h(n)$  for all  $|\sigma| \leq n < |\rho|$  and such that  $\Phi_e(\rho, m) \downarrow = i$ . Then the functional returns  $i$ . According to (4) such research is necessarily successful. According to (3) the value  $i$  necessarily equals  $X(m)$ .

As the existence of such a functional contradicts our hypotheses, we can therefore for each code  $e$  and each forcing condition find an extension satisfying (1) or (2). In each case we make sure that our generic will not compute  $X$  via  $\Phi_e$ . The generic will then consist of a function  $g \geq f$  such that  $X \not\leq_T g$  which contradicts the fact that  $f$  is a modulus for  $X$ .

So if  $X$  admits a modulus it admits a uniform modulus. ■

**PROOF OF THEOREM 5.4.** We have seen with Theorem 28-2.4 that any hyperarithmetic set admits a modulus. Suppose now that  $X$  admits a modulus. According to Lemma 5.5,  $X$  admits a uniform modulus via a functional  $\Phi_e$ . In what follows we write  $\sigma \geq f$  to mean  $\sigma(m) \geq f(m)$  for all  $m < |\sigma|$ . Let  $\Psi(n)$  be the  $\Sigma_1^1$  predicate given by

$$\exists f \text{ such that } \forall \sigma \geq f \text{ we have } \Phi_e(\sigma, n) \downarrow \rightarrow \Phi_e(\sigma, n) = 1.$$

If  $n \in X$  then  $\Psi(n)$  is clearly true by taking  $f$  as being the witness of our modulus. If now  $\Psi(n)$  is true for some function  $f$ , then by taking  $g > f$

large enough such that  $\Phi_e(g) = X$  then we have  $\Phi_e(\sigma, n) \downarrow = X(n)$  for some  $\sigma \prec g$ . Since  $\Psi(n)$  is true with  $f$  we actually have  $\Phi_e(\sigma, n) \downarrow = 1$  and therefore  $n \in X$ . It follows that  $n \in X$  iff  $\Psi(n)$  is true. So  $X$  is  $\Sigma_1^1$ . We proceed in the same way to show that the complement of  $X$  is  $\Sigma_1^1$ , which implies that  $X$  is  $\Delta_1^1$  and therefore hyperarithmetical. ■

### 5.3. Uniformity of the equivalence theorem

We have seen that it is possible to obtain the  $\Delta_1^1$  code of a set  $A$  from its hyperarithmetical code. On the other hand, Theorem 5.2 does not give us a way to obtain a hyperarithmetical code of a set  $A$  from its  $\Delta_1^1$  code. The crucial point lies in the proof of Spector's  $\Sigma_1^1$  boundedness lemma, which establishes the existence of a bound on a  $\Sigma_1^1$  set of ordinals  $A \subseteq \mathcal{O}$ , but without specifying it.

We see here that it is possible to obtain such a bound uniformly, at the cost of course of a more complex construction. For this, we will use representations of ordinals via well-founded trees in Baire space, and manipulation lemmas on them.

**Definition 5.6.** Let  $T_1, T_2 \subseteq \mathbb{N}^{<\mathbb{N}}$  be two trees. We call *morphism* from  $T_1$  to  $T_2$  a function  $f : T_1 \rightarrow T_2$  such that  $|f(\sigma)| = |\sigma|$  and such that  $\sigma \preceq \tau$  implies  $f(\sigma) \preceq f(\tau)$ . ◇

The notation  $|\sigma|$  above means the length of  $\sigma$ . We start with a first proposition allowing to compare the ordinals encoded by well-founded trees via the existence of a morphism from one to the other. Let us remember for the following proof of the notation  $|T|_n$  introduced in Section 27-5.3.

**Proposition 5.7.** Given two well-founded trees  $T_1, T_2 \subseteq \mathbb{N}^{<\mathbb{N}}$  we have  $|T_1| \leq |T_2|$  iff there is a morphism from  $T_1$  to  $T_2$ . ★

PROOF. We prove the proposition by induction on the ordinals encoded by  $T_1$ . If  $|T_1| = 0$  then  $T_1$  is the empty set and clearly the proposition is true with any tree  $T_2$ . Suppose now the proposition true for any tree  $T_1$  such that  $|T_1| < \alpha$  and any tree  $T_2$ . Consider a tree  $T_1$  with  $|T_1| = \alpha$ .

Suppose first  $|T_1| \leq |T_2|$ . Then, for each node  $n \in T_1$  — that is to say each node of  $T_1$  of length 1 — there is a node  $m_n \in T_2$  such that  $|T_1 \upharpoonright_n| \leq |T_2 \upharpoonright_{m_n}|$  (if there are several such nodes, the smallest is chosen). By induction hypothesis we therefore have morphisms  $f_n : T_1 \upharpoonright_n \rightarrow T_2 \upharpoonright_{m_n}$ . We then define  $f : T_1 \rightarrow T_2$  by  $f(n\sigma) = m_n f_n(\sigma)$  and we can easily verify that it is a morphism from  $T_1$  to  $T_2$ .

Conversely, suppose that there exists a morphism  $f : T_1 \rightarrow T_2$ . Then, for each  $n \in T_1$  the function  $f_n : T_1 \upharpoonright_n \rightarrow T_2 \upharpoonright_{f(n)}$  is a morphism from  $T_1 \upharpoonright_n$

to  $T_2 \upharpoonright_{f(n)}$ . It follows by induction hypothesis that  $|T_1 \upharpoonright_n| \leq |T_2 \upharpoonright_{f(n)}|$ . So  $|T_1| \leq |T_2|$ . ■

We now define computable functions  $\wedge : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  and  $\vee : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  on the pairs of c.e. codes of well-founded trees of  $\mathbb{N}^{<\mathbb{N}}$ .

**Lemma 5.8.** There is a total computable function  $\wedge : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ , which on two codes of c.e. trees  $T_1, T_2$  returns the code of a c.e. tree  $T_1 \wedge T_2$  well-founded iff  $T_1$  and  $T_2$  are well-founded, in which case we have  $|T_1 \wedge T_2| \geq \sup(|T_1|, |T_2|)$ . ★

PROOF. We simply define  $T_1 \wedge T_2$  as being the disjoint union of  $T_1$  and  $T_2$ . Concretely we put in  $T_1 \wedge T_2$  the nodes  $\langle n, i \rangle \sigma$  for all  $n\sigma \in T_i$  for  $i \in \{1, 2\}$ . The existence of a morphism from  $T_1$  or  $T_2$  into  $T_1 \wedge T_2$  is clear. We therefore have  $|T_1 \wedge T_2| \geq \sup(|T_1|, |T_2|)$ . ■

**Lemma 5.9.** There exists a total computable function  $\vee$ , which on two codes of c.e. trees  $T_1, T_2$  returns the code of a c.e. tree  $T_1 \vee T_2$  which is well-founded iff  $T_1$  or  $T_2$  is well-founded, in which case  $|T_1 \vee T_2| \geq \min(|T_1|, |T_2|)$ . ★

PROOF. In what follows we use a pairing function on the strings of  $\mathbb{N}^{<\mathbb{N}}$  of the same length, defined by successively applying the bijection  $\langle \rangle : \mathbb{N}^2 \rightarrow \mathbb{N}$  on each pair of integers located at the same position in both strings. Formally:  $\langle \sigma_1, \sigma_2 \rangle = \langle \sigma_1(0), \sigma_2(0) \rangle \wedge \dots \wedge \langle \sigma_1(n-1), \sigma_2(n-1) \rangle$ , where  $n = |\sigma_1| = |\sigma_2|$ .

The definition of  $T_1 \vee T_2$  is simple: we enumerate a node  $\sigma$  in  $T_1 \vee T_2$  if  $\sigma = \langle \sigma_1, \sigma_2 \rangle$  where for  $i \in \{1, 2\}$ , each string  $\sigma_i$  has been enumerated in  $T_i$ . It is clear that we have an infinite path in  $T_1 \vee T_2$  iff we have an infinite path in  $T_1$  and  $T_2$ .

To show  $|T_1 \vee T_2| \geq \min(|T_1|, |T_2|)$  we consider without loss of generality  $|T_1| \leq |T_2|$ . Let  $f : T_1 \rightarrow T_2$  be a morphism (note that in the case  $T_2$  is ill-founded, a morphism always exists by sending all the nodes of  $T_1$  to the infinite path of  $T_2$ ). So each node  $\langle \sigma, f(\sigma) \rangle$  belongs to  $T_1 \vee T_2$ . We can then define the morphism  $g : T_1 \rightarrow |T_1 \vee T_2|$  which to  $\sigma \in T_1$  associates  $\langle \sigma, f(\sigma) \rangle$ . It follows that  $|T_1 \vee T_2| \geq |T_1|$ . ■

We will also need to combine infinite sequences of trees.

**Lemma 5.10.** There exists a total computable function which for each code enumerating an infinite sequence of c.e. trees  $(T_i)_{i \in \mathbb{N}}$ , returns the

code of a c.e. tree  $\bigwedge_{i \in \mathbb{N}} T_i$  which is well-founded iff each  $T_i$  is well-founded, in which case we have  $|\bigwedge_{i \in \mathbb{N}} T_i| \geq \sup_i |T_i|$ . ★

PROOF. We simply define  $T$  as the disjoint union of  $T_i$ . Concretely one puts in  $T$  the nodes  $\langle n, i \rangle \sigma$  for all  $n\sigma \in T_i$ . It is clear that we have a morphism from each  $T_i$  to  $|\bigwedge_{i \in \mathbb{N}} T_i|$  and therefore that  $|\bigwedge_{i \in \mathbb{N}} T_i| \geq \sup_i |T_i|$ . ■

**Exercise 5.11. (★)** Show the existence of a total computable function  $\vee$ , which for each code enumerating an infinite sequence of c.e. trees  $(T_i)_{i \in \mathbb{N}}$ , returns the code of a c.e. tree  $\bigvee_{i \in \mathbb{N}} T_i$  which is well-founded iff at least one  $T_i$  is well-founded, in which case we have  $\min_i (|T_i|) + \omega > |\bigvee_{i \in \mathbb{N}} T_i| \geq \min_i (|T_i|)$ . ◇

We can now show the uniform version of Spector's  $\Sigma_1^1$  boundedness lemma.

**Lemma 5.12 (Uniform bound,  $\Sigma_1^1$  boundedness for the integers).**

There exists a computable function such that if  $e$  is the code of a  $\Sigma_1^1$  set  $A \subseteq \mathcal{O}$  then  $f(e) \in \mathcal{O}$  is such that  $\sup_{a \in A} |a| \leq |f(e)|$ . ★

PROOF. Let  $A \subseteq \mathcal{O}$  be a  $\Sigma_1^1$  set. According to Proposition 27-5.18 we have a computable function  $f$  such that  $f(e)$  codes for a c.e. tree  $T_e$  such that  $T_e$  is well-founded iff  $e \in \mathcal{O}$ . Moreover if  $e \in \mathcal{O}$  we have  $|e| \leq |T_e|$ . Using the fact that  $A$  is  $\Sigma_1^1$ , we obtain via Theorem 3.2 and Proposition 27-5.18 — uniformly in a code of  $A$  — a computable function  $g$  such that  $g(e)$  codes for a c.e. tree  $U_e$  such that  $U_e$  is well-founded iff  $e \notin A$ . Note that for all  $e \in \mathbb{N}$ , the tree  $T_e \vee U_e$  is well-founded. Moreover according to Lemma 5.9, if  $e \in A \subseteq \mathcal{O}$  we have  $|e| \leq |T_e \vee U_e|$ .

We then consider the c.e. tree  $T$  of code  $\bigwedge_{e \in \mathbb{N}} (T_e \vee U_e)$ . According to Lemma 5.10 we have  $|T_e \vee U_e| \leq |\bigwedge_{e \in \mathbb{N}} (T_e \vee U_e)|$  for all  $e$  and therefore  $|e| \leq |\bigwedge_{e \in \mathbb{N}} (T_e \vee U_e)|$  for all  $e \in A \subseteq \mathcal{O}$ . We can then transform the code of  $T$  back into a code of  $\mathcal{O}$  using Theorem 27-5.10 to conclude. ■

We easily deduce from the lemma a uniform bound making it possible to transform the  $\Delta_1^1$  code of a set into a hyperarithmetic code.

## 5.4. The equivalence theorem for classes

We now see the Kleene/Souslin equivalence theorem for classes, and this time we directly show the uniform version of the theorem.

**Proposition 5.13.** If a class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is hyperarithmetic then it is  $\Delta_1^1$ . ★

PROOF. Let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  be a  $\Sigma_\alpha^0$  class. Let  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  be the  $\Pi_2^0$  class of theorem 28-3.4, i.e., such that for  $a \in \mathcal{O}^X$  and  $X \in 2^{\mathbb{N}}$  the class  $\{Y : \mathcal{P}(a, X, Y)\}$  is the  $\Pi_2^0(X)$  class containing only  $H_a^X$ .

According to Theorem 28-4.4 we have a code  $e$  such that  $X \in \mathcal{A}$  iff  $e \in H_{2^a}^X$  for  $a \in \mathcal{O}_{=\alpha}$ . Note that we can assume without loss of generality  $\mathcal{O} \subseteq \mathcal{O}^X$  in order to formally satisfy the hypotheses of Theorem 28-3.4. We have the following  $\Sigma_1^1$  definition for  $\mathcal{A}$ :

$$\mathcal{A} = \{X : \exists Y \mathcal{P}(2^a, X, Y) \text{ and } e \in Y\}.$$

And the following  $\Sigma_1^1$  definition for  $2^{\mathbb{N}} \setminus \mathcal{A}$ :

$$2^{\mathbb{N}} \setminus \mathcal{A} = \{X : \exists Y \mathcal{P}(2^a, X, Y) \text{ and } e \notin Y\}.$$

So the class  $\mathcal{A}$  is  $\Delta_1^1$ . ■

Let us now show the  $\Sigma_1^1$  boundedness lemma for classes, due to Spector [213].

**Lemma 5.14 ( $\Sigma_1^1$  boundedness for classes).** Let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  be a  $\Sigma_1^1$  class such that  $X \in \mathcal{A} \rightarrow e \in \mathcal{O}^X$  for some  $e$ . Then, we can find  $a \in \mathcal{O}$  uniformly in  $e$  and in a code of  $\mathcal{A}$  such that  $X \in \mathcal{A} \rightarrow |e|^X < |a|$ . ★

PROOF. We reuse here the pairing function on strings of  $\mathbb{N}^{<\mathbb{N}}$  of the same length, defined in the proof of Lemma 5.9. Let  $U_1^X$  be the c.e. tree well-founded iff  $X \notin \mathcal{A}$ . Let  $U_2^X$  be the c.e. tree well-founded iff  $e \in \mathcal{O}^X$ . We compute uniformly in  $X$  the well-founded tree  $T^X = U_1^X \vee U_2^X$ . We finally compute the following c.e. tree  $T$ : for each string  $\sigma \in 2^{<\mathbb{N}}$  such that a node  $m$  of length 1 is enumerated in  $T^\sigma$  we enumerates the node  $\langle m, b(\sigma) \rangle$  of length 1 in  $T$ , using a bijection  $b : 2^{<\mathbb{N}} \rightarrow \mathbb{N}$ . For any node  $\langle \tau, b(\sigma_1) \dots b(\sigma_n) \rangle$  enumerated in  $T$  with  $|\tau| = n$ , if a string  $\tau m$  is enumerated in  $T^{\sigma_{n+1}}$  for a string  $\sigma_{n+1} \succeq \sigma_n$  we enumerate  $\langle \tau m, b(\sigma_1) \dots b(\sigma_n) b(\sigma_{n+1}) \rangle$  in  $T$ .

The tree  $T$  is well-founded: an infinite path  $g \oplus f$  of  $[T]$  would correspond to an element  $g \in [T^Y]$  for  $b^{-1}(f(0)) \preceq b^{-1}(f(1)) \preceq b^{-1}(f(2)) \preceq \dots \preceq Y$ , contradicting the fact that each  $T^Y$  is well-founded. Moreover for all  $X$  we have a morphism from  $T^X$  into  $T$ , defined for a node  $\tau$  of length  $n$  by  $f(\tau)$  as being  $\tau \oplus b(\sigma_1) \dots b(\sigma_n)$  for the first strings found  $\sigma_1 \preceq \sigma_2 \preceq \dots \preceq \sigma_n \prec X$  such that  $\tau \in T^{\sigma_n}$ . We therefore deduce  $|T^X| \leq |T|$  for all  $X$ . ■

### Theorem 5.15

A class  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is  $\Delta_1^1$  iff it is hyperarithmetical.

PROOF. We have seen with Proposition 5.13 that a hyperarithmetical class is  $\Delta_1^1$ . Now suppose that  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  is a  $\Delta_1^1$  class. As  $\mathcal{A}$  is  $\Pi_1^1$ , according to Theorem 3.2 there is a code  $e$  such that  $X \in \mathcal{A}$  iff  $e \in \mathcal{O}^X$ . Since  $\mathcal{A}$  is  $\Sigma_1^1$ , the class  $\{X : e \in \mathcal{O}^X\}$  is  $\Sigma_1^1$ . According to the  $\Sigma_1^1$  boundedness lemma for

the classes, there exists  $\alpha < \omega_1^{ck}$  such that  $\mathcal{A} = \{X : e \in \mathcal{O}_{<\alpha}^X\}$ . According to Theorem 28-4.5 the class  $\mathcal{A}$  is therefore hyperarithmetical. ■

## 6. Other boundedness theorems

The technique used for the  $\Sigma_1^1$  boundedness lemma is very powerful, and can be exploited to highlight several other remarkable phenomena. Let us present two of them.

### Theorem 6.1

*Let  $T$  be a well-founded  $\Sigma_1^1$  tree. Then,  $|T| < \omega_1^{ck}$ .*

PROOF. Suppose by contradiction  $|T| \geq \omega_1^{ck}$ . We then show that we have a  $\Sigma_1^1$  description of  $\mathcal{O}$ , which is a contradiction.

Let  $f$  be the computable function such that  $f(e)$  is a code of a c.e. tree  $T_e$  and such that  $T_e$  is well-founded iff  $e \in \mathcal{O}$ . We can then give a  $\Sigma_1^1$  description of  $\mathcal{O}$  as follows:  $e \in \mathcal{O}$  iff there exists a morphism  $f$  from  $T_e$  to  $T$ . Given  $f$ , the statement uses only positive  $T$  membership questions (i.e., of the form  $\sigma \in T$  and not of the form  $\sigma \notin T$ ) and is therefore  $\Sigma_1^1$ :

$$\forall \sigma \in T_e \ f(\sigma) \in T \wedge |f(\sigma)| = |\sigma| \wedge \sigma \preceq \tau \text{ implies } f(\sigma) \preceq f(\tau).$$

We deduce that  $\mathcal{O}$  is  $\Sigma_1^1$ , which is a contradiction. ■

The previous theorem is truly fascinating: the smallest non-computable ordinal is the same as the smallest non-hyperarithmetical ordinal! All the power conferred by arbitrary iterations of the jump does not suffice to compute a representation of  $\omega_1^{ck}$ . The proof we have given is not constructive. It is however possible to give a constructive version, but of course with a little more work.

**Exercise 6.2. (★★)** Show that one can uniformly transform a  $\Sigma_1^1$ -code of a well-founded tree  $T$  into a code c.e. well-founded tree  $T'$  such that  $|T'| \geq |T|$ .

Hint: we can take elements of the proof of lemma 5.12, and use the function of Exercise 5.11. ◇

Let us now see our second theorem, which again relies on the same technique. We start with a general lemma.

**Lemma 6.3.** Let  $\mathcal{A}$  be a  $\Sigma_1^1$  class such that  $\forall X \in \mathcal{A} \exists a \in \mathcal{O} \mathcal{B}(a, X)$  where  $\mathcal{B}$  is a  $\Pi_1^1$  predicate. Then, there exists an ordinal  $\alpha < \omega_1^{ck}$  such that  $\forall X \in \mathcal{A} \exists a \in \mathcal{O}_{<\alpha} \mathcal{B}(a, X)$ . ★

PROOF. Suppose by contradiction that for any ordinal  $\alpha < \omega_1^{ck}$  there exists  $X \in \mathcal{A}$  such that  $\forall a \in \mathcal{O}_{<\alpha} \neg \mathcal{B}(a, X)$ . Then, we can give the following  $\Sigma_1^1$  description of Kleene's  $\mathcal{O}$ :  $b \in \mathcal{O}$  iff

$$(1) \quad \exists X \in \mathcal{A} \forall a (a \notin \mathcal{O} \vee \neg \mathcal{B}(a, X) \vee b \in \mathcal{O}_{<|a|})$$

Suppose  $b \in \mathcal{O}$ . By hypothesis there is  $X \in \mathcal{A}$  such that  $\forall a \in \mathcal{O}_{<\text{succ}(|b|)} \neg \mathcal{B}(a, X)$ . We can then easily verify that (1) is true for this  $X$ . Now suppose  $b \notin \mathcal{O}$ . Let  $X \in \mathcal{A}$ . Let  $a \in \mathcal{O}$  be such that  $\mathcal{B}(a, X)$ . As  $b \notin \mathcal{O}$  we also have  $b \notin \mathcal{O}_{<|a|}$ . So for all  $X \in \mathcal{A}$  there exists  $a \in \mathcal{O}$  such that  $\mathcal{B}(a, X)$  and  $b \notin \mathcal{O}_{<|a|}$ . So (1) is false. We therefore have  $b \in \mathcal{O}$  iff (1) is true. So  $\mathcal{O}$  has a  $\Sigma_1^1$  description which is a contradiction. ■

#### Theorem 6.4

Let  $\mathcal{A}$  be a  $\Sigma_1^1$  class containing only hyperarithmetical elements. Then, there exists  $\alpha < \omega_1^{ck}$  such that all the elements of  $\mathcal{A}$  are computable in  $\emptyset^{(\alpha)}$ .

PROOF. It suffices to apply the previous lemma with the following formula:

$$\forall X \in \mathcal{A} \exists a \in \mathcal{O} \text{ such that } \emptyset^{(|a|)} \geq_T X$$

.

■

**Exercise 6.5. (★★)** Let  $\mathcal{A}$  be a  $\Sigma_1^1$  class containing only hyperarithmetical elements. Show that we can uniformly find an ordinal  $\alpha < \omega_1^{ck}$  such that all the elements of  $\mathcal{A}$  are computable by  $\emptyset^{(\alpha)}$ . ◇

## 7. Hyperarithmetical reduction

In the same way that the relativized version of computability induces a notion of reduction on sets, called Turing reduction, the relativization of hyperarithmetical sets induces a reduction on sets, called *hyperarithmetical reduction*. This reduction, “coarser” than Turing reduction, is often more appropriate in higher computability.

**Definition 7.1.** A set  $X$  is *hyperarithmetically reducible* to another set  $Y$  if  $X$  is  $\Delta_1^1(Y)$ . We will then write  $X \leq_h Y$ , as well as  $X <_h Y$  if  $X \leq_h Y$  and  $Y \not\leq_h X$ .  $\diamond$

Equivalently,  $X \leq_h Y$  iff there exists  $a \in \mathcal{O}^Y$  such that  $X \leq_T H_a^Y$ , or with our alternative notations, iff there exists  $\alpha < \omega_1^Y$  such that  $X \leq_T Y^{(\alpha)}$ . The following two theorems further illustrate the analogy between  $\mathcal{O}^X$  and  $X'$ .

**Theorem 7.2**

Let  $X, Y \in 2^{\mathbb{N}}$ . The following two statements are equivalent:

- (1)  $X \leq_h Y$
- (2)  $\mathcal{O}^X \leq_m \mathcal{O}^Y$

PROOF. Suppose that  $X$  is  $\Delta_1^1(Y)$ . As  $\mathcal{O}^X$  is  $\Pi_1^1(X)$  then according to Exercize 2.9,  $\mathcal{O}^X$  is  $\Pi_1^1(Y)$ . Since  $\mathcal{O}^Y$  is many-one complete for  $\Pi_1^1(Y)$  sets we have  $\mathcal{O}^X \leq_m \mathcal{O}^Y$ .

Conversely, suppose  $\mathcal{O}^X \leq_m \mathcal{O}^Y$ . We then also have  $X \leq_m \mathcal{O}^X \leq_m \mathcal{O}^Y$  and  $\mathbb{N} \setminus X \leq_m \mathcal{O}^X \leq_m \mathcal{O}^Y$ . Since  $\mathcal{O}^Y$  is  $\Pi_1^1(Y)$  then  $X$  and  $\mathbb{N} \setminus X$  are both  $\Pi_1^1(Y)$ . So  $X$  is  $\Delta_1^1(Y)$ .  $\blacksquare$

**Theorem 7.3**

Let  $X \in 2^{\mathbb{N}}$ . We have  $X <_h \mathcal{O}^X$ . More precisely  $X <_m \mathcal{O}^X$  and  $\mathcal{O}^X \not\leq_h X$ .

PROOF. It is clear that  $X <_m \mathcal{O}^X$ . We cannot have  $\mathcal{O}^X \leq_h X$  because otherwise  $\mathcal{O}^X$  would be  $\Delta_1^1(X)$  which is a contradiction.  $\blacksquare$

The many analogies between  $X'$  and  $\mathcal{O}^X$  justify the following vocabulary.

**Definition 7.4.** Given a set  $X$ , the set  $\mathcal{O}^X$  is called the *hyperjump* of  $X$ .  $\diamond$

Let us finally see the relations between the computation Kleene's  $\mathcal{O}$  and the computation of  $\omega_1^{ck}$ .

**Theorem 7.5**

Let  $X, Y \in 2^{\mathbb{N}}$ . We have  $X \leq_h Y$  implies  $\omega_1^X \leq \omega_1^Y$ .

PROOF. Let  $X$  be a  $\Delta_1^1(Y)$  set. Let  $\alpha$  be an  $X$ -computable ordinal. Then,  $\alpha$  has a representation which is  $\Delta_1^1(Y)$ . According to the relativized version of Theorem 6.1 we therefore have  $\alpha < \omega_1^Y$ . So  $\omega_1^X \leq \omega_1^Y$ .  $\blacksquare$

We will see with Theorem 30-2.2 that the converse is not true. In particular, there exists a non-hyperarithmetical set  $X$  such that  $\omega_1^X = \omega_1^{ck}$ .

**Theorem 7.6**

Let  $X \in 2^{\mathbb{N}}$ . The following two statements are equivalent:

- (1)  $X \geq_h \mathcal{O}$
- (2)  $\omega_1^X > \omega_1^{ck}$

PROOF. Suppose  $X \geq_h \mathcal{O}$ . Then,  $\omega_1^X \geq \omega_1^{\mathcal{O}}$  according to Theorem 7.5. We also have  $\omega_1^{\mathcal{O}} > \omega_1^{ck}$ : we can define the functional  $\Phi_e$  which using the oracle  $\mathcal{O}$  returns on its input  $n$  the sum of the  $n$  first elements of  $\mathcal{O}$  (via the function  $+_o$  of Example 27-5.4). The element  $3.5^e \in \mathcal{O}^{\mathcal{O}}$  will code for an ordinal greater than all the computable ordinals. We therefore have  $\omega_1^X \geq \omega_1^{\mathcal{O}} > \omega_1^{ck}$  and therefore  $\omega_1^X > \omega_1^{ck}$ . Suppose now  $\omega_1^X > \omega_1^{ck}$ . Let  $T$  be an  $X$ -computable tree such that  $|T| = \omega_1^{ck}$ . We then proceed as in the proof of Theorem 6.1 to give a  $\Sigma_1^1(X)$  description of  $\mathcal{O}$ : given  $f$  the computable function such that  $f(e)$  is a code of c.e. tree  $T_e$  which is well-founded iff  $e \in \mathcal{O}$ , we have  $e \in \mathcal{O}$  iff there exists a morphism  $f$  from  $T_e$  to  $T$ . This is indeed a  $\Sigma_1^1(X)$  description of  $\mathcal{O}$ . So  $\mathcal{O}$  is  $\Delta_1^1(X)$  and therefore  $X \geq_h \mathcal{O}$ . ■

**Exercise 7.7. (★★)** Show that for any set  $X$  which is  $\Pi_1^1$  and not  $\Delta_1^1$ , we have  $X \geq_h \mathcal{O}$ . ◇

# Chapter 30

## $\Sigma_1^1$ and $\Pi_1^1$ classes

We will now focus our study on the  $\Sigma_1^1$  and  $\Pi_1^1$  classes. Following the analogy between Higher Computability Theory and Classical Computability Theory that we have approached in Section 29-4, the  $\Sigma_1^1$  classes informally play the role of the  $\Pi_1^0$  classes (see Chapter 8). In particular, we will study a number of  $\Sigma_1^1$  basis theorems, including analogues of the low and cone avoidance basis theorems for  $\Pi_1^0$  classes. We will see that the  $\Pi_1^1$  classes behave quite differently: if the  $\Sigma_1^1$  singletons are exactly hyperarithmetic sets, the  $\Pi_1^1$  singletons can be of much higher complexity, and contain the hierarchy of hyperjumps (see Definition 6.2).

### 1. Canonical representation of $\Sigma_1^1$ classes

The  $\Sigma_1^1$  classes are easily represented by computable trees of the Baire space. Indeed let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  be a  $\Sigma_1^1$  class. Then, there exists a code  $e$  such that  $X \in \mathcal{A}$  iff  $\exists f \Phi_e(f, X) \uparrow$ . We then define the tree

$$T = \{ \langle \sigma, \tau \rangle : |\sigma| = |\tau| = t \text{ and } \Phi_e(\sigma, \tau)[t] \uparrow \text{ with } \sigma \in \mathbb{N}^{<\mathbb{N}} \text{ and } \tau \in 2^{<\mathbb{N}} \}.$$

The elements of  $\mathcal{A}$  are then exactly the elements  $X$  such that  $\langle f, X \rangle \in [T]$  for some  $f$ . This representation directly comes from Kleene's normal form theorem. Let us see immediately a first consequence with forthcoming Proposition 1.2 : given a collection  $\{\mathcal{A}_i\}_{i \in I}$  of non-empty  $\Sigma_1^1$  classes, we can uniformly choose an element in each  $\mathcal{A}_i$ . In particular the axiom of choice is superfluous for the  $\Sigma_1^1$  classes. We should however clarify what we mean by "collection of non-empty  $\Sigma_1^1$  classes". According to our definition

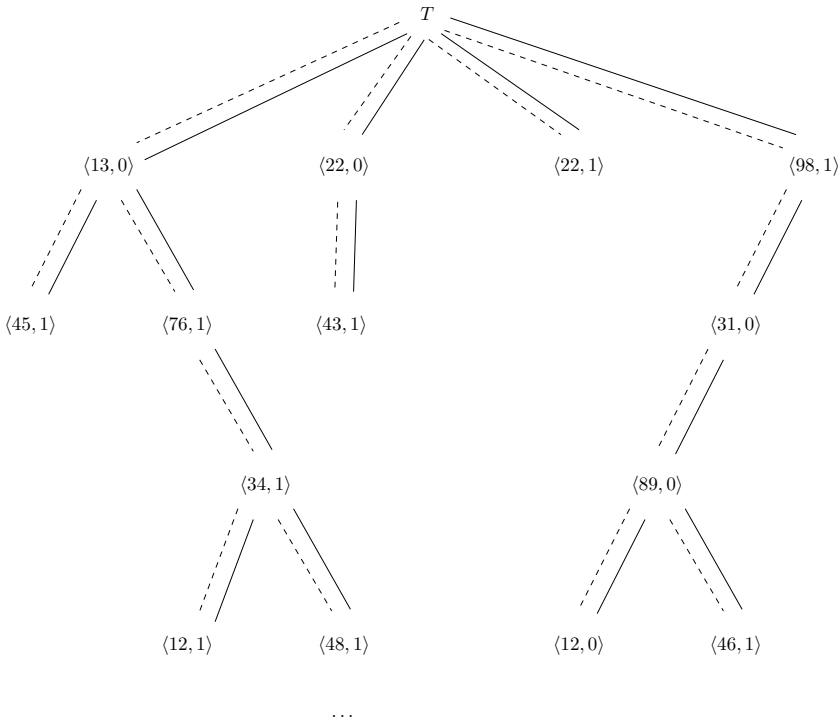


Figure 1.1: Representation of  $\Sigma_1^1$  classes by a tree : each element  $X$  of the class is considered with its potential witness functions. These witness functions are represented by the dotted lines, and the elements of the class by plain lines.

a class is  $\Sigma_1^1$  when it is a  $\Sigma_1^1(Z)$  class for some  $Z$ . We suppose here that the presentation of each class of our sequence is a concrete object, that is to say a couple  $(e, Z)$  where  $e$  is the  $\Sigma_1^1(Z)$  formula corresponding to our class.

**Proposition 1.2.** We can choose an element in each  $\Sigma_1^1(Z)$  class uniformly in a  $\Sigma_1^1(Z)$  code of the class and in  $Z$ . In particular the axiom of choice is superfluous for the  $\Sigma_1^1$  classes. ★

PROOF. Let  $\mathcal{A} \subseteq 2^{\mathbb{N}}$  be a  $\Sigma_1^1(Z)$  class. Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be the  $Z$ -computable canonical tree such that  $X \in \mathcal{A}$  iff there exists  $f$  such that  $\langle f, X \rangle \in [T]$ . Note that Kleene's normal form theorem is uniform and that the corresponding tree is therefore also obtained uniformly.

It suffices to choose the smallest path  $\langle f, X \rangle$  of  $[T]$ , for the lexicographic order. The chosen element will therefore be  $X$  — as well as its “witness”  $f$

that we can then forget. ■

Let us insist once again on the uniformity in the presentation of our  $\Sigma_1^1$  classes, and without which we cannot do much, as evidenced by the Turing degrees equivalence classes, which are all  $\Sigma_3^0$ , but for which we cannot demonstrate the existence of a choice function.

## 2. Basis theorems for $\Sigma_1^1$ classes

What is the power required to compute an element of a  $\Sigma_1^1$  class? According to Proposition 4.2 to come, the class of non-hyperarithmetic sets is  $\Sigma_1^1$ . So we need something that goes beyond hyperarithmetic computational power. It is not very difficult to see that Kleene's  $\mathcal{O}$  is sufficient: given a tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ , we search using  $\mathcal{O}$  for the first node  $\sigma \in T$  of length 1 such that  $T \upharpoonright_\sigma$  is ill-founded, and we start again inductively to gradually compute the leftmost infinite path of  $T$ . The following theorem proves something stronger: an analogue for the  $\Sigma_1^1$  classes of the  $\Pi_1^0$  low basis theorem (Theorem 8-4.3).

### 2.1. Hyperlow basis theorem

Recall that Kleene's  $\mathcal{O}$  plays the role of the halting problem in the correspondence between Higher Computability Theory and Classical Computability Theory. The following definition therefore corresponds to the notion of low set in Higher Computability Theory.

**Definition 2.1.** A set  $X$  is *hyperlow* if  $\mathcal{O}^X \leqslant_T \mathcal{O}$ . ◇

Note that if  $X$  is hyperlow, we cannot have  $\mathcal{O} \leqslant_h X$  because we would then have  $\mathcal{O}^X \leqslant_h X$  which is impossible by Theorem 29-7.3. In particular  $\omega_1^X = \omega_1^{ck}$  for any hyperlow set  $X$ .

#### **Theorem 2.2 (Gandy's basis theorem [71])**

*Any non-empty  $\Sigma_1^1$  class contains an hyperlow element  $X$ ; in particular with  $\omega_1^X = \omega_1^{ck}$ .*

PROOF. Let  $\mathcal{A}$  be a  $\Sigma_1^1$  class. At step  $n$  of the construction, suppose we have a computable tree  $T_n \subseteq \mathbb{N}^{<\mathbb{N}}$  such that  $[T_n]$  is non-empty and a string  $\sigma_n$  such that all the nodes of  $T_n$  are compatible with  $\sigma_n$ . Suppose that the infinite paths of  $[T_n]$  computably encode elements of  $\mathcal{A}$ , via a decoding function  $\Phi_n$ , such that for  $f \in [T_n]$  and for all  $m$  we have  $\Phi_n(f \upharpoonright_m) = X \upharpoonright_m$  where  $X$  is the element of  $\mathcal{A}$  encoded by  $f$ . We start the construction

with  $T_0$  being the tree encoding the elements of  $\mathcal{A}$ ,  $\Phi_0$  the function of decoding these elements and  $\sigma_0 = \epsilon$ .

We consider  $\mathcal{B}_{n+1}$  the  $\Sigma_1^1$  class of  $f \in [T_n]$  such that  $n \notin \mathcal{O}^{\Phi_n(f)}$ . There are two possible cases. Case 1:  $\mathcal{B}_{n+1}$  is empty in which case we take  $T_{n+1} = T_n \upharpoonright_{(\sigma_n m)}$  for  $m$  the smallest such that  $[T_n \upharpoonright_{(\sigma_n m)}]$  is not empty and  $\sigma_{n+1} = \sigma_n m$ . We also take  $\Phi_{n+1} = \Phi_n$ . Note that in this case all the elements  $X \in \mathcal{A}$  encoded in  $T_{n+1}$  are such that  $n \in \mathcal{O}^X$ .

We then have case 2:  $\mathcal{B}_{n+1}$  is not empty. In this case let  $U$  be the computable tree whose infinite paths encode the elements of  $\mathcal{B}_{n+1}$ . Let  $\Psi$  be the decoding function of  $\mathcal{B}_{n+1}$ , that is, which returns elements of  $[T_n]$ . Let  $\tau$  be the smallest string of  $U$  in lexicographic order, such that  $\Psi(\tau) = \sigma_n$  and such that  $[U \upharpoonright_{(\tau m)}]$  is non-empty for some  $m$ . Let  $m$  be the smallest such that  $[U \upharpoonright_{(\tau m)}]$  is non-empty. We take  $T_{n+1} = U \upharpoonright_{\tau m}$  and  $\sigma_{n+1} = \tau m$ . The  $\Phi_{n+1}$  decoding function is given by  $\Phi_{n+1}(\tau) = \Phi_n(\Psi(\tau))$ . Note that in this case all the elements  $X \in \mathcal{A}$  encoded in  $T_{n+1}$  via  $\Phi_{n+1}$  are such that  $n \notin \mathcal{O}^X$ .

Note finally that  $\mathcal{O}$  can carry out the construction in a computable way, the predicate “ $[T]$  is non-empty” for a computable tree  $T$  being  $\Sigma_1^1$ . We gradually obtain an element  $X = \Phi_0(\sigma_0) \prec \Phi_1(\sigma_1) \prec \Phi_2(\sigma_2) \prec \dots$ . We must then show that this limit point  $X$  does indeed belong to  $\mathcal{A}$ . The set  $X$  is built little by little with its witness  $g$ , so that each prefix of  $f_0 = \langle g, X \rangle$  belongs to  $T_0$ . Since  $T_0$  is a tree then  $f_0$  belongs to  $[T_0]$  and therefore  $X$  to  $\mathcal{A}$ . As verified during construction, at each step  $n$ , if we are in case 1 then  $n \in \mathcal{O}^X$  and if we are in case 2 then  $n \notin \mathcal{O}^X$ . As the construction can be computed in  $\mathcal{O}$ , the latter can thus decide little by little for each integer  $n$  if it belongs to  $\mathcal{O}^X$  or not. So  $\mathcal{O}^X \leq_T \mathcal{O}$ . ■

## 2.2. Topological space generated by the $\Sigma_1^1$ classes

The proof of Gandy’s basis theorem is very similar to the proof of Theorem 8-4.3 (that any non-empty  $\Pi_1^0$  class contains a low set). In each of the proofs, we construct a descending sequence of classes  $\mathcal{A}_0 \supseteq \mathcal{A}_1 \supseteq \dots$  such that  $\bigcap_n \mathcal{A}_n$  contains our element, low in the first case and hyperlow in the second, and where  $\mathcal{A}_n$  is  $\Pi_1^0$  in the first case and  $\Sigma_1^1$  in the second. There is nevertheless a complication in the case of  $\Sigma_1^1$  classes: while a decreasing intersection  $\bigcap_n \mathcal{A}_n$  of  $\Pi_1^0$  classes where each  $\mathcal{A}_n$  is non-empty is necessarily itself non-empty, this is not at all the case for the  $\Pi_2^0$  classes and therefore neither it is for the  $\Sigma_1^1$  classes. We therefore had to be smarter in the above proof by working with the tree which encodes the elements of our  $\Sigma_1^1$  classes, by not building only one element common to each of these classes, but also by constructing for each of these classes a witness function

associated with  $X$ . The following theorem is an abstraction built on this idea to say in scholarly terms the following thing: in the topological space generated by the  $\Sigma_1^1$  classes, an intersection of dense open classes is dense. This is an important result which has many applications, in Descriptive Set Theory as well as in higher computability. We will see an example with Theorem 2.4.

**Theorem 2.3 (Gandy)**

*Consider the class  $\bigcap_n \bigcup_m \mathcal{A}_{n,m}$  where each  $\mathcal{A}_{n,m}$  is a  $\Sigma_1^1$  class such that for all  $n$  the class  $\bigcup_m \mathcal{A}_{n,m}$  has a non-empty intersection with any non-empty  $\Sigma_1^1$  class. Then,  $\bigcap_n \bigcup_m \mathcal{A}_{n,m}$  has a non-empty intersection with any non-empty  $\Sigma_1^1$  class.*

PROOF. We consider  $\mathcal{A}$  an arbitrary non-empty  $\Sigma_1^1$  class, and we construct an element in  $\mathcal{A} \cap \bigcap_n \bigcup_m \mathcal{A}_{n,m}$ . We proceed in a similar way as in the previous theorem construction. At step 0 we define  $T_0$  as being the canonical tree which encodes the paths of  $\mathcal{A}$ , then we define  $\tau_0^0 = \epsilon, \sigma^0 = \epsilon$  and  $\eta_0 = \langle \tau_0^0, \sigma^0 \rangle$ . At step  $n$  of the construction, suppose that we have computable trees  $T_0, \dots, T_n$  where  $[T_i]$  is non-empty for  $i \leq n$ , where the nodes of  $T_i$  for  $i \leq n$  are of the form  $\langle \tau_i, \dots, \tau_0, \sigma \rangle$ , with  $\langle \tau_{i+1}, \dots, \tau_0, \sigma \rangle \in T_{i+1}$  implies  $\langle \tau_i, \dots, \tau_0, \sigma \rangle \in T_i$  for  $i < n$ , where the infinite paths  $\langle f, X \rangle$  of  $[T_0]$  encode the elements  $X \in \mathcal{A}$  and the infinite paths  $\langle f_{i+1}, \dots, f_0, X \rangle$  of  $[T_{i+1}]$  encode the infinite paths  $\langle f_i, \dots, f_0, X \rangle$  of  $[T_i]$  for  $i < n$ . We also suppose that we have a node  $\eta_n = \langle \tau_n^n, \dots, \tau_0^n, \sigma^n \rangle \in T_n$  of length  $n$  such that the nodes of  $T_n$  are all compatible with  $\eta_n$ . Note that since  $[T_n]$  is non-empty then also, by noting  $\eta_i = \langle \tau_i^n, \dots, \tau_0^n, \sigma^n \rangle$  for  $i \leq n$ , the class  $[T_i \upharpoonright_{\eta_i}]$  is non-empty.

The step  $n+1$  is then simple: by considering that  $\bigcup_m \mathcal{A}_{n,m}$  intersects any non-empty  $\Sigma_1^1$  class, we consider the  $\Sigma_1^1$  class of elements  $X$  such that there are functions  $f_n, \dots, f_0$  for which  $\langle f_n, \dots, f_0, X \rangle \in T_n$ . We choose  $m$  such that the intersection between  $\mathcal{A}_{n,m}$  and this class is not empty. Let  $U$  be the tree of nodes  $\langle \tau, \sigma \rangle$  which encode the elements of  $\mathcal{A}_{n,m}$ . Let  $T$  be the tree of all nodes  $\langle \tau_{n+1}, \tau_n, \dots, \tau_0, \sigma \rangle$  for  $\langle \tau_n, \dots, \tau_0, \sigma \rangle \in T_n$  and  $\langle \tau_{n+1}, \sigma \rangle \in U$ . Note that by hypothesis on  $\mathcal{A}_{n,m}$  the class  $[T]$  is not empty. We then define  $\tau_{n+1}$  as being the first string such that for  $\eta = \langle \tau_{n+1}, \tau_n^n, \dots, \tau_0^n, \sigma^n \rangle$  the class  $[T \upharpoonright_{\eta}]$  is not empty, then we define  $\eta_{n+1}$  as being the first child of  $\eta$  in  $T$  such that  $[T \upharpoonright_{\eta_{n+1}}]$  is not empty. Finally, we define  $T_{n+1} = T \upharpoonright_{\eta_{n+1}}$ .

From the sequence  $(\eta_n)_{n \in \mathbb{N}}$  we can extract for any  $i$  the sequence of elements of  $T_i$  given by  $\langle \tau_i^0, \dots, \tau_0^0, \sigma^0 \rangle \prec \langle \tau_i^1, \dots, \tau_0^1, \sigma^1 \rangle \prec \langle \tau_i^2, \dots, \tau_0^2, \sigma^2 \rangle \prec \dots$  which converges to  $\langle f_i, \dots, f_0, X \rangle$ . Since each  $T_i$  is a tree, we then have  $\langle f_i, \dots, f_0, X \rangle \in [T_i]$  for all  $i$ . In particular  $X \in \mathcal{A}$  and  $X \in \bigcup_m \mathcal{A}_{n,m}$  for all  $n$ . So  $\mathcal{A} \cap \bigcap_n \bigcup_m \mathcal{A}_{n,m}$  is non-empty. Since  $\mathcal{A}$  is arbitrary, the class  $\bigcap_n \bigcup_m \mathcal{A}_{n,m}$  intersects any non-empty  $\Sigma_1^1$  class. ■

### 2.3. Hyperarithmetical cone avoidance basis theorem

Note that in the previous theorem neither the intersection nor the unions need to be effective. Let us see immediately an application, with another basis theorem for  $\Sigma_1^1$  classes, this one analogous to the basis theorem 8-4.7 of cone avoidance for  $\Pi_1^0$  classes.

**Theorem 2.4 (Kreisel's basis theorem)**

Let  $\mathcal{A}$  be a non-empty  $\Sigma_1^1$  class and let  $X$  be a non-hyperarithmetical set. Then, there exists  $Y \in \mathcal{A}$  such that  $Y \not\leq_h X$ .

PROOF. The proof of this theorem uses two results that we will prove a bit later: Proposition 4.3 which implies that the class  $\{Y : \omega_1^Y = \omega_1^{ck}\}$  is  $\Sigma_1^1$ , and Proposition 3.1 which states that if a  $\Sigma_1^1$  class contains only one element, then that element is  $\Delta_1^1$ .

According to Theorem 2.2 there exists an element  $Y \in \mathcal{A}$  such that  $\omega_1^Y = \omega_1^{ck}$ . According to Proposition 4.3 (from the next section) the class  $\{Y : \omega_1^Y = \omega_1^{ck}\}$  is  $\Sigma_1^1$ . By intersecting  $\mathcal{A}$  with this class, we can therefore assume without loss of generality  $\omega_1^Y = \omega_1^{ck}$  for all the elements  $Y \in \mathcal{A}$ . In particular if  $Y \geq_h X$  for  $Y \in \mathcal{A}$  there exists  $a \in \mathcal{O}$  such that  $H_a^Y \geq_T X$ .

Let's show a first thing: it is not possible to have  $\Phi_e(H_a^Y) = X$  for  $a \in \mathcal{O}, e \in \mathbb{N}$  and all the  $Y$  elements of a non-empty  $\Sigma_1^1$  class. Indeed if this was the case then  $X$  would be the following  $\Sigma_1^1$  singleton:  $\{X : \exists Y \in \mathcal{A} \Phi_e(H_a^Y) = X\}$  (we use here the class  $\mathcal{P}$  of Theorem 28-3.4 to retrieve  $H_a^Y$  from  $Y$  in a  $\Sigma_1^1$  way). According to Proposition 3.1 — the next proposition —  $X$  would therefore be  $\Delta_1^1$ , in contradiction with our hypotheses.

Let  $e \in \mathbb{N}, a \in \mathcal{O}$  and  $\mathcal{A}_d$  be the  $\Sigma_1^1$  class of code  $d$ . For all  $m$  let  $\mathcal{A}_{\langle e, a \rangle, \langle d, m \rangle}$  be the  $\Sigma_1^1$  class of elements  $Y \in \mathcal{A}_d$  such that  $\Phi_e(H_a^Y) \neq X(m)$ . From what we have just shown, if  $\mathcal{A}_d$  is non-empty then there exists  $m$  such that  $\mathcal{A}_{\langle e, a \rangle, \langle d, m \rangle}$  is non-empty. In particular the union  $\bigcup_{d, m} \mathcal{A}_{\langle e, a \rangle, \langle d, m \rangle}$  has a non-empty intersection with any non-empty  $\Sigma_1^1$  class. Moreover  $\Phi_e(H_a^Y) \neq X$  for any element  $Y \in \bigcup_{d, m} \mathcal{A}_{\langle e, a \rangle, \langle d, m \rangle}$ . According to Theorem 2.3 there is an element  $Y$  in  $\mathcal{A} \cap \bigcap_{e, a} \bigcup_m \mathcal{A}_{\langle e, a \rangle, \langle d, m \rangle}$ . We therefore have  $H_a^Y \not\geq_T X$  for all  $a \in \mathcal{O}$ , and therefore  $Y \not\leq_h X$ . ■

**Preservation of  $\omega_1^{ck}$  and  $\leq_h$**

Note that in the previous theorem we started by restricting our class to elements  $Y$  such that  $\omega_1^Y = \omega_1^{ck}$ . The hyperarithmetical reduction hides a new phenomenon compared to the Turing reduction:  $\omega_1^{ck}$ , unlike  $\omega$ , is a relative notion. In particular, there are oracles  $Y$  such that  $\omega_1^Y > \omega_1^{ck}$ ,

and therefore which “increase” the value of the smallest non-computable ordinal.

In Theorem 2.4, to obtain  $Y \not\geq_h X$ , it is therefore not enough to ensure  $H_a^Y \not\geq_T X$  for all  $a \in \mathcal{O}$ . Indeed, there may be an  $a \in \mathcal{O}^Y \setminus \mathcal{O}$  such that  $H_a^Y \geq_T X$ . For example, if  $\omega_1^Y > \omega_1^{ck}$ , there exists  $a \in \mathcal{O}^Y$  such that  $|a| = \omega_1^{ck}$ , and thus such that  $H_a^Y$  computes  $\mathcal{O}$  (which can be seen as  $\emptyset^{(\omega_1^{ck})}$ ). Forcing  $\omega_1^Y = \omega_1^{ck}$  therefore makes it possible to restrict oneself to iterations along  $\omega_1^{ck}$  and not  $\omega_1^Y$ , which simplifies the proof knowing that  $Y$  is under construction.

### 3. The continuum hypothesis for $\Sigma_1^1$ classes

We addressed Cantor’s question in Section 9-4.6.2, whether there exists a set  $\mathcal{A} \subseteq \mathbb{R}$  such that  $|\mathbb{N}| < |\mathcal{A}| < |2^{\mathbb{N}}|$ . Although the question in all its generality is independent of ZFC, we have proved through Proposition 8-2.14 that the continuum hypothesis restricted to the closed classes of Cantor space was true, in the sense that if a closed class  $\mathcal{F}$  is uncountable, then there exists an injection of  $2^{\mathbb{N}}$  into  $\mathcal{F}$ , which amounts to saying that  $\mathcal{F}$  is either of finite cardinality, or countable, or of cardinality  $|2^{\mathbb{N}}| = |\mathbb{R}|$ . We will now expand Proposition 8-2.14 to  $\Sigma_1^1$  classes.

#### 3.1. Uncountable $\Sigma_1^1$ classes

We have seen that the axiom of countable choice was superfluous for  $\Sigma_1^1$  classes. Let us now see that this is also the case for the continuum hypothesis. We start with a simple proposition: the  $\Sigma_1^1$  singletons are  $\Delta_1^1$ .

**Proposition 3.1.** Let  $A$  be a  $\Sigma_1^1$  class containing only one  $X$  element. Then,  $X$  is  $\Delta_1^1$ . ★

PROOF. The set  $X$  has the following  $\Sigma_1^1$  description:  $n \in X$  iff  $\exists Y \in A$  such that  $n \in Y$ . The set  $\mathbb{N} \setminus X$  has the same description:  $n \notin X$  iff  $\exists Y \in A$  such that  $n \notin Y$ . The set  $X$  is therefore  $\Delta_1^1$ . ■

#### Theorem 3.2

Let  $\mathcal{A}$  be an uncountable  $\Sigma_1^1$  class. Then, there exists an injection  $f : 2^{\mathbb{N}} \rightarrow \mathcal{A}$  computable in  $\mathcal{O}$ .

PROOF. Given a computable tree  $T \subseteq \mathbb{N}^{<\mathbb{N}}$ , the question of whether  $[T]$  is non-empty is  $\Sigma_1^1$  and  $\mathcal{O}$  can therefore answer it. We consider the class  $\mathcal{A}' = \mathcal{A} \cap \{X : X \text{ is not hyperarithmetic}\}$ . We will see with Proposition 4.2 that

the class  $\mathcal{A}'$  remains  $\Sigma_1^1$ . As the class of hyperarithmetic sets is countable, the class  $\mathcal{A}'$  remains uncountable.

Let  $T$  be the computable tree such that  $[T]$  contains elements  $\langle f, X \rangle$  for all  $X \in \mathcal{A}'$ . Using  $\mathcal{O}$  we find  $\langle \tau_0, \sigma_0 \rangle \in \mathbb{N}^{<\mathbb{N}} \times 2^{<\mathbb{N}}$  and  $\langle \tau_1, \sigma_1 \rangle \in \mathbb{N}^{<\mathbb{N}} \times 2^{<\mathbb{N}}$  such that  $\sigma_0$  and  $\sigma_1$  are incomparable and such that  $[T \upharpoonright_{\langle \tau_0, \sigma_0 \rangle}]$  and  $[T \upharpoonright_{\langle \tau_1, \sigma_1 \rangle}]$  are not empty (that is, such that the corresponding trees are ill-founded). Then inductively suppose that  $\langle \tau_\rho, \sigma_\rho \rangle$  are defined and pairwise incomparable for any string  $\rho$  of length  $n$ , and that  $[T \upharpoonright_{\langle \tau_\rho, \sigma_\rho \rangle}]$  is not empty for each of them.

Suppose by contradiction that one of  $[T \upharpoonright_{\langle \tau_\rho, \sigma_\rho \rangle}]$  contains only one element. Then, according to Proposition 3.1 this element is hyperarithmetic, which contradicts the fact that  $\mathcal{A}'$  does not contain any hyperarithmetic element. We can therefore continue: using  $\mathcal{O}$  we look for  $\langle \tau_{\rho 0}, \sigma_{\rho 0} \rangle \in \mathbb{N}^{<\mathbb{N}} \times 2^{<\mathbb{N}}$  and  $\langle \tau_{\rho 1}, \sigma_{\rho 1} \rangle \in \mathbb{N}^{<\mathbb{N}} \times 2^{<\mathbb{N}}$  such that  $\sigma_{\rho i} \succeq \sigma_\rho$ ,  $\tau_{\rho i} \succeq \tau_\rho$  for  $i \in \{0, 1\}$ , such that  $\sigma_{\rho 0}, \sigma_{\rho 1}$  are incomparable and such that  $[T \upharpoonright_{\langle \tau_{\rho 0}, \sigma_{\rho 0} \rangle}]$  and  $[T \upharpoonright_{\langle \tau_{\rho 1}, \sigma_{\rho 1} \rangle}]$  are not empty. The injection is given by  $f(X)$  as being the only element of  $\bigcap_{\rho \prec X} [\sigma_\rho]$ . We have  $X \neq Y$  implies  $f(X) \neq f(Y)$ . ■

### Corollary 3.3

*The continuum hypothesis is superfluous for the  $\Sigma_1^1$  classes.*

PROOF. Any class  $\Sigma_1^1$  is  $\Sigma_1^1(X)$  for some  $X$ . We relativize the previous theorem to  $X$ . ■

Note that this is the continuum hypothesis as initially defined by Cantor (as opposed to the version of the continuum hypothesis involving cardinals, as formulated in Section 27-4.3): any class  $\Sigma_1^1$  is either finite, or countable, or contains an injection from  $2^{\mathbb{N}}$  into itself.

### 3.2. Countable $\Sigma_1^1$ classes

The technique used in the proof of Theorem 3.2 in fact allows us to considerably strengthen Proposition 3.1:

#### Theorem 3.4

*A countable  $\Sigma_1^1$  class contains only hyperarithmetic elements.*

PROOF. Let  $\mathcal{A}$  be a countable  $\Sigma_1^1$  class. Let

$$\mathcal{A}' = \mathcal{A} \cap \{X : X \text{ is not hyperarithmetic}\}.$$

Suppose by contradiction that  $\mathcal{A}'$  is non-empty. Then, we start again the construction of the previous theorem (Theorem 3.2). If we can carry out

this construction to the end we have an injection from  $2^{\mathbb{N}}$  to  $\mathcal{A}'$  which contradicts the fact that  $\mathcal{A}'$  is countable. Otherwise the construction crashes after a while: there is a string  $\sigma$  such that  $\mathcal{A}' \cap [\sigma]$  contains only one element. According to Proposition 3.1, this element is hyperarithmetic which contradicts that  $\mathcal{A}'$  does not contain any hyperarithmetic element. So  $\mathcal{A}'$  is empty and therefore  $\mathcal{A}$  only contains hyperarithmetic sets. ■

Note that according to Theorem 29-6.4, one can thus obtain uniformly from a code of a countable  $\Sigma_1^1$  class, the code of a computable ordinal  $\alpha$  such that  $\emptyset^{(\alpha)}$  can compute all its elements. We end this section by showing that the countable  $\Pi_1^0$  classes of Cantor space are already sufficient to contain hyperarithmetic elements of arbitrary complexity.

**Proposition 3.5.** Let  $T \subseteq \mathbb{N}^{<\mathbb{N}}$  be a computable tree. Then, there exists a  $\Pi_1^0$  class  $\mathcal{F} \subseteq 2^{\mathbb{N}}$  whose elements are up to a computable encoding, the elements of  $[T]$ , to which we must add finite sets. ★

PROOF. It suffices to encode each element of  $f \in [T]$  as being the set  $X_f = 0^{f(0)}10^{f(1)}10^{f(2)}1\dots$ . In the same way, we can associate with each  $\tau \in T$  the finite string  $X_\tau = 0^{\tau(0)}10^{\tau(1)}1\dots10^{\tau(|\tau|-1)}1$ . Let  $S \subseteq 2^{<\mathbb{N}}$  be the computable tree defined by  $\sigma \in S$  iff  $\sigma \prec X_\tau$  for  $\tau \in T$ . Then,  $[S]$  contains exactly the paths  $X_f$  such that  $f \in [T]$ , to which we must add the limit points of sequences of such paths, which are necessarily finite sets. Suppose indeed that an infinite set  $X$  belongs to  $S$ . Then each prefix of the form  $0^{n_0}10^{n_1}1\dots10^{n_k}1$  of  $X$  matches a string  $\tau \in T$  such that  $\tau(0) = n_0, \tau(1) = n_1, \dots, \tau(k) = n_k$ . These prefixes being all comparable and increasingly long,  $X$  thus corresponds to a function  $f \in [T]$ . ■

It is interesting to examine the different basis theorems for  $\Pi_1^0$  classes in the light of the previous theorem. Whether it is the low basis theorem or the computably dominated basis theorem — and still others — we see that we can be forced to choose finite sets to satisfy them, the other sets of the class being able to be of any complexity imposed by the  $\Sigma_1^1$  definitional power (for instance all computing  $\emptyset^{(\omega)}$ ).

#### Corollary 3.6

*Countable  $\Pi_1^0$  classes can contain hyperarithmetic elements of arbitrary complexity.*

## 4. Some emblematic $\Pi_1^1$ classes

We begin this section by presenting some special  $\Pi_1^1$  classes. Besides the specific interest that one may have in studying these natural classes, they are also useful as an analysis tool in the study of  $\Sigma_1^1$  classes, as we have seen with Section 2.3 on the cone avoidance basis theorem, and Section 3 on countable  $\Sigma_1^1$  classes.

**Definition 4.1.** We denote by HYP the class of hyperarithmetical sets.  $\diamond$

We now see that HYP is  $\Pi_1^1$ , in other words, the class of non-hyperarithmetical sets is  $\Sigma_1^1$ . Proposition 4.2 is useful for removing from a  $\Sigma_1^1$  class all its hyperarithmetical members while keeping it  $\Sigma_1^1$ .

**Proposition 4.2.** HYP class is  $\Pi_1^1$ .  $\star$

PROOF. Let  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}}$  be the  $\Pi_2^0$  class of Theorem 28-2.1 such that  $\{Y : \mathcal{P}(a, Y)\}$  is the singleton  $H_a$  for all  $a \in \mathcal{O}$ . The class of hyperarithmetical sets is then given by

$$\{X : \exists a \in \mathcal{O} \exists e \forall Y (\mathcal{P}(a, Y) \rightarrow \forall n \Phi_e(Y, n) \downarrow = X(n))\}.$$

The HYP class is therefore  $\Pi_1^1$ .  $\blacksquare$

We have seen with Theorem 29-6.4 that HYP is not  $\Sigma_1^1$ . HYP remains a Borel class, however: as there are only countably many hyperarithmetical elements and each of them is a  $\Pi_1^0$  singleton, the class of hyperarithmetical sets is therefore  $\Sigma_2^0$ . It is in fact  $\Sigma_2^0(\mathcal{O})$ .

We now study the class of sets allowing the computation of the smallest non-computable ordinal, denoted by  $\mathcal{S}$  for this section.

**Proposition 4.3.** The class  $\mathcal{S} = \{X : \omega_1^X > \omega_1^{ck}\}$  is  $\Pi_1^1$ .  $\star$

PROOF. Let  $R$  be the set of codes of c.e. linear orders. Let us show that we have  $\omega_1^X > \omega_1^{ck}$  iff:

$$\exists a \in \mathcal{O}^X \forall f \in \mathbb{N}^{\mathbb{N}} \forall e \in R$$

“ $f$  is not an isomorphism from the order coded by  $a$  to the one coded by  $e$ .”

If  $\omega_1^X > \omega_1^{ck}$ , there is  $a \in \mathcal{O}^X$  such that  $|a| = \omega_1^{ck}$ . Then, for any  $e$  encoding a computable linear order, either  $e$  corresponds to an ordinal  $\alpha < \omega_1^{ck}$ , or  $e$  corresponds to an ill-founded order. In both cases there is no isomorphism between the order coded by  $a$  and that coded by  $e$ .

Conversely if  $\omega_1^X = \omega_1^{ck}$  then for any element  $a \in \mathcal{O}^X$ , the order coded by  $a$  is isomorphic to a computable well-order.  $\blacksquare$

We have seen with Theorem 2.2 that the class  $\mathcal{S}$  is not  $\Sigma_1^1$  and even more: it does not contain any non-empty  $\Sigma_1^1$  subclass. However there again this class remains Borel. Here is a  $\Sigma_{\omega_1^{ck}+2}^0(\mathcal{O})$  description of it using the existence of codes  $3 \cdot 5^e \in \mathcal{O}_{=\omega_1^{ck}}^X$ :

$$\mathcal{S} = \{X : \exists e (\forall n \exists a \in \mathcal{O} \Phi_e(X, n) \in \mathcal{O}_{<|a|}^X \wedge \forall a \in \mathcal{O} \exists n \Phi_e(X, n) \notin \mathcal{O}_{<|a|}^X)\}.$$

According to Theorem 2.2 the class  $\{X : \Phi_e(X, n) \in \mathcal{O}_{<a}^X\}$  is  $\Sigma_{|a|+1}^0$  uniformly in  $a \in \mathcal{O}$ . We thus obtain the complexity  $\Sigma_{\omega_1^{ck}+2}^0(\mathcal{O})$ . It is possible to show using *Steel forcing* [215] that this class is not  $\Pi_{\omega_1^{ck}+2}^0$ .

**Exercise 4.4. (★★)** (*Sacks [188], Tanaka [220]*). According to Exercise 29-3.9 any  $\Pi_1^1$  class is measurable. Show that the class  $\mathcal{S} = \{X : \omega_1^X > \omega_1^{ck}\}$  has measure 0.  $\diamond$

**Exercise 4.5. (★★)** (*Keckris [109], Hjorth and Nies [91]*). Deduce from the previous exercise that there exists a  $\Pi_1^1$  class of zero measure which contains all the  $\Pi_1^1$  classes of zero measure.  $\diamond$

We finally see our third  $\Pi_1^1$  class example, which this time will not be  $\Sigma_1^1$ . The idea is to consider the class of  $X$  which are computable by  $\emptyset^{(\alpha)}$  for  $\alpha < \omega_1^X$ . The problem is that for  $\alpha \geq \omega_1^{ck}$  the set  $\emptyset^{(\alpha)}$  is not well defined. One could of course iterate the definition of  $\emptyset^{(\alpha)}$  for  $\alpha \geq \omega_1^{ck}$ , but without external help to compute  $\alpha$ , the computational power of the set  $\emptyset^{(\alpha)}$  is not clear. It is also excluded to use the oracle  $X$  itself as an external aid, because it is difficult to formally specify that  $X$  can only be used in the computation of  $\alpha$  and not in the computation of itself. We then introduce the following notation.

#### Notation

Let  $\alpha < \omega_1$  be an ordinal. We write  $X \leq_T \emptyset^{(\alpha)}$  to mean

$$\forall Y (\alpha < \omega_1^Y \rightarrow X \leq_T Y^{(\alpha)}).$$

We now define our last  $\Pi_1^1$  class:

**Definition 4.6.** We define the class

$$\mathcal{C} = \{X : \exists \alpha < \omega_1^X X \leq_T \emptyset^{(\alpha)}\}.$$

The class  $\mathcal{C}$  contains in particular all the hyperarithmetic sets, as well as for example the Kleene's  $\mathcal{O}$  (see Proposition 6.1 and Theorem 5.1) and many other elements: all the sets  $X$  that one can compute via an iteration  $\alpha$  of the Turing jump, for  $\alpha$  computable in  $X$ . The class  $\mathcal{C}$  seems less natural than HYP or  $\mathcal{S}$ . We will nevertheless see that it has remarkable properties, which we will study in the next section. Let's start by showing that this class is  $\Pi_1^1$ .

**Proposition 4.7.** The class  $\mathcal{C}$  is  $\Pi_1^1$ . ★

PROOF. The class  $\mathcal{C}$  can be expressed as follows:

$$\mathcal{C} = \left\{ X : \exists b \in \mathcal{O}^X \ \forall Y \ (\omega_1^Y \leq |b| \text{ or } \exists a \in \mathcal{O}_{=|b|}^Y \ X \leq_T H_a^Y) \right\}.$$

The predicate  $b \in \mathcal{O}^X$  is  $\Pi_1^1(X)$  uniformly in  $X$ . Using the ideas of Proposition 4.3, we show that for  $b \in \mathcal{O}^X$  the predicate  $\omega_1^Y \leq |b|$  is  $\Pi_1^1(X \oplus Y)$  uniformly in  $X$  and  $Y$ . Indeed, by considering the set  $R^Y$  of the codes  $Y$ -c.e. linear orders, we have  $\omega_1^Y \leq |b|$  iff

$$\forall e \in R^Y \ \forall f \in \mathbb{N}^{\mathbb{N}}$$

“ $f$  is not an isomorphism from the order coded by  $b$  to the one coded by  $e$ .”

The predicate  $\exists a \in \mathcal{O}^Y \ X \leq_T H_a^Y$  is for its part  $\Pi_1^1(X \oplus Y)$  uniformly in  $X$  and  $Y$ . The class is therefore  $\Pi_1^1$ . ■

## 5. Study of a very special $\Pi_1^1$ class

The class  $\mathcal{C}$  defined above is not a  $\Sigma_1^1$  class. We will see it in particular with Theorem 5.6. We will see that  $\mathcal{C}$  has many remarkable properties, which justifies devoting a section to it.

### Theorem 5.1 (Guaspari [79])

*The  $\mathcal{C}$  class is a basis for the  $\Pi_1^1$  classes: any non-empty  $\Pi_1^1$  class contains an element of  $\mathcal{C}$ .*

PROOF. Let  $\mathcal{A}$  be a non-empty  $\Pi_1^1$  class. According to the proof of Theorem 29-3.2 there exists a code  $e$  which is for all  $X$  the code of an  $X$ -c.e. well-founded tree iff  $X \in \mathcal{A}$ . By using the Kleene-Brouwer ordering on this tree we obtain a code  $e$  which is for all  $X$  the code of a linear order on a set of integers, and such that  $X \in \mathcal{A}$  iff  $e$  code for a well order. We denote by  $<_e^X$  the order in question,  $\text{dom}_e^X$  the domain of  $<_e^X$  and  $<_e^\sigma$  the order  $<_e$  enumerated with the “piece of oracle”  $\sigma$ . If  $<_e^X$  is well-founded we denote

by  $|e|^X$  the ordinal  $|\prec_e^X|$  and for  $a \in \text{dom}_e^X$  we denote by  $|a|^X$  the ordinal corresponding to  $\prec_e^X$  restricted to elements strictly smaller than  $a$ .

We start by building a  $T$  tree in the space  $(\mathbb{N} \times \omega_1 \times 2^{<\mathbb{N}})^{<\mathbb{N}}$  whose paths will be encodings of elements of  $\mathcal{A}$ . The smallest path of  $T$  for the lexicographic order will be an encoding of an element  $X \in \mathcal{A} \cap \mathcal{C}$ . The technique explained in this proof will be reused several times in this section. We will denote by  $\langle \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle$  for  $\tau \in \mathbb{N}^{<\mathbb{N}}$  and  $\rho \in \omega_1^{<\mathbb{N}}$  with  $|\tau| = |\rho| = n$  the elements of this space to formally signify the concatenation of the elements  $(\tau(0), \rho(0), \sigma_1), \dots, (\tau(n-1), \rho(n-1), \sigma_n)$ . Given such an element  $\langle \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle$ , we denote by  $f_{\tau, \rho} : \mathbb{N} \rightarrow \omega_1$  the partial finite domain function, such that  $f_{\tau, \rho}(\tau(i)) = \rho(i)$  for  $i < n$ .

To begin with, we put in  $T$  nodes  $\langle a, \alpha, \sigma \rangle$  of length 1 for any ordinal  $\alpha < \omega_1$  and any  $a, \sigma$  such that  $a$  is the first element enumerated in the order  $\prec_e^\sigma$ . Given a node  $\langle \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle$  of  $T$  with  $\sigma_1 \prec \dots \prec \sigma_n$ ,  $|\tau| = |\rho| = n$ , we put for all  $\alpha < \omega_1$  the node  $\langle \tau a, \rho \alpha, (\sigma_1, \dots, \sigma_n, \sigma_{n+1}) \rangle$  in  $T$  for  $\sigma_{n+1} \succ \sigma_n$  if  $a$  is the  $n+1$ -th item enumerated in  $\prec_e^{\sigma_{n+1}}$  — repetitions not included — such that  $f_{\tau a, \rho \alpha}$  is an order-preserving function.

Note  $\langle f, X \rangle$  the limit  $\langle \tau_1, \rho_1, (\sigma_1) \rangle \prec \langle \tau_2, \rho_2, (\sigma_1, \sigma_2) \rangle \prec \dots$  of  $[T]$  where  $\sigma_1 \prec \sigma_2 \prec \dots X$  and  $f_{\tau_1, \rho_1} \prec f_{\tau_2, \rho_2} \prec \dots f$ . Let us show that  $\langle f, X \rangle \in [T]$  for a certain function  $f$  iff  $X \in \mathcal{A}$ . Let  $\langle f, X \rangle \in [T]$ . Then,  $f$  is an order preserving function from  $\text{dom}_e^X$  to  $\omega_1$ . In particular for  $a, b \in \text{dom}_e^X$  we have  $a \prec_e^X b$  iff  $f(a) < f(b)$ . It follows that  $\prec_e^X$  must be well-founded. So  $X \in \mathcal{A}$ . Conversely if  $X \in \mathcal{A}$  then for the function  $f : \text{dom}_e^X \rightarrow \omega_1$  such that  $f(a) = |a|^X$  we must have  $\langle f, X \rangle \in [T]$ . Since  $\mathcal{A}$  is non-empty then  $[T]$  is also non-empty.

Let  $\langle f, X \rangle$  be the smallest infinite path of  $T$  for the lexicographic order. Note that we necessarily have  $f(a) = |a|^X$  for such an infinite path — because it is the smallest path for the lexicographic order. In particular  $|\prec_e^X| < \omega_1^X$ . Note that although the construction of  $T$  is intuitively effective, it is not possible to speak of the computability of  $T$  because we are handling arbitrary countable ordinals.

The objective is now to find  $\beta < \omega_1^X$  such that for any  $Y$  satisfying  $\beta < \omega_1^Y$ , the set  $X$  is computable in  $Y^{(\beta)}$ . Let  $\alpha = |\prec_e^X|$  and  $Y$  such that  $\alpha < \omega_1^Y$ . We then start again the construction of a tree  $T_Y$  whose goal is to copy  $T$ , but with ordinal codes of  $\mathcal{O}_{<\alpha}^Y$  instead of the ordinals themselves, and therefore discarding the part of  $T$  whose nodes mention ordinals greater than  $\alpha$ . As several elements of  $\mathcal{O}_{<\alpha}^Y$  can code for the same ordinal, we will work with the set  $\mathcal{O}_{1, <\alpha}^Y$  which keeps for each ordinal  $\beta < \alpha$  only the smallest code of  $\beta$  (smaller for the usual order on integers). Formally  $\mathcal{O}_{1, <\alpha}^Y = \{b \in \mathcal{O}_{<\alpha}^Y : \forall c < b \ c \notin \mathcal{O}_{=|b|}^Y\}$ . In particular  $\mathcal{O}_{1, <\alpha}^Y$  is a  $\Delta_1^1(Y)$  set. The tree  $T_Y$

is then a  $\mathcal{O}_{1, < \alpha}^Y$ -computable subset of  $(\mathbb{N} \times \mathcal{O}_{1, < \alpha}^Y \times 2^{< \mathbb{N}})^{< \mathbb{N}}$ . It is clear that the leftmost path  $\langle f, X \rangle$  of  $[T_Y]$  is the same — up to ordinals encoding of the function  $f$  — as the leftmost path of  $T$ . It is also clear that for  $Y_1, Y_2$  such that  $\alpha \leq \omega_1^{Y_1}$  and  $\alpha \leq \omega_1^{Y_2}$ , the trees  $T_{Y_1}$  and  $T_{Y_2}$  are the same, up to ordinal encoding.

In each tree  $T_Y$  (for  $Y$  such that  $\alpha \leq \omega_1^Y$ ) the leftmost path is therefore the same, and it is computable in  $\mathcal{O}^Y$  by reusing the technique of Theorem 2.2. This is not quite sufficient: this path must be hyperarithmetic in  $Y$ . To show this, we are going to go momentarily through the tree  $T_X$  for  $X$  such that  $\langle f, X \rangle$  is the leftmost path of  $[T]$ . Note first that the function  $f$  is hyperarithmetic in  $X$  as the unique function such that  $\langle f, X \rangle \in [T_X]$  and such that  $|a|^X = |f(a)|$  for all  $a \in \text{dom}_e^X$ . So  $\langle f, X \rangle$  is hyperarithmetic in  $X$ . Moreover each node located to the left of  $\langle f, X \rangle$  is the starting point of a well-founded hyperarithmetic tree in  $X$  and therefore coding for an ordinal less than  $\omega_1^X$ . Now it is possible to compute from  $\langle f, X \rangle$  the set of hyperarithmetic codes in  $X$  of all the well-founded trees lying to the left of  $\langle f, X \rangle$ . As  $\langle f, X \rangle$  is hyperarithmetic in  $X$ , according to Spector's  $\Sigma_1^1$  boundedness (combined with Theorem 28-1.10 and 29-6.1) the supremum  $\beta$  of the ordinals encoded by these trees is less than  $\omega_1^X$ .

Now for an arbitrary  $Y$  set such that  $\max(\alpha, \beta) < \omega_1^Y$ , the ordinal  $\beta$  is also the supremum of ordinals encoded by well-founded trees to the left of the branch  $\langle f, X \rangle \in [T_Y]$ . We can then compute in  $T_Y$  and  $\mathcal{O}_{< \beta}^{T_Y}$  the path  $\langle f, X \rangle$ : given a prefix  $\langle \sigma, \rho \rangle$  of  $\langle f, X \rangle$ , it suffices to look for the smallest extension under which the tree is ill-founded, i.e., under which the tree does not code for an ordinal of  $\mathcal{O}_{< \beta}^{T_Y}$ . According to a relativized version of Theorem 28-1.10 we have  $\mathcal{O}_{< \beta}^{T_Y} \leq_T T_Y^{(\beta+1)}$ . Also  $T_Y \leq_T \mathcal{O}_{1, < \alpha}^Y \leq_T Y^{(\alpha+1)}$ . So  $\mathcal{O}_{< \beta}^{T_Y} \leq_T Y^{(\beta+1+\alpha+1)}$ . In particular  $X \leq_T Y^{(\beta+1+\alpha+1)}$  for any  $Y$  such that  $\beta + 1 + \alpha + 1 < \omega_1^Y$ . Furthermore  $\beta + 1 + \alpha + 1 < \omega_1^X$ . So  $X \in \mathcal{C}$ . ■

### Theorem 5.2

*For any non-empty  $\Pi_1^1$   $\mathcal{A}$  class, we can uniformly find in a code of  $\mathcal{A}$  the code of a  $\Pi_1^1$  singleton included in  $\mathcal{A}$ .*

PROOF. It is essentially enough to notice that in the previous proof, the computation of the infinite branch  $\langle f, X \rangle$  is uniform in  $Y^{(\beta+1+\alpha+1)}$ .

Using the notations of the previous proof, let  $T \subseteq (\mathbb{N} \times \omega_1 \times 2^{< \mathbb{N}})^{< \mathbb{N}}$  be the tree corresponding to the class  $\mathcal{A}$ . Let  $\langle f, X \rangle$  be the leftmost infinite path of  $T$ . Let  $\alpha = |<_e^X|$  where  $e$  is the code used to build  $T$ . For  $Y$  such that  $\omega_1^Y > \alpha$  let  $T_Y$  be defined as in the previous proof.

For questions of uniformity we need here  $T_Y^\gamma$  the tree  $T_Y$  which copies  $T_Y$  keeping only the part of  $T_Y$  whose nodes mention ordinals lower than  $\gamma$ . Each infinite path to the right of  $\langle f, X \rangle$  in  $T$  is of the form  $\langle g, Z \rangle$  for  $g \geq f$ . In particular for  $\gamma < |\alpha|$  the tree  $T_Y^\gamma$  must be empty and for  $\gamma \geq |\alpha|$  the trees  $T_Y^\gamma$  all have  $\langle f, X \rangle$  as the leftmost infinite path, and are all the same on the left of  $\langle f, X \rangle$  (up to ordinals encoding).

We have shown in the previous proof that for some  $\beta < \omega_1^X$ , the trees — necessarily well-founded — to the left of the branch  $\langle f, X \rangle \in [T_X^\alpha]$  are of height at most  $\beta$ . The tree  $T_Y^\alpha$  is computable in  $\mathcal{O}_{1, < \alpha}^Y$  itself computable in  $Y^{(\alpha+1)}$ . As  $\mathcal{O}_{< \beta}^Z \leq_T Z^{(\beta+1)}$  for any oracle  $Z$ , taking  $Z = \mathcal{O}_{1, < \alpha}^Y$ , we get that the trees to the left of  $\langle f, X \rangle$  in  $T_Y^\alpha$  all have a code in  $\mathcal{O}_{< \beta}^{\mathcal{O}_{1, < \alpha}^Y}$  itself computable in  $Y^{(\alpha+1+\beta+1)}$ . As  $\mathcal{O}_{< \beta}^{\mathcal{O}_{1, < \alpha}^Y} \subseteq \mathcal{O}_{< \gamma}^{\mathcal{O}_{1, < \alpha}^Y}$  for  $\beta < \gamma$  then also any oracle of the form  $Y^{(\alpha+1+\gamma+1)}$  for  $\beta < \gamma$  can uniformly recognize that trees to the left of  $\langle f, X \rangle$  are well-founded.

We can then define the functional  $\Phi_e$  which for any  $Y$  such that  $\omega_1^Y \geq \omega_1^X$ , for any  $\gamma_1$  with  $\alpha \leq \gamma_1 < \omega_1^Y$  and for any  $\gamma_2$  with  $\beta \leq \gamma_2 < \omega_1^Y$  will be such that  $\Phi_e(T_Y^{\gamma_1}, Y^{(\gamma_1+1+\gamma_2+1)})$  computes the path  $\langle f, X \rangle$  uniformly in  $\gamma_1$  and  $\gamma_2$ , and for  $\gamma_1 < \alpha$  or  $\gamma_2 < \beta$ , the computation  $\Phi_e(T_Y^{\gamma_1}, Y^{(\gamma_1+1+\gamma_2+1)})$  does not produce an infinite object (either the tree  $T_Y^{\gamma_1}$  does not have any infinite path, or  $Y^{(\gamma_1+1+\gamma_2+1)}$  does not know enough well-founded tree codes and starts the computation on the left “too early”).

The  $\Pi_1^1$  singleton  $\mathcal{A}_1 \subseteq \mathcal{A}$  is then given by the following formula:

$$\left\{ X \in \mathcal{A} : \forall Y \left( \begin{array}{l} \omega_1^Y < \omega_1^X \text{ or } \exists \gamma_1, \gamma_2 < \omega_1^Y \text{ s.t. } \Phi_e(T_Y^{\gamma_1}, Y^{(\gamma_1+1+\gamma_2+1)}) \\ \text{computes an infinite path } \langle f, Z \rangle \text{ with } Z = X \end{array} \right) \right\}.$$

■

We have as a corollary a generalization of Theorem 29-4.1, due to Kondô [122] and Addison [4].

**Corollary 5.3 (Uniformization of  $\Pi_1^1$  classes)**

*For any  $\Pi_1^1$  class  $\mathcal{A} \subseteq 2^\mathbb{N} \times 2^\mathbb{N}$ , there exists a partial  $\Pi_1^1$  function  $f : 2^\mathbb{N} \rightarrow 2^\mathbb{N}$  such that for all  $X$ , if  $\{Y : (X, Y) \in \mathcal{A}\}$  is non-empty then  $(X, f(X)) \in \mathcal{A}$ .*

PROOF. We define  $f(X) = Y$  if  $Y$  belongs to the  $\Pi_1^1(X)$  singleton included in the class  $\{Y : (X, Y) \in \mathcal{A}\}$ . ■

**Corollary 5.4**

*The axiom of choice is superfluous for  $\Pi_1^1$  classes.*

According to Corollary 5.3, the axiom of choice is verified as long as there is uniformity in the presentation of our  $\Pi_1^1$  classes.

**Corollary 5.5**

*Let  $X$  be a  $\Pi_1^1$  singleton. Then,  $\exists \alpha < \omega_1^X$  such that  $X \leq_T \emptyset^{(\alpha)}$ .*

We will come back to the  $\Pi_1^1$  singletons in the next section, and we devote ourselves for the moment to ending our study of  $\mathcal{C}$ , by showing that it does not necessarily verify the continuum hypothesis. In particular, it does not contain any perfect closed class. It is in fact the largest  $\Pi_1^1$  class which does not contain a perfect closed class.

**Theorem 5.6 (Mansfield [146] Solovay [210])**

*Let  $\mathcal{A}$  be a  $\Pi_1^1$  class. Then  $\mathcal{A}$  contains a perfect closed class iff it contains an element  $X \notin \mathcal{C}$ .*

PROOF. Suppose that  $\mathcal{A}$  contains a perfect closed class  $T$ . According to Proposition 4.2 the class  $\mathcal{U}$  of elements of  $[T]$  which are not  $\Delta_1^1(T)$  is an uncountable  $\Sigma_1^1(T)$  class. According to the relativization of Theorem 2.2 the class  $\mathcal{U}$  contains an element  $X$  such that  $\omega_1^{X \oplus T} = \omega_1^T$ . We therefore have  $\omega_1^X \leq \omega_1^T$ . On the other hand  $X$  is not hyperarithmetic in  $T$ . In particular  $X \notin \mathcal{C}$ .

Now suppose that  $\mathcal{A}$  contains an element  $X \notin \mathcal{C}$ . Let  $T \subseteq (\mathbb{N} \times \omega_1 \times 2^{<\mathbb{N}})^{<\mathbb{N}}$  be the tree defined as in the proof of Theorem 5.1, that is to say — taking the notations of the proof — such that  $X \in \mathcal{A}$  iff  $\langle f, X \rangle \in [T]$  for a certain function  $f : \text{dom}_e^X \rightarrow \omega_1$  (where  $\text{dom}_e^X$  is defined as in the proof of Theorem 5.1). For an ordinal  $\gamma$  and a set  $Y$  with  $\omega_1^Y > \gamma$  we denote by  $T_Y^\gamma$  the tree  $T$  restricted to nodes only mentioning ordinals smaller than  $\gamma$  and using codes of  $\mathcal{O}_{1, < \gamma}^Y$  to represent ordinals. Note that up to ordinal encoding, the trees  $T_Y^\gamma$  are the same for all  $Y$  satisfying  $\gamma < \omega_1^Y$ . Let  $\alpha$  be the smallest ordinal such that for any  $Y$  satisfying  $\alpha < \omega_1^Y$ , the tree  $T_Y^\alpha$  contains an infinite path  $\langle f, X \rangle$  for some  $X \in \mathcal{A} \setminus \mathcal{C}$ . Note that in this case we necessarily have  $\alpha < \omega_1^X$ .

Let  $U \subseteq \mathbb{N}^{<\mathbb{N}}$  be a computable tree such that  $X \notin \mathcal{C}$  iff  $\exists f \langle f, X \rangle \in [U]$ . Consider now the tree  $S_Y \subseteq (\mathbb{N}^{<\mathbb{N}} \times \mathbb{N} \times \alpha \times 2^{<\mathbb{N}})^{<\mathbb{N}}$  such that  $\langle (\tau_1, \dots, \tau_n), \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle \in S$  iff  $\langle \tau_i, \sigma_i \rangle \in U$  for all  $i \leq n$  and  $\langle \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle \in T_Y^\alpha$ . Note that when coded near ordinals the trees  $S_Y$  are the same for all  $Y$  such that  $\alpha < \omega_1^Y$ . By hypothesis on  $\alpha$  the tree  $S_Y$  is not empty. We

claim that for any node

$$\eta = \langle (\tau_1, \dots, \tau_n), \tau, \rho, (\sigma_1, \dots, \sigma_n) \rangle \in S_Y$$

such that  $[S_Y \upharpoonright_\eta]$  is non-empty, there exist two nodes

$$\begin{aligned} \eta_1 &= \langle (\tau_1, \dots, \tau_n, \dots, \tau_{m_1}^1), \tau_1, \rho_1, (\sigma_1, \dots, \sigma_n, \dots, \sigma_{m_1}^1) \rangle \succ \eta \\ \text{et } \eta_2 &= \langle (\tau_1, \dots, \tau_n, \dots, \tau_{m_2}^2), \tau_2, \rho_2, (\sigma_1, \dots, \sigma_n, \dots, \sigma_{m_2}^2) \rangle \succ \eta \end{aligned}$$

such that  $\sigma_{m_1}^1$  and  $\sigma_{m_2}^2$  are incompatible and such that  $[S_Y \upharpoonright_{\eta_1}]$  and  $[S_Y \upharpoonright_{\eta_2}]$  are both non-empty. If such is the case, we construct an injection from  $2^{\mathbb{N}}$  to the elements of  $\mathcal{A} \setminus \mathcal{C}$  and the proposition is verified. If this is not the case then there exists a node  $\eta \in S_Y$  such that  $[S_Y \upharpoonright_\eta]$  contains at least one infinite path and such that all the infinite paths of  $[S_Y \upharpoonright_\eta]$  are of the form  $\langle f', f, X \rangle$  for the same set  $X$ . Note that for  $Y$  such that  $\omega_1^Y > \alpha$  the tree  $S_Y$  is uniformly  $\Delta_1^1(Y)$ . For  $n$  sufficiently large we have therefore  $n \in X$  iff  $\exists f', f, Z \langle f', f, Z \rangle \in [S_Y \upharpoonright_\eta] \wedge n \in Z$  and  $n \notin X$  iff  $\exists f', f, Z \langle f', f, Z \rangle \in [S_Y \upharpoonright_\eta] \wedge n \notin Z$ . So  $X$  and  $\mathbb{N} \setminus X$  are  $\Sigma_1^1(S_Y)$  and therefore  $X$  is  $\Delta_1^1(S_Y)$  and therefore  $\Delta_1^1(Y)$ . The procedure is also uniform in  $Y$ . So for any  $Y$  such that  $\omega_1^Y > \alpha$  there exists  $\beta$  with  $\alpha \leq \beta < \omega_1^Y$  such that  $Y^{(\beta)} \geq_T X$ . According to Lemma 29-6.3 relativized to  $X$  we therefore have an ordinal  $\beta$  with  $\alpha \leq \beta < \omega_1^X$  such that  $Y^{(\beta)} \geq_T X$  for all  $Y$  satisfying  $\omega_1^Y > \beta \geq \alpha$ . So  $X \in \mathcal{C}$ , which is a contradiction. ■

### Corollary 5.7

*The  $\mathcal{C}$  class is the largest  $\Pi_1^1$  class that does not contain a perfect closed.*

Therefore, We cannot construct a continuous injection from  $2^{\mathbb{N}}$  into  $\mathcal{C}$ . But is the class  $\mathcal{C}$  itself countable or uncountable? It is in fact not possible to answer this question. We can show that in the ZFC model of Gödel constructibles any set  $X \in 2^{\mathbb{N}}$  is Turing computable by an element of  $\mathcal{C}$ . In this model  $\mathcal{C}$  is uncountable and satisfies the continuum hypothesis (despite the fact that we cannot account for it via a continuous injection of  $2^{\mathbb{N}}$  into  $\mathcal{C}$ ). Via Cohen forcing technique we can build ZFC models for which  $\mathcal{C}$  can remain uncountable and does not satisfy the continuum hypothesis, or even become countable. More details can be found in section 4.3 of [34].

## 6. $\Pi_1^1$ singletons

By Theorem 5.2, any non-empty  $\Pi_1^1$  class contains a  $\Pi_1^1$  singleton. It is therefore natural to pay them some attention. In particular, any basis theorem for  $\Pi_1^1$  classes can be proved by restricting it to the singletons classes.

Unlike  $\Sigma_1^1$  singletons,  $\Pi_1^1$  singletons are not necessarily hyperarithmetic. For example:

**Proposition 6.1.** Kleene's  $\mathcal{O}$  is a  $\Pi_1^1$  singleton. ★

PROOF. It is clear the following class  $\mathcal{A}$  is  $\Pi_1^1$  and contains exactly Kleene's  $\mathcal{O}$ :

$$\left\{ X : \begin{array}{l} \forall e \in X \ e \in \mathcal{O} \wedge 1 \in X \\ \forall a \in \mathbb{N} \ a \in X \rightarrow 2^a \in X \\ \forall e \in \mathbb{N} \ \left( \begin{array}{l} \forall n \ (\Phi_e(n) \downarrow \in X \wedge \Phi_e(n) <_o \Phi_e(n+1)) \\ \rightarrow 3 \cdot 5^e \in X \end{array} \right) \end{array} \right\} \wedge \wedge$$

Note that the relation  $<_o$  used for the definition of  $\mathcal{A}$  is the relation “extended” to all elements of the form  $2^b$  or  $3 \cdot 5^b$ , as in the proof of Theorem 29-3.1. ■

We can in fact show that for any  $\Pi_1^1$  singleton  $A$ , the set  $\mathcal{O}^A$  is also a  $\Pi_1^1$  singleton. We will see it formally in the next proposition. Thus,  $\mathcal{O}, \mathcal{O}^{\mathcal{O}}, \mathcal{O}^{\mathcal{O}^{\mathcal{O}}}$ , etc., are all  $\Pi_1^1$  singletons. This consideration brings us to the transfinite iteration of the hyperjump. Contrary to the transfinite hierarchy of the jump, this time we can continue beyond  $\omega_1^{ck}$ , since our oracle then decodes  $\omega_1^{ck}$ .

**Definition 6.2.** We inductively define the hierarchy of hyperjumps:

- (1)  $J_1 = \emptyset$  and  $J_2 = \mathcal{O}$ . We define  $1 <_{ho} 2$ .
- (2) If  $a$  is in the domain of  $<_{ho}$  then  $J_{2^a} = \mathcal{O}^{J_a}$ . We define  $a <_{ho} 2^a$  and  $b <_{ho} 2^a$  for all  $b <_{ho} a$ .
- (3) If  $a$  is in the domain of  $<_{ho}$  and if  $e$  is the code of a functional such that  $\Phi_e(J_a, 0) \downarrow = a$  and such that  $\Phi_e(J_a, n) \downarrow <_{ho} \Phi_e(J_a, n+1) \downarrow$  for all  $n$  then  $J_{3^a \cdot 5^e} = \bigoplus_{n \in \mathbb{N}} J_{\Phi_e(J_a, n)}$ . We define  $b <_{ho} 3^a \cdot 5^e$  for any  $b$  such that  $b <_{ho} \Phi_e(n)$  for some  $n$ . ◇

Note the difference with the order  $<_o$ : this time the limit elements are of the form  $3^a \cdot 5^e$  and not of the form  $3 \cdot 5^e$ , where the element  $a$  serves as a code of the hyperjump iteration needed as an oracle to unfold the order coded by  $e$ .

**Proposition 6.3.** Let  $a$  be in the domain of  $<_{ho}$ . Then,  $J_a$  is a  $\Pi_1^1$  singleton. ★

PROOF. The proof is similar to that of Theorem 28-2.1: we use the fixed point theorem to define a  $\Pi_1^1$  class  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}}$  such that for any code  $e$  in the domain of  $<_{ho}$  the class  $\{X : (e, X) \in \mathcal{P}\}$  is equal to  $\{J_e\}$ . Let  $\Psi$  be

the functional such that  $\Psi(\mathcal{O}^X) = X$  for all  $X$ . The definition of  $\mathcal{P}$  is as follows:  $(b, X) \in \mathcal{P}$  if  $b = 1$  and  $X = \emptyset$  or if  $b = 2^a$  and (taking the elements of the proof of Proposition 6.1) if  $X$  is in the following class:

$$\left\{ Y : (a, \Psi(Y)) \in \mathcal{P} \wedge \begin{array}{l} \forall e \in Y \ e \in \mathcal{O}^{\Psi(Y)} \wedge 1 \in Y \\ \forall a \in \mathbb{N} \ a \in Y \rightarrow 2^a \in Y \\ \forall e \in \mathbb{N} \left( \begin{array}{l} \forall n \left( \begin{array}{l} \Phi_e(n) \downarrow \in Y \\ \wedge \ \Phi_e(n) <_o \Phi_e(n+1) \end{array} \right) \\ \rightarrow 3 \cdot 5^e \in Y \end{array} \right) \end{array} \right\}.$$

or if  $b = 3^a 5^e$  and if  $X$  is in the following class:

$$\left\{ \bigoplus_n Y_n : (a, Y_0) \in \mathcal{P} \wedge \forall n \ (\Phi(Y_0, n), Y_n) \in \mathcal{P} \right\}.$$

Above, the relation  $<_o$  is computed using the oracle  $\Psi(Y)$ . ■

It is interesting to wonder how far the hierarchy of hyperjumps goes. The order  $<_{ho}$  that we have defined jointly with this hierarchy corresponds to a well-order, which goes well beyond  $\omega_1^{ck}$ . What is the value of this ordinal? To answer this question we introduce a hierarchy of ordinals which goes alongside that of hyperjumps:

**Definition 6.4.** We define the following hierarchy of ordinals.

1. Let  $\omega_1^{ck}$  be the smallest non-computable ordinal.
2. Suppose  $\omega_\alpha^{ck}$  defined for a countable ordinal  $\alpha$ . Then,  $\omega_{\alpha+1}^{ck}$  is the smallest ordinal greater than  $\omega_\alpha^{ck}$  and of the form  $\omega_1^X$  for some  $X$ .
3. Suppose  $\omega_{\alpha_n}^{ck}$  defined for  $\alpha_0 < \alpha_1 < \dots$ . Then,  $\omega_{\sup_n \alpha_n}^{ck} = \sup_n \omega_{\alpha_n}^{ck}$ .

◇

Using and iterating Theorem 29-7.6 we easily show that  $\omega_2^{ck} = \omega_1^{\mathcal{O}}$ ,  $\omega_3^{ck} = \omega_1^{\mathcal{O}^{\mathcal{O}}}$ , etc. Note that  $\omega_\omega^{ck} = \sup_n \omega_n^{ck}$  itself is not of the form  $\omega_1^X$  for some  $X$ :

**Proposition 6.5.** Suppose  $\omega_1^X > \omega_n^{ck}$  for all  $n$ . Then,  $\omega_1^X > \omega_\omega^{ck}$ . ★

PROOF. We will define a total function  $f : \mathbb{N} \rightarrow \mathcal{O}^X$  which will be  $\Pi_1^1(X)$  and such that  $|f(n)| = \omega_n^{ck}$ . As the function is total on  $\mathbb{N}$  the relation  $f(n) = a$  is equivalent to  $\forall b \neq a \ f(n) \neq b$ . The relation  $f(n) = a$  is therefore also  $\Sigma_1^1(X)$ . In particular the image of  $f$  is a  $\Sigma_1^1(X)$  set included in  $\mathcal{O}^X$ . According to Spector's boundedness theorem, its supremum is therefore strictly less than  $\omega_1^X$  which gives us  $\omega_1^X > \omega_\omega^{ck}$ .

Now let's define our function:  $f(0) = a$  if

- (1)  $a \in \mathcal{O}^X$ .
- (2) For any  $e$  no function  $g$  is an isomorphism between the order encoded by  $a$  and that encoded by  $e$ .
- (3)  $\forall b <_o^X a \exists c \in \mathcal{O} \ b \in \mathcal{O}_{<c}^X$ .
- (4)  $a$  is the smallest integer of the set  $\mathcal{O}_{=|a|}^X$ .

Each condition is  $\Pi_1^1(X)$ . The first one makes sure that  $a$  is indeed an ordinal, the second and the third that  $|a| = \omega_1^{ck}$ , and the last one is there for the uniqueness of  $a$ . Note that we can then define a  $\Delta_1^1(X)$  code for  $\mathcal{O}$  uniformly in  $f(0)$ . We then denote by  $H_0$  the set  $\mathcal{O}$ .

We then inductively define  $f(n+1) = a$  if:

- (1)  $a \in \mathcal{O}^X$ .
- (2) For any  $e$  no function  $g$  is an isomorphism between the order encoded by  $a$  and that encoded by  $e$  with oracle  $H_n$ .
- (3)  $\forall b <_o^X a \exists c \in \mathcal{O}^{H_n} \ b \in \mathcal{O}_{<c}^X$ .
- (4)  $a$  is the smallest integer of the set  $\mathcal{O}_{=|a|}^X$ .

Finally, we inductively define  $H_{n+1}$  as being the set  $\mathcal{O}^{H_n}$  which is  $\Delta_1^1(X)$  uniformly in  $f(n+1)$ . Each condition is  $\Pi_1^1(X)$ . For condition (2)  $H_n$  is simply a notation for a  $\Delta_1^1(X)$  set whose code is uniformly known in  $f(n)$ . The fact that  $f(n+1)$  needs to reuse the value  $f(n)$  is not a problem in giving a  $\Pi_1^1(X)$  definition of the predicate  $f(n) = a$ . We can for example use the fixed point theorem to have access to the code of the graph of  $f$  in the definition of  $f$ . ■

We can show that for most limit ordinals  $\alpha$ , the ordinal  $\omega_\alpha^{ck}$  is not of the form  $\omega_1^X$  for some  $X$ . This is in fact the case for any limit ordinal  $\alpha < \omega_\alpha^{ck}$ . It suffices to repeat the previous proof, but with a function  $f : \alpha \rightarrow \mathcal{O}^X$  — of course we use codes from  $\mathcal{O}_{=\alpha}^X$  instead of  $\alpha$  in practice.

**Definition 6.6.** A countable ordinal  $\alpha$  is *recursively inaccessible* if  $\alpha$  is limit and if  $\omega_\alpha^{ck} = \omega_1^X$  for some  $X$ . ◇

Note that by the remark preceding this definition, any recursively inaccessible ordinal  $\alpha$  is such that  $\alpha = \omega_\alpha^{ck}$ . However, this is not a sufficient condition, and the first ordinal such that  $\alpha = \omega_\alpha^{ck}$  (defined by  $\sup_n f^{(n)}(1)$  where  $f(\alpha) = \omega_\alpha^{ck}$ ) is not recursively inaccessible.

Finally, let us note that our definition of recursively inaccessible ordinal is not the original definition (whose equivalence with ours follows from a theorem of Sacks [190]), which requires working in the universe of constructibles of Gödel and is beyond the scope of this book.

Sacks [189] showed that the first recursively inaccessible ordinal is equal to  $|<_{ho}|$ , the smallest ordinal inaccessible by our hierarchy of hyperjumps. It is also possible to show that the first recursively inaccessible ordinal can be encoded by a  $\Pi_1^1$  singleton. In particular the  $\Pi_1^1$  singletons go “beyond” the hyperjumps hierarchy.



# Chapter 31

## The systems $\text{ATR}_0$ and $\Pi_1^1\text{-CA}_0$

Reverse Mathematics has five major axiomatic systems of reference, namely  $\text{RCA}_0$ ,  $\text{WKL}_0$ ,  $\text{ACA}_0$ ,  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$ , linearly ordered by logical implication. The first systems,  $\text{RCA}_0$ ,  $\text{WKL}_0$  and  $\text{ACA}_0$ , correspond to computational notions known in Classical Computability Theory: the computable sets, the completions of Peano arithmetic, and finally the arithmetic sets. We therefore had the tools necessary to the study these systems in Chapter 22.

The systems  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$ , on the other hand, involve transfinite iterations of arithmetic operators, and second-order quantifications. Their corresponding computational powers are therefore naturally rather at the level of higher computability theory. Let us therefore take advantage of the knowledge we have now acquired to come back to the study of these two reverse mathematical systems.

### 1. Definitions

Let us recall the formal definitions of the systems  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$ , as presented in Chapter 22.

**The system  $\text{ATR}_0$ .** The system  $\text{ACA}_0$  was characterized by the comprehension axiom on the  $\Sigma_1^0$  formulas. The system  $\text{ATR}_0$  is characterized by a more powerful set existence axiom, informally allowing the transfinite iteration of mathematical definitions over ordinals. The canonical example is the existence of the  $\alpha$ -iteration of the Turing jump, for an ordinal  $\alpha$  in the model considered. The precise formulation of  $\text{ATR}_0$  raises an important

point: the notion of well-order is *model-relative*. More precisely, consider a strict total order  $<$  on a set  $A \subseteq \mathbb{N}$ . The statement “ $<$  is well-founded on  $A$ ” is formally expressed by the following second-order formula that we denote  $\text{WO}(<, A)$ :

$$\forall B \subseteq A \ (B \neq \emptyset \rightarrow \exists x \in B \ \forall y \in B \ y \not< x).$$

In other words,  $\text{WO}(<, A)$  means that any non-empty subset of  $A$  has a smallest element in the sense of the order  $<$ . In particular, a model  $\mathcal{M}$  satisfies  $\text{WO}(<, A)$  if any subset  $B$  of  $A$  *in the model* has a smallest element. So it may be that an ill-founded order  $<$  on  $A$  seems to be a well-order from  $\mathcal{M}$ 's viewpoint, but is not a well-order in absolute, because there exists outside of  $\mathcal{M}$  a subset of  $A$  with no smallest element.

Let  $\theta(x, X)$  be an arithmetic formula containing in particular the free variables  $x$  and  $X$ . This formula induces an operator  $\Theta : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}}$  on the sets defined by  $\Theta(X) = \{n \in \mathbb{N} : \theta(n, X)\}$ . Recall the transfinite recursion scheme presented in Section 22-8.3.

**Definition 1.1.** The *transfinite recursion scheme* states, for any arithmetic formula with parameter  $\theta(x, X)$ , any set  $A$  and any strict total order  $< \subseteq A \times A$  such that  $\text{WO}(<, A)$ , the existence of a set  $Y = \bigoplus_{a \in A} Y_a$  defined for all  $a \in A$  by

$$Y_a = \Theta \left( \bigoplus_{b < a} Y_b \right)$$

For example, if we consider the formula  $\theta(x, X) = \Phi_x^X(x) \downarrow$  and the well-ordered set  $(\mathbb{N}, <_{\mathbb{N}})$ , then the operator  $\Theta$  is the Turing jump defined by  $\Theta(X) = X'$ . We have  $Y_0 = \Theta(\emptyset) = \emptyset'$ ,  $Y_1 = \Theta(Y_0) = \emptyset''$ ,  $Y_2 = \Theta(Y_0 \oplus Y_1) = (\emptyset' \oplus \emptyset'')$ , and so on. The resulting set  $Y$  is of the same degree as that of the  $\omega$  Turing jump of  $\emptyset$ .

#### Notation

$\text{ATR}_0$  is the system  $\text{RCA}_0$  augmented with the transfinite recursion scheme.

It is clear from the previous remark that  $\text{ATR}_0$  implies  $\text{ACA}_0$  over  $\text{RCA}_0$ . Indeed,  $\text{RCA}_0$  proves that the natural order over integers is a well-order. Thus,  $\text{RCA}_0 + \text{ATR}_0$  proves the existence of  $X'$  for any set  $X$ , so prove  $\text{ACA}_0$  by Exercize 22-6.2.

**The system  $\Pi_1^1\text{-CA}_0$ .** The system  $\Pi_1^1\text{-CA}_0$  is itself simpler to define formally, using second-order arithmetic formulas. Recall that the comprehen-

sion scheme is given for any formula  $F(x)$  by

$$\exists X \forall y (y \in X \leftrightarrow F(y)) \quad (9)$$

### Notation

$\Pi_1^1\text{-CA}_0$  is the system  $\text{RCA}_0$  augmented with the comprehension scheme (9) for  $\Pi_1^1$  formulas with parameters.

Let's start by proving that  $\Pi_1^1\text{-CA}_0$  implies  $\text{ATR}_0$ .

**Proposition 1.2.**  $\Pi_1^1\text{-CA}_0$  implies  $\text{ATR}_0$ . ★

PROOF. Let  $\theta(x, X)$  be an arithmetic formula and  $< \subseteq A \times A$  a well-order on a set  $A$ . Let us specify here the notation  $\bigoplus_{b < a} X_b$  for  $a \in A$  which denotes the set  $\{\langle b, n \rangle : b < a, n \in X_b\}$ . Note also that  $\Pi_1^1\text{-CA}_0$  shows the comprehension scheme for the  $\Sigma_1^1$  formulas (the comprehension  $\Delta_0$  with second-order parameter allows to show that the complement of each set exists).

Consider the following arithmetic formula:

$$F(a, X) \leftrightarrow \forall b < a \ X_b = \left\{ m : \theta \left( m, \bigoplus_{c < b} X_c \right) \right\} \text{ where } X = \bigoplus_{b < a} X_b.$$

Assuming  $\Pi_1^1\text{-CA}_0$ , let us show that for all  $a \in A$  there exists a unique  $Z$  such that  $F(a, Z)$ . Suppose by contradiction that this is not the case. By hypothesis on our order  $< \subseteq A \times A$  there exists a smallest  $a \in A$  such that it is not the case. In particular for all  $b < a$  there exists a unique set  $Z^b$  such that  $F(b, Z^b)$ . Let us show that the set  $Z = \bigoplus_{b < a} Z^b$  exists. Consider the formula  $F(\langle b, n \rangle) \equiv b < a \wedge \exists Z \ F(b, Z) \wedge n \in Z$ . According to the  $\Sigma_1^1$  comprehension scheme there exists a set  $Z$  such that  $\langle b, n \rangle \in Z \leftrightarrow F(\langle b, n \rangle)$ . By induction hypothesis this set is necessarily unique and equal to  $\bigoplus_{b < a} Z^b$ . Now it suffices to apply the comprehension scheme for arithmetic formulas to obtain the existence of  $Z^a = \{m : \theta(m, \bigoplus_{b < a} Z^b)\}$ , which thus verifies  $F(a, Z^a)$  which contradicts our hypothesis on  $a$ .

So for all  $a \in A$  there exists a unique set  $Z$  such that  $F(a, Z)$ . Finally, we prove the existence of the set  $\bigoplus_{a \in A} Z^a$  by using the  $\Sigma_1^1$  comprehension scheme on the formula  $F(\langle a, n \rangle) \equiv a \in A \wedge \exists Z \ F(a, Z) \wedge n \in Z$ . ■

## 2. $\text{ATR}_0$ and $\Pi_1^1\text{-CA}_0$ in higher computability

Just as  $\text{WKL}_0$  corresponds to the existence of PA degrees relative to any set, or  $\text{ACA}_0$  to the existence of the Turing jump, the systems  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$

also have computational equivalences. Informally  $\text{ATR}_0$  states that for all  $X$  the set  $X^{(\alpha)}$  exists for all ordinal  $\alpha < \omega_1^X$ , and  $\Pi_1^1\text{-CA}_0$  that  $\mathcal{O}^X$  exists for all  $X$ .

However, care should be taken as to how formally expressing these two axioms. In the same way that the notion of well-order relates to the model, the notion of ordinals, as well-ordered sets, is subject to the same problems: it is possible that an integer  $a$  does not code for an ordinal, but that all the subsets of  $\{b : b <_o a\}$  in the considered model have a smallest element. In other words, from a model's viewpoint,  $a$  will be the code of a valid ordinal. We will see that one of the consequences of this phenomenon is that  $\text{HYP}$  is not an  $\omega$ -model of  $\text{ATR}_0$ , and that an  $\omega$ -model of  $\Pi_1^1\text{-CA}_0$  does not necessarily contain Kleene's  $\mathcal{O}$ . Indeed, the axiom "for all  $X$  the set  $\mathcal{O}^X$  exists" must be expressed in the model, and it implies the existence of Kleene's  $\mathcal{O}$  *from the model's viewpoint* and not necessarily the authentic Kleene's  $\mathcal{O}$ .

Let's see all this more precisely. Let us recall for a moment the proof that  $\mathcal{O}^X$  is  $\Pi_1^1(X)$  uniformly in  $X$ . The definition is done as the conjunction of two conditions for  $a$  to belong to  $\mathcal{O}$ . The first condition is  $\Pi_2^0$ . Letting  $A$  be the set containing  $a$  as well as all elements  $b <_o a$ , the condition is as follows:

- (1) The relation  $<_o$  is a linear and total order restricted to the elements of  $A$ , which are equal to 1 or a  $2^b$  for a certain  $b$  or to  $3 \cdot 5^e$  for a code  $e$  such that  $\Phi_e$  is total with  $\Phi_e(n) <_o \Phi_e(n+1)$  for all  $n$ .

This condition is not sufficient: it could be that a code  $a$  satisfies it but is such that the enumeration of  $A$  contains an infinite sequence of smaller and smaller elements:  $\dots <_o a_3 <_o a_2 <_o a_1 <_o a$ . It is then additionally necessary to satisfy the following  $\Pi_1^1$  condition:

- (2)  $\forall B \subseteq A$  if  $B$  is not empty then  $B$  contains a smallest element for  $<_o$ .

#### — Notation —

We will denote by  $L_{\mathcal{O}}(a, X)$  the predicate (1) relativized to an oracle  $X$ , to which we add the condition that if  $b <_o c <_o a$  then also  $2^b \leq_o c$ .

#### — Notation —

We will note  $W_{\mathcal{O}}(a, X)$  the predicate (2) relativized to an oracle  $X$ .

Let us also remember  $\mathcal{P} \subseteq \mathbb{N} \times 2^{\mathbb{N}} \times 2^{\mathbb{N}}$  the  $\Pi_2^0$  class of Theorem 28-2.1 such that for all  $X$  and all  $a \in \mathcal{O}^X$  the set  $H_a^X$  is the unique element such that  $\mathcal{P}(a, X, H_a^X)$ . The existence of each set  $H_a^X$  for  $a \in \mathcal{O}^X$

is the canonical example of the use of  $\text{ATR}_0$ . Of course if we are completely formal,  $\text{ATR}_0$  applied to the formula  $\theta(x, X) = \Phi_x^X(x) \downarrow$  shows the existence of sets  $\emptyset', \emptyset'', (\emptyset' \oplus \emptyset'')', \dots$ , instead of  $\emptyset', \emptyset'', \emptyset'''$ , but we easily show by induction in  $\text{ACA}_0$  that each  $H_a$  is many-one reducible to the set  $Y_a = \Theta(\bigoplus_{b < a} Y_b)$  where  $\Theta$  is the operator induced by the formula  $\theta(x, X) = \Phi_x^X(x) \downarrow$ . Conversely it is possible to show by induction in  $\text{ACA}_0$  that the sets definable by transfinite recursion are all computable by a certain transfinite iteration of the jump. All of this brings us to the next proposition.

**Proposition 2.1.** In  $\text{ACA}_0$ ,  $\text{ATR}_0$  is equivalent to the following statement: “For all  $X$  for all  $a \in \mathbb{N}$ , if  $L_{\mathcal{O}}(a, X)$  and  $W_{\mathcal{O}}(a, X)$ , then there exists a set  $Y$  such that  $\mathcal{P}(a, X, Y)$ ”. ★

The reader can find a detailed proof in [202]. Regarding  $\Pi_1^1\text{-CA}_0$ , the proof that any  $\Pi_1^1$  set is many-one reducible to  $\mathcal{O}$  is easily done in  $\text{ACA}_0$ , and we therefore also have a computational characterization of  $\Pi_1^1\text{-CA}_0$ .

**Proposition 2.2.** In  $\text{ACA}_0$ ,  $\Pi_1^1\text{-CA}_0$  is equivalent to the following statement: “For all  $X$ , the set  $\{a \in \mathbb{N} : L_{\mathcal{O}}(a, X) \text{ and } W_{\mathcal{O}}(a, X)\}$  exists”. ★

In particular in each of the two cases the predicate  $W_{\mathcal{O}}(a, X)$  equivalent to:

“ $\forall B \subseteq \{b : b <_o^X a\}$  if  $B$  is non-empty then  $B$  contains a smallest element for  $<_o^X$ ”

must be interpreted in the model, and therefore with the quantification  $\forall B$  made in the model. So it could be that  $a$  is not really the code of a well-founded order, but that no subset  $B \subseteq \{b : b <_o^X a\}$  with no smallest element belongs to the model, and therefore from the model’s viewpoint — which does not have enough sets to detect all the ill-founded thing —  $a$  is well-founded.

We will see in the next section a consequence of these considerations: HYP is not a model of  $\text{ATR}_0$ .

### 3. HYP is not a model of $\text{ATR}_0$

Hyperarithmetical sets are exactly those which can be obtained by transfinite recursion of an arithmetic operator along a computable well-order. In particular, any  $\omega$ -model of  $\text{ATR}_0$  contains all the hyperarithmetical sets, and one could at first glance conjecture that HYP forms a model of  $\text{ATR}_0$ . However, we will show that this is not the case, and that some  $\omega$ -model of  $\text{ATR}_0$  necessarily have non-well-orders which seem to be well-orders from

the model's viewpoint. The axioms of  $\text{ATR}_0$  will therefore allow the iteration of arithmetic operations along these ill-founded orders, and thus imply the existence of non-hyperarithmetic sets. For this, we introduce the notion of pseudo-well-order, that is to say, orders which would be considered as well-orders in the HYP model.

**Definition 3.1.** Let  $X \in 2^{\mathbb{N}}$ . We say that  $a$  is a  $X$ -pseudo well-order if  $L_{\mathcal{O}}(a, X)$  and if any  $\Delta_1^1(X)$  subset of  $\{b : b <_o a\}$  has a smallest element.  $\diamond$

### Notation

We note  $\text{PBO}^X \subseteq \mathbb{N}$  the set of  $X$ -pseudo well-orders. We note  $\text{PBO}$  the set of pseudo well-orders without relativization.

Note that we have  $\mathcal{O}^X \subseteq \text{PBO}^X$  for all  $X$ .

**Proposition 3.2.** For all  $X$  the set  $\text{PBO}^X$  is  $\Sigma_1^1(X)$ .  $\star$

PROOF. We have  $a \in \text{PBO}^X$  iff

$$\forall n \in \mathbb{N} \, n \notin \mathcal{O}^X \vee \exists Y \left( \mathcal{P}(n, X, Y) \wedge \begin{array}{l} Y \text{ does not compute a sub-} \\ \text{set of } \{b : b <_o a\} \\ \text{with no smallest element.} \end{array} \right)$$

which is indeed a  $\Sigma_1^1(X)$  predicate.  $\blacksquare$

The following theorem due to Harrison, is based on a simple definability argument to show the necessary existence of pseudo well-orders which are not well-orders, but along which there is still a hierarchy of Turing jumps.

### Theorem 3.3 (Harrison [83])

For all  $X$  there exists  $a \in \text{PBO}^X \setminus \mathcal{O}^X$  such that there is  $Y$  for which  $(a, X, Y) \in \mathcal{P}$ .

PROOF. Let  $X$  be any set. Note that for all  $a \in \mathcal{O}^X$  we have  $L_{\mathcal{O}}(a, X)$  and  $\exists Y (a, X, Y) \in \mathcal{P}$ . Suppose this is the case for no element  $a \in \text{PBO}^X \setminus \mathcal{O}^X$ . Then, we can give the following  $\Sigma_1^1(X)$  definition of  $\mathcal{O}^X$ :  $a \in \mathcal{O}^X$  iff  $a \in \text{PBO}^X$  and  $\exists Y (a, X, Y) \in \mathcal{P}$ . Since  $\mathcal{O}^X$  is not  $\Sigma_1^1(X)$  we have a contradiction.  $\blacksquare$

This brings us to the next definition.

**Definition 3.4.** Let  $X \in 2^{\mathbb{N}}$ . We say that a code  $a$  such that  $L_{\mathcal{O}}(a, X)$  is a *jump support* of  $X$  if  $\{Y : \mathcal{P}(a, X, Y)\}$  is not empty.  $\diamond$

### Notation

Let  $\text{SUPP}_X \subseteq \mathbb{N}$  be the set of jump supports of  $X$ .

Theorem 3.3 therefore shows that we have  $\mathcal{O}^X \subsetneq \text{PBO}^X \cap \text{SUPP}_X$  for all  $X$ . What does a set  $Y$  such that  $(a, X, Y) \in \mathcal{P}$  look like for  $a \in \text{PBO}^X \setminus \mathcal{O}^X$ ? We can “unfold”  $Y$  along the codes  $b <_o a$ . Let  $Y_b$  be the element corresponding to the code  $b$ . If  $b = 2^c$  then we have  $Y_b = Y'_c$ . If  $b = 3 \cdot 5^e$  then we have  $Y_b = \bigoplus_n Y_{\Phi_e(n)}$ . We see in particular that for all  $b <_o c$  we have  $Y_b \leq_T Y_c$ . In fact as  $L_{\mathcal{O}}(a, X)$ , then for all  $b <_o c <_o a$  we have  $2^b \leq_o c$  and therefore  $Y'_b \leq_T Y_c$ . As  $a \notin \mathcal{O}^X$  there must be a sequence  $\dots < a_3 <_o a_2 <_o a_1 <_o a$ . So the  $(Y_{a_n})_{n \in \mathbb{N}}$  form a descending sequence in the Turing jumps, that is to say that we have:  $Y'_{a_{n+1}} \leq_T Y_{a_n}$  for all  $n$ . Such a sequence cannot be hyperarithmetic, as the following theorem shows.

### Theorem 3.5 (Enderton, Putnam [55])

Let  $(X_n)_{n \in \mathbb{N}}$  be a sequence such that  $X'_{n+1} \leq_T X_n$ . Then, for all  $n$  the set  $X_n$  Turing computes all the hyperarithmetic sets.

PROOF. Let  $\alpha < \omega_1^{ck}$ . Suppose that for all  $n$  the set  $X_n$  computes  $\emptyset^{(\alpha)}$ . Let us show that for all  $n$  the set  $X_n$  computes  $\emptyset^{(\alpha+1)}$ . Let us fix  $X_n$ . As  $\emptyset^{(\alpha)} \leq_T X_{n+1}$  then  $\emptyset^{(\alpha+1)} \leq_T X'_{n+1}$ . As  $X'_{n+1} \leq_T X_n$  then  $\emptyset^{(\alpha+1)} \leq_T X_n$ .

Suppose now that for  $\alpha = \sup_m \beta_m$  — for a computable sequence of ordinals  $\beta_1 < \beta_2 < \dots$  — we have  $\emptyset^{(\beta_m)} \leq_T X_n$  for all  $n, m$ . Let us fix  $X_n$ . Note that  $X_{n+2}$  also computes  $\emptyset^{(\beta_m)}$  for all  $\beta_m$ . Using  $X''_{n+2}$  we can list all the elements  $Y_e$  which are computed by a total functional  $\Phi_e$  on  $X_{n+2}$ . We can then decide uniformly in a code of  $\beta_m$  using  $Y''_e$  — and thus using  $X''_{n+2}$  — if  $Y_e$  is an element of the  $\Pi_2^0$  singleton containing  $\emptyset^{(\beta_m)}$ . By using  $X''_{n+2}$  we can therefore gradually reconstruct  $\bigoplus_m \emptyset^{(\beta_m)}$ . As  $X''_{n+2} \leq_T X_n$  we deduce that  $X_n$  computes  $\emptyset^{(\alpha)}$ . ■

The preceding theorem can be easily relativized to any oracle.

**Corollary 3.6**

Let  $X \in 2^{\mathbb{N}}$  and  $a \in \text{SUPP}_X \setminus \mathcal{O}^X$ . Then, for any set  $Y \in 2^{\mathbb{N}}$  such that  $\mathcal{P}(a, X, Y)$ ,  $Y$  computes all the hyperarithmetical sets in  $X$ .

PROOF. It suffices to see that for such a set  $Y$  there exists a sequence  $(Y_n)_{n \in \mathbb{N}}$  such that  $Y'_{n+1} \leq_T Y_n$  with  $Y_0 = Y$ . ■

**Corollary 3.7 (Enderton, Putnam [55])**

There are sets  $X$  which Turing compute all hyperarithmetical sets and with  $\omega_1^X = \omega_1^{ck}$  — in particular with  $X \not\leq_h \mathcal{O}$ .

PROOF. Let  $a \in \text{SUPP}_X \setminus \mathcal{O}^X$ . As the class  $\{Y : \mathcal{P}(a, \emptyset, Y)\}$  is  $\Pi_2^0$  and therefore  $\Sigma_1^1$ , according to Theorem 30-2.2 it contains an element  $X$  such that  $\omega_1^X = \omega_1^{ck}$ . At the same time  $X$  computes all the hyperarithmetical sets according to Corollary 3.6. ■

We saw with Theorem 28-2.1 that for any code  $a \in \mathcal{O}^X$ , the class  $\{Y : \mathcal{P}(a, X, Y)\}$  is a  $\Pi_2^0$  singleton. The situation is quite different when  $a \in \text{SUPP}_X \setminus \mathcal{O}^X$ . Intuitively, since  $a$  does not code for a well-founded order, there is an infinite decreasing sequence, and there are many possibilities to iterate the Turing jump along this sequence. The following corollary supports this intuition:

**Corollary 3.8**

Let  $X \in 2^{\mathbb{N}}$ . Let  $a \in \text{SUPP}_X \setminus \mathcal{O}^X$ . Then,  $\{Y : \mathcal{P}(a, X, Y)\}$  is uncountable.

PROOF. As the class  $\{Y : \mathcal{P}(a, X, Y)\}$  only contains elements computing all hyperarithmetical elements, it therefore does not contain any hyperarithmetical element. As this class is  $\Sigma_1^1$ , it must be uncountable according to Theorem 30-3.4. ■

**Corollary 3.9**

HYP is not a model of  $\text{ATR}_0$ .

PROOF. Let  $X = \emptyset$  and suppose by contradiction that HYP is a model of  $\text{ATR}_0$ . According to Theorem 3.3, there is a code  $a \in \text{PBO}^X \cap \text{SUPP}_X \setminus \mathcal{O}^X$ . As  $a \in \text{PBO}^X$ , from the point of view of the model HYP,  $a$  codes for an ordinal. Since HYP is a model of  $\text{ATR}_0$ , there must exist  $Y \in \text{HYP}$  such that  $(a, \emptyset, Y) \in \mathcal{P}$ . As  $a \in \text{SUPP}_X$ , according to Theorem 3.5 no element  $Y$  such that  $(a, X, Y) \in \mathcal{P}$  is hyperarithmetical, which is a contradiction. So HYP is not a model of  $\text{ATR}_0$ . ■

## 4. Non-standard ordinal codes

The set  $\text{PBO} \setminus \mathcal{O}$  is in a way a set of *non-standard ordinals*: they are not well-orders, but it is impossible to notice it even with the power of hyperarithmetic computation. This is a fascinating class of objects and probably hasn't revealed all of its secrets yet. Here is a small summary of what we have managed to observe so far.

**Theorem 4.1 (Harrison [83])**

Suppose  $e \in \text{PBO}^X \setminus \mathcal{O}^X$ . Then the order type of  $e$  is  $\omega_1^X + \omega_1^X \nu + \alpha$  for  $\nu$  the order type of  $\mathbb{Q}$  and for some  $\alpha < \omega_1^X$ .

PROOF. We show the theorem for  $X = \emptyset$ , it can be easily relativized to any set  $X$ . Let  $e \in \text{PBO} \setminus \mathcal{O}$ . For  $a <_o e$  let  $A_{<a} = \{b : b <_o a\}$  and  $A_{>a} = \{b : a <_o b <_o e\}$ . For  $a <_o b <_o e$  let  $A_{]a,b[} = \{c : a <_o c <_o b\}$ . If  $A_{]a,b[}$  is well-ordered, we denote by  $|A_{]a,b[}|$  the ordinal corresponding to it.

*Fact 1.* Let  $a <_o e$  be such that  $A_{>a}$  is not well-ordered for  $<_o$ . Let us show that

$$\omega_1^{ck} = \sup\{|A_{]a,b[}| : b \in A_{>a} \text{ such that } A_{]a,b[} \text{ is well-ordered}\}.$$

Suppose the set  $A_{>a}$  is not well-ordered for  $<_o$ . Let  $\alpha = \sup\{|A_{]a,b[}| : b \in A_{>a} \text{ such that } A_{]a,b[} \text{ is well-ordered}\}$ . Note that we necessarily have  $\alpha \leq \omega_1^{ck}$ , because in the opposite case we would have an element  $b$  such that  $A_{]a,b[}$  is well-ordered and such that  $|A_{]a,b[}| = \omega_1^{ck}$  which would give a computation of  $\omega_1^{ck}$ . Let us show that we necessarily have  $\alpha = \omega_1^{ck}$ . Suppose by contradiction  $\alpha < \omega_1^{ck}$ . Then the set  $Z = \{b \in A_{>a} : A_{]a,b[} \text{ is well ordered}\}$  is equal to the set

$$\{b \in A_{>a} : \text{there exists a morphism from } |A_{]a,b[}| \text{ to } \alpha\}$$

which is a  $\Sigma_1^1$  definition. The set also has a natural  $\Pi_1^1$  definition and is therefore  $\Delta_1^1$ . We can now compute via  $Z$  an infinite descending sequence of elements of  $A_{>a}$ : it suffices to take an element  $c_0$  such that  $A_{]a,c_0[}$  is ill-founded, then to look for an element  $c_1 <_o c_0$  such that  $a <_o c_1$  and  $c_1 \notin Z$ , then search for  $c_2 <_o c_1$  such that  $a <_o c_2$  and  $c_2 \notin Z$ , etc. Since  $Z$  is hyperarithmetic, this contradicts  $e \in \text{PBO}$ . So  $\alpha = \omega_1^{ck}$ . This completes the proof of fact 1.

We can now apply this to the ordinal 1 encoding the smallest element of  $e$ . We thus have

$$\omega_1^{ck} = \sup\{|A_{]1,b[}| : b \in A_{>1} \text{ such that } A_{]1,b[} \text{ is well-ordered}\}.$$

The order of  $e$  therefore begins with a well-founded initial segment of length  $\omega_1^{ck}$ .

*Fact 2.* Let  $a <_o e$ , let us show that the set

$$Z = \{b : A_{]b,a[} \text{ is well ordered}\}$$

has a smallest element. Suppose by contradiction that this is not the case. We now define the  $\Pi_1^1$  set  $B = \{(n, b) : n, b \in Z \text{ and } b <_o n <_o a\}$ . Using Theorem 29-4.1 for  $\Pi_1^1$  uniformization, let  $g$  be a partial  $\Pi_1^1$  function such that if  $\{b : (n, b) \in B\}$  is not empty then  $(n, g(n)) \in B$ . We then define a  $\Pi_1^1$  function total  $f$  with  $f(0)$  as being an element of any  $Z$ , then  $f(n+1) = g(f(n))$ . We then have  $f(n+1) <_o f(n)$  for all  $n$ . As  $f$  is total it is therefore  $\Delta_1^1$  according to Proposition 29-4.2. His image is therefore  $\Delta_1^1$  which contradicts  $e \in \text{PBO}$ . So for all  $a <_o e$  the set  $Z = \{b : A_{]b,a[} \text{ is well-ordered}\}$  has a smallest element. This proves fact 2.

For  $a <_o e$  let  $b_a$  be such an element. If  $A_{>b_a}$  is well-ordered then it is a computable ordinal  $\alpha$ , which corresponds to the final segment of the order. Otherwise for all  $a <_o e$  we have  $\omega_1^{ck} = \sup_{b \in A} \{|A_{]b_a, b[}| : A_{]b_a, b[} \text{ is well-ordered}\}$

*Fact 3.* Let  $a_1 <_o a_2 < e$  each be the beginnings of a well-founded part (i.e. of the form  $b_a$  for some  $a$ ). Let us show that there exists a  $c <_o a_2$  such that  $c \notin \{b : A_{]a_1, b[} \text{ is well-ordered}\}$ .

Knowing that  $a_1 <_o a_2$  are the beginnings of well-ordered parts, we have  $a_2 \notin \{b : A_{]a_1, b[} \text{ is well-ordered}\}$ . Then, we necessarily have  $c <_o a_2$  such that  $c \notin \{b : A_{]a_1, b[} \text{ is well-ordered}\}$ , because otherwise  $a_2$  would compute  $\omega_1^{ck}$  since  $a_1$  starts a well-founded part of length  $\omega_1^{ck}$ .

Fact 1 shows that the order type of  $e$  begins with  $\omega_1^{ck}$ . Facts 1 and 2 together show that each element is either in a  $\omega_1^{ck}$  type suborder, or belongs to the final part  $\alpha < \omega_1^{ck}$ . Finally, Fact 3 shows that between two copies of  $\omega_1^{ck}$ , or between one copy of  $\omega_1^{ck}$  and  $\alpha$ , there is another copy of  $\omega_1^{ck}$ . The order type of  $e$  is therefore  $\omega_1^{ck} + \omega_1^{ck}\nu + \alpha$ . ■

**Corollary 4.2 (Feferman and Spector [57])**

*There is a set  $\mathcal{O}_1 \subseteq \mathcal{O}$  which is  $\Pi_1^1$ , well-ordered by  $<_o$ , and whose supremum is  $\omega_1^{ck}$ .*

PROOF. Let  $e \in \text{PBO}^X \setminus \mathcal{O}^X$ . Then, by taking the notations of the previous theorem we have  $\mathcal{O}_1 = \{b <_o e : A_{]1, b[} \text{ is well-founded}\}$ . ■

The set  $\mathcal{O}_1$  can be useful to get away even more from coding systems: instead of considering that we are working with the code of an ordinal  $\alpha$ ,

we can then act as if we were working directly with the ordinal  $\alpha$  itself, via its unique code in  $\mathcal{O}_1$ . We now see a remarkable theorem on the relation between the sets  $\text{PBO}^X$  and  $\text{SUPP}^X$ . We will see in particular that if an ill-founded order is the support of a jump hierarchy, then necessarily this order does not contain a hyperarithmetic subset with no smallest element. In other words  $\text{SUPP}^X \subseteq \text{PBO}^X$ . We use the following Steel lemma for this:

**Lemma 4.3 (Steel).** Let  $\Phi$  be a computable functional. Then, there is no sequence  $(X_n)_{n \in \mathbb{N}}$  such that  $\Phi(X_n) = X'_{n+1}$  for all  $n$ . ★

PROOF. Let  $(X_n)_{n \in \mathbb{N}}$  be a sequence such that  $\Phi(X_n) = X'_{n+1}$  for all  $n$ . We easily construct a functional  $\Psi$  such that  $\Psi(X_n, m) = X'_{n+m}$  for all  $n, m \in \mathbb{N}$  with  $m > 0$ . Using the fixed point theorem, let  $e$  be the code of a program that halts on the oracle  $X$  if  $\exists m > 0 \ e \notin \Psi(X, m)$ .

Suppose  $e \notin X'_0$ . Then, for all  $m > 0$  we have  $e \in \Psi(X_0, m)$  that is to say  $e \in X'_m$  for all  $m > 0$ . In particular we have  $e \in X'_1$  and therefore  $\exists k > 0$  such that  $e \notin \Psi(X_1, k)$ . We therefore have  $e \notin X'_{1+k}$  which contradicts  $e \in X'_m$  for all  $m > 0$ .

Now suppose  $e \in X'_0$ . Then,  $\exists m > 0$  such that  $e \notin \Psi(X_0, m)$  and therefore such that  $e \notin X'_m$ . We can therefore repeat the previous argument but with the sequence  $Y_0 = X_m, Y_1 = X_{m+1}, \dots$ . In all cases, we end up with a contradiction. ■

We can now show the expected result:

**Theorem 4.4 (Friedman [67])**  
 $\text{SUPP}^X \subseteq \text{PBO}^X$  for all  $X$ .

PROOF. We show the result for  $X = \emptyset$ , the proof can be straightforwardly relativized to any oracle  $X$ . Suppose the class  $\{Y : \mathcal{P}(e, \emptyset, Y)\}$  is non-empty for  $e \notin \mathcal{O}$  and let  $Y$  be one of its members. Suppose by contradiction that there exists a hyperarithmetic set  $Z \subseteq \{a : a <_o e\}$  having no smallest element. We will then build a sequence of sets  $(X_n)_{n \in \mathbb{N}}$  and a functional  $\Phi$  such that  $\Phi(X''_n) = X'''_{n+1}$ . It will suffice to apply Lemma 4.3 to the sequence  $(X''_n)_{n \in \mathbb{N}}$  to obtain a contradiction.

We can unfold the set  $Y$  along the codes  $a <_o e$ . Let  $Y_a$  be the set corresponding to the code  $a$ . According to Theorem 3.5 each set  $Y_a$  for  $a \in Z$  computes all the hyperarithmetic sets and therefore in particular  $Z$ . Let  $a \in \mathcal{O}$  be such that  $\Psi(H_a) = Z$  for a functional  $\Psi$ . The  $\Phi$  functional proceeds as follows: on an oracle of the form  $(1^b 0 Y_b)''$  for  $b \in Z$ , the functional lists all the total functions for  $Y_b$  and checks if the result of the computation is indeed a member of the class  $\{X : \mathcal{P}(a, \emptyset, X)\}$ , in which case it

is necessarily  $H_a$ . Once  $H_a$  has been identified, the functional uses it to compute  $Z$ , then to find  $c \in Z$  with  $c <_o b$  and such that  $2^c \leq_o b$ . Once this element is found the functional can then compute  $Y_{2^c} = Y'_c$  from  $Y_b$ , then compute  $(1^c 0 Y_c)' \leq_T Y'_c \leq_T Y_b$  then finally return  $(1^c 0 Y_c)'''$  using  $Y_b''$ . ■

Do we have equality between  $\text{SUPP}^X$  and  $\text{PBO}^X$ ? Friedman [65] showed that this was not the case, by constructing an element  $e \in \text{PBO} \setminus \text{SUPP}$ .

**Theorem 4.5 (Friedman)**

$\text{SUPP}^X \subsetneq \text{PBO}^X$  for all  $X$ .

PROOF. We show the theorem for  $X = \emptyset$ , the proof relativizing straightforwardly to any oracle  $X$ . Let  $f$  be a total computable function, defined using Theorem 27-5.10 and Proposition 27-5.18 such that  $e \in \mathcal{T}$  iff  $f(e) \in \mathcal{O}$ . Let  $g$  be the total computable function which on the code  $e$  of a c.e. tree  $T_e \subseteq \mathbb{N}^{<\mathbb{N}}$ , returns the code of the tree  $T_{g(e)}$  of Baire space which encodes the elements of the following  $\Pi_2^0$  class.

$$\{X : L_{\mathcal{O}}(f(e), \emptyset) \text{ and } (f(e), \emptyset, X) \in \mathcal{P}\}.$$

Let's explain the tree  $T_{g(e)}$  a little more concretely: given a  $\Pi_2^0$  description  $\bigcap_n \mathcal{U}_n$  of the class  $\{X : (f(e), \emptyset, X) \in \mathcal{P}\}$  and  $\bigcap_n A_n$  a  $\Pi_2^0$  description of the set  $\{a : L_{\mathcal{O}}(a, \emptyset)\}$ , we proceed as follows: suppose a node  $\rho$  of length  $n+1$  is enumerated in  $T_{g(e)}$  and corresponding to strings  $\sigma_0 \prec \dots \prec \sigma_n$  where  $\sigma_i$  is enumerated in  $\mathcal{U}_i$ . Before enumerating the children of  $\rho$  in  $T_{g(e)}$ , we find out whether  $f(e) \in A_{n+1}$ . If this is correct, we enumerate  $\rho m$  in  $T_{g(e)}$  for any  $m$  corresponding to an extension  $\sigma_{n+1} \succ \sigma_n$  with  $\sigma_{n+1}$  enumerated in  $\mathcal{U}_{n+1}$ . It is clear that  $L_{\mathcal{O}}(f(e), \emptyset)$  and  $\{X : (f(e), \emptyset, X) \in \mathcal{P}\}$  is non-empty iff the tree  $T_{g(e)}$  has an infinite path, in which case the infinite paths of  $T_{g(e)}$  are exactly the encodings of the elements of  $\{X : (f(e), \emptyset, X) \in \mathcal{P}\}$ .

Using the fixed point theorem, let us now consider  $e$  such that  $T_e = T_{g(e)}$ . Suppose by contradiction that  $T_e$  does not contain an infinite path. Then,  $f(e) \in \mathcal{O}$  and in this case the tree  $T_{g(e)}$  must contain an infinite path, which is a contradiction. So  $T_e$  is ill-founded. As  $T_{g(e)}$  is ill-founded, we deduce that  $L_{\mathcal{O}}(f(e), \emptyset)$  and  $\{X : (f(e), \emptyset, X) \in \mathcal{P}\}$  is not empty, which implies  $f(e) \in \text{SUPP} \setminus \mathcal{O}$ . In particular  $f(e) \in \text{PBO}$ . Note that any element  $X$  such that  $(f(e), \emptyset, X) \in \mathcal{P}$  computes an element of  $[T_e]$  — its own representation — and therefore computes a subset of  $Z_X \subseteq \{a : a <_o f(e)\}$  which has no smallest element. We now consider the order  $f(e) +_o f(e)$ . The proof that we have  $L_{\mathcal{O}}(f(e) +_o f(e), \emptyset)$  is left to the reader. Any element  $a <_o f(e)$  has its analogue  $f(e) +_o a$

which is such that  $f(e) <_o f(e) +_o a <_o f(e) +_o f(e)$ . Conversely, for any element  $a$  such that  $f(e) <_o a <_o f(e) +_o f(e)$  we can search for an element  $b <_o a$  such that  $a = f(e) +_o b$ . It is clear that  $f(e) +_o f(e) \in \text{PBO}$ : indeed any  $\Delta_1^1$  subset  $Z \subseteq \{a : a <_o f(e) +_o f(e)\}$  with no smallest element is cofinite in  $\{a : a <_o f(e)\}$  — in which case  $f(e) \notin \text{PBO}$  — or cofinite in  $\{a : a <_o f(e) +_o f(e)\}$  in which case  $Z$  can compute its analogue under  $f(e)$  and again  $f(e) \notin \text{PBO}$ . We claim that on the other hand  $f(e) +_o f(e) \notin \text{SUPP}$ . Suppose by contradiction that there exists a set  $Y$  such that  $(f(e) +_o f(e), \emptyset, Y) \in \mathcal{P}$ . We can then “unfold”  $Y$  along the pseudo ordinal codes and for  $a <_o f(e) +_o f(e)$  we denote by  $Y_a$  the set corresponding to  $a$  in this unfolding. Note that  $Y_{f(e)}$  — which is such that  $(f(e), \emptyset, Y_{f(e)}) \in \mathcal{P}$  — computes the set  $Z_{Y_{f(e)}} \subseteq \{a : a <_o f(e)\}$  with no smallest element, and therefore a set  $Z \subseteq \{a : f(e) <_o a <_o f(e) +_o f(e)\}$  with no smallest element. We will then define a computable function  $\Phi$  and a sequence  $(X_n)_{n \in \mathbb{N}}$  which will be such that  $\Phi(X_n) = X'_n$ , in contradiction with Lemma 4.3. The sequence  $(X_n)_{n \in \mathbb{N}}$  is simply given by  $X_n = 1^{a_n} 0 Y_{a_n}$  for a sequence  $\dots < a_3 <_o a_2 < a_1 \in Z$ . The functional  $\Phi$  acts as follows on the set  $1^{a_n} 0 Y_{a_n}$  with  $f(e) <_o a_n$  and  $a_n \in Z$ : it computes  $Y_{f(e)}$  from  $Y_{a_n}$  and the knowledge of  $a_n$ , then it computes  $Z$  from  $Y_{f(e)}$  and finally it searches for  $a_{n+1} \in Z$  with  $a_{n+1} <_o a_n$  such that  $2^{a_{n+1}} \leq_o a_n$ . Note that we have  $Y_{2^{a_{n+1}}} = Y'_{a_{n+1}}$ . From  $Y_{a_n}$  the functional can therefore compute  $Y'_{a_{n+1}}$  then compute and return the set  $(1^{(a_{n+1})} 0 Y_{a_{n+1}})'$ . We then have a contradiction with Lemma 4.3. It follows that our hypothesis is false and therefore that  $f(e) +_o f(e) \notin \text{SUPP}$ . ■

We end this section with a Harrington theorem: PBO and SUPP are both  $\Sigma_1^1$ -complete.

**Theorem 4.6 (Harrington (unpublished))**

Let  $A$  be a  $\Sigma_1^1$  set such that  $\mathcal{O} \subseteq A \subseteq \text{PBO}$ . Then,  $A$  is  $\Sigma_1^1$ -complete.

PROOF. We first show the result for  $\mathcal{T}$  the set of c.e. codes of well-founded trees in the Baire space, and PBT the set of c.e. codes of trees without infinite hyperarithmetic paths. We suppose  $\mathcal{T} \subseteq A \subseteq \text{PBT}$  for a  $\Sigma_1^1$  set  $A$ . Let us now show that the set of codes of c.e. ill-founded trees is many-one reducible to  $A$ .

Since  $A$  is  $\Sigma_1^1$  there exists a function  $f$  such that  $n \in A$  iff  $f(n)$  is the c.e. code of an ill-founded tree. Using the fixed point theorem we define a function  $g$  such that for all  $e$ ,  $g(e)$  is the c.e. tree of morphisms (see Definition 29-5.6) from the c.e. tree encoded by  $e$  to that encoded by  $f(g(e))$ .

Let us show first that for all  $e$ ,  $g(e)$  codes for an ill-founded tree. Suppose by contradiction that  $g(e)$  codes for a well-founded tree. Then,  $g(e)$  belongs to  $A$  and therefore  $f(g(e))$  codes for an ill-founded tree. There then

necessarily exists a morphism from the tree encoded by  $e$  to that encoded by  $f(g(e))$  and therefore  $g(e)$  — which codes for all of these morphisms — must be ill-founded.

Let us now show that  $g$  is the expected reduction, i.e., such that  $e$  codes for an ill-founded tree iff  $g(e) \in A$ . Suppose that  $e$  codes for an ill-founded tree. As  $g(e)$  always codes for an ill-founded tree there is a morphism from the tree encoded by  $e$  to that encoded by  $f(g(e))$ . So  $f(g(e))$  also codes for an ill-founded tree and therefore  $g(e) \in A$  by definition of  $f$ . Now suppose that  $e$  codes for a well-founded tree. Let  $\alpha < \omega_1^{ck}$  be the height of this tree. We know that there is a morphism from the tree encoded by  $e$  to the one encoded by  $f(g(e))$  because  $g(e)$  codes for an ill-founded tree. According to Proposition 28-1.7 the set  $\emptyset^{(\alpha+1)}$  can obtain uniformly in  $\beta \leq \alpha$  the set  $\mathcal{T}_{<\beta}$  of c.e. well-founded tree codes of height lower than  $\beta$ . One can then using  $\emptyset^{(\alpha+1)}$  compute such a morphism: it suffices to send nodes starting from trees of heights  $\beta$  towards nodes of the same length starting from trees which are not of height less than  $\beta$  (while remaining consistent with the part of the morphism already defined). Knowledge of all the codes of trees of heights  $\beta \leq \alpha$  is therefore sufficient. So the tree encoded by  $g(e)$  contains hyperarithmetical infinite paths and therefore does not belong to PBO and therefore not to  $A$ .

Now to get the result for a  $\Sigma_1^1$  set  $A$  with  $\mathcal{O} \subseteq A \subseteq \text{PBO}$  it suffices to consider with the help of Theorem 27-5.10 and Proposition 27-5.18 a computable function  $f$  such that  $e$  is a code c.e. for a well-founded tree — i.e.  $e \in \mathcal{T}$  — iff  $f(e) \in \mathcal{O}$ , and to consider the set  $B = \{e \in \mathbb{N} : f(e) \in A\}$ . Such a set is  $\Sigma_1^1$ . It is clear by definition of  $f$  that  $\mathcal{T} \subseteq B$  since  $\mathcal{O} \subseteq A$ . Moreover we easily show that if  $e$  codes for a tree with a hyperarithmetical path, then if we have  $L_{\mathcal{O}}(f(e), \emptyset)$ , the set  $\{a : a <_o f(e)\}$  contains a hyperarithmetical subset which has no smallest element. So we have  $e \notin \text{PBT}$  implies  $f(e) \notin \text{PBO}$ , so  $f(e) \in \text{PBO}$  implies  $e \in \text{PBT}$ . As  $A \subseteq \text{PBO}$  then  $B \subseteq \text{PBT}$ . We then reduce all  $\Sigma_1^1$  sets to  $B$  which itself is reduced to  $A$  via the function  $f$ . ■

#### Corollary 4.7

*The sets PBO and SUPP are  $\Sigma_1^1$ -complete.*

PROOF. We have seen with Proposition 3.2 that the set PBO is  $\Sigma_1^1$ . The set SUPP is also  $\Sigma_1^1$ :  $a \in \text{SUPP}$  iff  $\exists Y \mathcal{P}(a, \emptyset, Y)$ . They are therefore both  $\Sigma_1^1$ -complete. ■

## 5. Separation between $\text{ATR}_0$ and $\Pi_1^1\text{-CA}_0$

While some models of  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$  may be complex to apprehend due to orders that they cannot detect as being ill-founded, it does not necessarily have to be so. It is quite possible to define models which reflect well what is happening outside of themselves.

**Definition 5.1.** Let  $\mathcal{M} \subseteq 2^{\mathbb{N}}$ . Then,  $\mathcal{M}$  is a  $\beta$ -model if for all  $X$  in  $\mathcal{M}$  and for all non-empty  $\Sigma_1^1(X)$  class  $\mathcal{A}$ , there exists  $Y \in \mathcal{A} \cap \mathcal{M}$ .  $\diamond$

Note that we implicitly assume that the  $\beta$ -models are always  $\omega$ -models, for which only the second-order part is therefore specified.

**Proposition 5.2.** Let  $\mathcal{M}$  be a  $\beta$ -model closed under Turing join. Then,  $\mathcal{M}$  is a model of  $\text{ATR}_0$ .  $\star$

PROOF. For each  $X \in \mathcal{M}$  and each  $Y \leq_T X$  the element  $Y$  is a  $\Pi_1^0(X)$  singleton and therefore  $Y \in \mathcal{M}$ . Thus,  $\mathcal{M}$  is closed under join and by Turing reduction, so is a Turing ideal. We therefore have  $\mathcal{M} \models \text{RCA}_0$ . Moreover for all  $X \in \mathcal{M}$  the class  $\{X'\}$  is  $\Sigma_1^1(X)$ , so  $X' \in \mathcal{M}$ . So  $\mathcal{M} \models \text{ACA}_0$ .

By Proposition 2.1, it suffices to show that if  $X \in \mathcal{M}$  and  $a \in \mathbb{N}$  is such that  $L_{\mathcal{O}}(a, X)$  and  $W_{\mathcal{O}}(a, X)$ , then there exists a  $Y \in \mathcal{M}$  such that  $(a, X, Y) \in \mathcal{P}$ . Let  $X \in \mathcal{M}$  and  $a \in \mathbb{N}$  such that  $L_{\mathcal{O}}(a, X)$  and  $W_{\mathcal{O}}(a, X)$ . Let us show that  $a \in \mathcal{O}^X$ . Indeed, if  $a \notin \mathcal{O}^X$ , the class of subsets of  $\{b : b <_o a\}$  with no smallest element is a non-empty  $\Sigma_1^1(X)$  and there is therefore a witness of this fact in  $\mathcal{M}$ , so  $\mathcal{M} \not\models W_{\mathcal{O}}(a, X)$ , contradiction. If  $a \in \mathcal{O}^X$  the class  $\{Y : (a, X, Y) \in \mathcal{P}\}$  is the  $\Pi_2^0$  singleton  $H_a^X$  and therefore  $H_a^X \in \mathcal{M}$  because  $\mathcal{M}$  is a  $\beta$ -model.  $\blacksquare$

**Proposition 5.3.** There exists a minimal  $\beta$ -model of  $\Pi_1^1\text{-CA}_0$ : the class of elements Turing reducible to a finite iteration of the hyperjump.  $\star$

PROOF. By Proposition 2.2, any  $\beta$ -model of  $\Pi_1^1\text{-CA}_0$  contains the true Kleene's  $\mathcal{O}$ , and therefore  $\mathcal{O}^{\mathcal{O}}$ ,  $\mathcal{O}^{\mathcal{O}^{\mathcal{O}}}$  and so on. As it is closed under Turing reduction it contains all Turing sets reducible to a finite iteration of the hyperjump.

Reciprocally let  $\mathcal{M}$  be the model of elements Turing reducible to a finite iteration of the hyperjump. Let us show that  $\mathcal{M}$  is a Turing ideal. For any  $X \in \mathcal{M}$ , there is an integer  $n$  such that  $X \leq_T J_n$ , where  $J_n$  is the  $n$ -th iteration of the hyperjump. If  $Y \leq_T X$ , then  $Y \leq_T J_n$ , so  $Y \in \mathcal{M}$ . Moreover, for all  $X, Y \in \mathcal{M}$  we have  $X \leq_T J_n$  and  $Y \leq_T J_m$  for  $n, m \in \mathbb{N}$  and therefore  $X \oplus Y \leq_T J_{\max(n, m)}$ . So  $X \oplus Y \in \mathcal{M}$ . So  $\mathcal{M} \models \text{RCA}_0$ . Let us show that for all  $X \in \mathcal{M}$ ,  $\mathcal{O}^X \in \mathcal{M}$ . By definition of  $\mathcal{M}$ , there is an  $n \in \mathbb{N}$

such that  $X \leq_T J_n$ . In particular  $\mathcal{O}^X \leq_T J_{n+1}$  and therefore  $\mathcal{O}^X \in \mathcal{M}$ . So  $\mathcal{M} \models \Pi_1^1\text{-CA}_0$ . Note that  $\mathcal{M}$  is a  $\beta$ -model because for a non-empty  $\Sigma_1^1(X)$  class, according to Theorem 30-2.2 an element of this class is computable using  $\mathcal{O}^X$  and therefore belongs to  $\mathcal{M}$ . ■

We now show our separation theorem between  $\text{ATR}_0$  and  $\Pi_1^1\text{-CA}_0$ .

**Theorem 5.4**

*There are  $\beta$ -models of  $\text{ATR}_0$  that do not contain  $\mathcal{O}$ .*

PROOF. Let  $e_1, e_2, \dots$  be a sequence where  $e_i$  is a code of a  $\Sigma_1^1$  class and where each possible code is repeated infinitely many times.

At step  $n$  suppose that we have  $n$  sets  $X_1, \dots, X_n$  with  $\mathcal{O}^{X_1 \oplus \dots \oplus X_n} \leq_T \mathcal{O}$  and such that  $X_{i+1}$  is an element of the class  $\Sigma_1^1(X_1 \oplus \dots \oplus X_i)$  coded by  $e_{i+1}$  for all  $i < n$ , if this class is not empty. We consider the class  $\Sigma_1^1(X_1 \oplus \dots \oplus X_n)$  of code  $e_{n+1}$ . If this is not empty, we define, using Theorem 30-2.2,  $X_{n+1}$  as being an element of this class such that  $\mathcal{O}^{X_1 \oplus \dots \oplus X_n \oplus X_{n+1}} \leq_T \mathcal{O}^{X_1 \oplus \dots \oplus X_n} \leq_T \mathcal{O}$ . If this class is empty we define  $X_{n+1} = \emptyset$ . Our model  $\mathcal{M}$  is given by  $\{X_n : n \in \mathbb{N}\}$ .

It is clear that  $\mathcal{M}$  only contains elements  $X$  such that  $\mathcal{O}^X \leq_T \mathcal{O}$  and therefore does not contain  $\mathcal{O}$ . Moreover  $\mathcal{M}$  is a  $\beta$ -model because for  $X_n \in \mathcal{M}$  and a  $\Sigma_1^1(X_n)$  class, there is a code  $e_m$  for  $m > n$  such that  $e_m$  corresponds with an oracle  $X_1 \oplus \dots \oplus X_n \oplus \dots \oplus X_{m-1}$  to the  $\Sigma_1^1(X_n)$  class (and this independently of  $m$ ). For such an element  $e_m$  we will therefore have an element  $X_m$  of this class in our model  $\mathcal{M}$ . Finally  $\mathcal{M}$  is closed under Turing join because for all  $n_1, n_2$  there exists a code  $e_m$  for  $m > n_1, n_2$  such that  $X_{n_1} \oplus X_{n_2}$  is the only element of the class  $\Sigma_1^1(X_1 \oplus \dots \oplus X_{n_1} \oplus \dots \oplus X_{n_2} \oplus \dots \oplus X_{m-1})$  of code  $e_m$ . According to Proposition 5.2,  $\mathcal{M}$  is therefore a  $\beta$ -model of  $\text{ATR}_0$ . ■

**Corollary 5.5**

*$\text{ATR}_0$  does not imply  $\Pi_1^1\text{-CA}_0$ .*

PROOF. We have a  $\beta$ -model of  $\text{ATR}_0$  not containing  $\mathcal{O}$ . Since this is a  $\beta$ -model it cannot be a model of  $\Pi_1^1\text{-CA}_0$  because every  $\beta$ -model of  $\Pi_1^1\text{-CA}_0$  contains  $\mathcal{O}$ . ■

It is tempting to draw a parallel between the systems  $\text{ATR}_0$  and  $\text{WKL}_0$  as well as between the systems  $\text{ACA}_0$  and  $\Pi_1^1\text{-CA}_0$ . Where  $\text{ACA}_0$  asserts the existence of the Turing jump for any set,  $\Pi_1^1\text{-CA}_0$  asserts the existence of the Turing hyperjump for any set. Where we separate  $\text{WKL}_0$  from  $\text{ACA}_0$  by building a model of  $\text{WKL}_0$  not containing the jump, we separate  $\text{ATR}_0$

from  $\Pi_1^1\text{-CA}_0$  by building a  $\beta$ -model of  $\text{ATR}_0$  not containing the hyperjump. Correspondence has its limits, however. For instance, it is possible to build a countable model of  $\text{WKL}_0$  whose representation is low, but it is impossible to build a countable  $\beta$ -model of  $\text{ATR}_0$  which is hyperlow.

**Definition 5.6.** A set  $M \subseteq \mathbb{N}$  is a *code* of a countable class  $\mathcal{M} = \{X_n : n \in \mathbb{N}\}$  if  $M = \bigoplus_n X_n$ . ◇

Note that the same countable class has infinitely many codes, because the order of its elements is not fixed.

**Theorem 5.7**

*Let  $\mathcal{M}$  be a countable  $\beta$ -model of  $\text{ATR}_0$  encoded by  $M$ . Then,  $\mathcal{O} \leq_T M''$ .*

PROOF. It is easy to give an arithmetic definition in  $\mathcal{M}$ , of the set of codes of ill-founded tree. This is the set of codes  $e$  encoding a tree  $T_e$  such that there exists  $X \in \mathcal{M}$  belonging to  $[T_e]$ . The second-order existential quantification on  $X$  turns into a first-order existential quantification on  $M$ . The predicate  $X \in [T_e]$  is  $\Pi_1^0(X)$  and therefore  $\Pi_1^0(M)$ . So the set of codes of ill-founded trees is computable in  $M''$ . We deduce  $\mathcal{O} \leq_T M''$ . ■

We end this chapter by finally showing that the models of  $\text{ATR}_0$  or  $\Pi_1^1\text{-CA}_0$  are not necessarily  $\beta$ -models.

**Theorem 5.8**

*There are models of  $\Pi_1^1\text{-CA}_0$  that do not contain  $\mathcal{O}$ .*

PROOF. It suffices to consider the class of sets  $X$  which code for countable models of  $\Pi_1^1\text{-CA}_0$ . Second-order universal or existential quantifications are transformed into first-order quantifications on the elements  $X_1 \oplus X_2 \oplus \dots$  which are encoded by  $X$ . It is therefore an arithmetic class and in particular  $\Sigma_1^1$ . Since there are countable models of  $\Pi_1^1\text{-CA}_0$  this class is non-empty. According to Theorem 30-2.2 it contains an element  $X$  such that  $\omega_1^X = \omega_1^{ck}$ . So  $X$  codes for a countable model of  $\Pi_1^1\text{-CA}_0$  that cannot contain  $\mathcal{O}$ . ■

**Corollary 5.9**

*There are models of  $\Pi_1^1\text{-CA}_0$  — and therefore also of  $\text{ATR}_0$  — which are not  $\beta$ -models.*

PROOF. The model of the previous theorem cannot be a  $\beta$ -model according to Proposition 5.3. ■



# Appendix A

## Exercise solutions

### Chapter 2

**Solution 3.4.** We define  $g : \mathbb{N} \rightarrow \mathbb{N}$  as follows:  $g(0)$  is the smallest integer  $x$  such that  $f(x) \in B$ . Suppose  $g(n)$  defined. Then, we define  $g(n+1)$  as being the smallest integer  $x$  strictly greater than  $g(n)$  and such that  $f(x) \in B$ . Since  $B$  is infinite, then  $g$  is well defined everywhere. We then define  $h(n) = f(g(n))$ . By construction  $h$  is bijective.

**Solution 3.7.**

1. Let us show that  $(x_1, y_1) \neq (x_2, y_2)$  implies  $\alpha_2(x_1, y_1) \neq \alpha_2(x_2, y_2)$ . Suppose  $(x_1, y_1) \neq (x_2, y_2)$ . Suppose first that  $x_1 + y_1 < x_2 + y_2$  (the case  $x_2 + y_2 < x_1 + y_1$  being symmetric). Then,

$$\begin{aligned}
 & (\sum_{i=0}^{x_1+y_1} i) + x_1 + y_1 + 1 & \leq & \sum_{i=0}^{x_2+y_2} i \\
 \rightarrow & y_1 + \sum_{i=0}^{x_1+y_1} i & < & \sum_{i=0}^{x_2+y_2} i \\
 \rightarrow & y_1 + \sum_{i=0}^{x_1+y_1} i & < & y_2 + \sum_{i=0}^{x_2+y_2} i \\
 \rightarrow & \alpha_2(x_1, y_1) & < & \alpha_2(x_2, y_2)
 \end{aligned}$$

Suppose now that  $x_1 + y_1 = x_2 + y_2$ . Note that we then have  $y_1 = y_2$  implies  $x_1 = x_2$ . As  $(x_1, y_1) \neq (x_2, y_2)$ , then necessarily  $y_1 \neq y_2$ . So we have:

$$\begin{aligned}
 & (\sum_{i=0}^{x_1+y_1} i) & = & \sum_{i=0}^{x_2+y_2} i \\
 \rightarrow & y_1 + (\sum_{i=0}^{x_1+y_1} i) & \neq & y_2 + \sum_{i=0}^{x_2+y_2} i \\
 \rightarrow & \alpha_2(x_1, y_1) & \neq & \alpha_2(x_2, y_2)
 \end{aligned}$$

Let us now show that  $\alpha_2$  is surjective. To do this, let's first show that:

$$\begin{aligned}\alpha_2(x-1, y+1) - \alpha_2(x, y) &= 1 \quad \text{for } x > 0 \\ \alpha_2(y+1, 0) - \alpha_2(0, y) &= 1\end{aligned}$$

In the first case we have:

$$\begin{aligned}\alpha_2(x-1, y+1) - \alpha_2(x, y) &= (y+1 + \sum_{i=0}^{x-1+y+1} i) - (y + \sum_{i=0}^{x+y} i) \\ &= (y+1 + \sum_{i=0}^{x+y} i) - (y + \sum_{i=0}^{x+y} i) \\ &= 1\end{aligned}$$

In the second case we have:

$$\begin{aligned}\alpha_2(y+1, 0) - \alpha_2(0, y) &= (\sum_{i=0}^{y+1} i) - (y + \sum_{i=0}^y i) \\ &= (y+1 + \sum_{i=0}^y i) - (y + \sum_{i=0}^y i) \\ &= 1\end{aligned}$$

Let us now show by induction that for all  $n$ , there exists  $(x, y)$  such that  $\alpha_2(x, y) = n$ . We have  $\alpha_2(0, 0) = 0$ . Now suppose that  $\alpha_2(x, y) = n$ . If ever  $x > 0$ , then we therefore have  $\alpha_2(x-1, y+1) = n+1$ . If ever  $x = 0$ , we have  $\alpha_2(y+1, 0) = n+1$ . We deduce that  $\alpha_2$  is surjective.

## 2. Trivial

**Solution 3.9.** We easily verify that  $f : \mathbb{N} \rightarrow \mathbb{Z}$  given by  $f(2n) = n$  and  $f(2n+1) = -n-1$  is a bijection. We deduce from it with the help of Corollary 3.8 that  $\mathbb{Z} \times \mathbb{Z}$  is countable.

We construct an injection from  $\mathbb{Q}$  to  $\mathbb{Z} \times \mathbb{Z}$  by assigning to  $r \in \mathbb{Q}$  the ordered pair  $(p, q)$  with  $p \in \mathbb{Z}$  and  $q \in \mathbb{N}^*$  where  $p/q$  is the irreducible fraction equal to  $r$ . We therefore have  $|\mathbb{Q}| \leq |\mathbb{Z} \times \mathbb{Z}|$  and since  $\mathbb{Z} \times \mathbb{Z}$  is countable, we also have  $|\mathbb{Z} \times \mathbb{Z}| \leq |\mathbb{N}|$ . Since  $\mathbb{Q}$  is infinite,  $|\mathbb{N}| \leq |\mathbb{Q}|$  which gives  $|\mathbb{Q}| = |\mathbb{N}|$  by Cantor-Bernstein Theorem 2.3.

**Solution 3.10.** Let  $g : A \rightarrow \mathbb{N}$  be the function which to  $y \in A$  associates the smallest element  $n$  such that  $f(n) = y$ . The function  $g$  is injective because if  $x \neq y \in A$ , then  $\{n : f(n) = x\} \cap \{n : f(n) = y\} = \emptyset$ , therefore  $g(x) \neq g(y)$ . Thus,  $A$  is subpotent to  $\mathbb{N}$ . Since  $A$  is infinite, by Proposition 3.2,  $\mathbb{N}$  is subpotent to  $A$ , so by Cantor-Bernstein's theorem 2.3,  $A$  and  $\mathbb{N}$  are equipotent.

Hidden question: Proposition 3.2 uses the axiom of choice. How can we solve the exercise constructively, without appealing to Proposition 3.2?

**Solution 3.11.** We define the function  $h : \mathbb{N} \times \mathbb{N} \rightarrow B$  by  $h(n, m) = f_n(m)$ . The function is clearly surjective (bijective if the sets  $B_n$  are pairwise disjoint). By Proposition 3.5, there is a bijection  $b : \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$ . The function  $h \circ b : \mathbb{N} \rightarrow B$  is therefore a surjective function. As  $B$  is infinite, the previous exercise allows us to conclude.

## Chapter 3

**Solution 2.3.** From Exercise 2-3.7 the pairing bijection and its inverses can be computed as follows.

```
int pairing (int a, int b){
    return (a + b + 1) (a + b) / 2;
}

int inverse1 (int c){
    for (int i = 0; i <= c; i = i + 1)
    for (int j = 0; j <= c; j = j + 1)
    if (pairing (i, j) == c)
    return i;
}

int inverse2 (int c){
    for (int i = 0; i <= c; i = i + 1)
    for (int j = 0; j <= c; j = j + 1)
    if (pairing (i, j) == c)
    return j;
}
```

**Solution 2.4.**

```
int A (int a, int b){
    // Function to compute A
}

int ins1 (int x, int y){
    for (int i = 0; i <y; i = i + 1)
    if (A (x, i) == 0)
    return 0;
    return 1;
}

int ins2 (int x, int y){
    for (int i = 0; i <y; i = i + 1)
    if (A (x, i) == 1)
    return 1;
    return 0;
}
```

**Solution 6.4.**

1. Let  $x_1 \in A$  and  $x_2 \notin A$ . We define  $f$  by  $f(x) = x_2$  if  $x \in A$  and  $f(x) = x_1$  otherwise.
2. According to the fixed point theorem there exists  $e$  such that  $\Phi_{f(e)} = \Phi_e$ .
3. Trivial.
4. Trivial.

**Solution 7.3.** Let  $A$  be a computable set. Then, we define the code of a function  $e$  which halts on its input  $x$  if  $x \in A$ , and which loops forever if  $x \notin A$ .

**Solution 7.9.** We create the program which on an input  $n$ , enumerates  $A$  until an element greater than  $n$  is enumerated in  $A$ . Since  $A$  is infinite this will necessarily happen. At this time, if  $n$  is one of the elements listed in  $A$ , then  $n \in A$ , otherwise  $n \notin A$  (because only elements strictly greater than  $n$  will be listed thereafter).

**Solution 7.10.** Let  $A$  be an infinite c.e. set. We define a c.e. set  $B \subseteq A$  which we enumerate in order, as follows: we enumerate  $x_0$  in  $B$  for  $x_0$  the first element listed in  $A$ . Suppose  $x_0 < x_1 < \dots < x_n$  have been enumerated in  $B$ , then the element  $x_{n+1}$  enumerated in  $B$  will be the next element enumerated in  $A$  which is greater than  $x_{n+1}$ . Since  $A$  is infinite, this necessarily happens.

**Solution 7.11.** In the computation step  $t$ , for any  $e \leq t$  such that  $e$  is neither enumerated in  $A$  nor  $B$ , if  $\Phi_e(e)[t] \downarrow \in \{0, 1\}$ , then we enumerate  $e$  in  $A$  if  $\Phi_e(e)[t] \downarrow = 0$  and we enumerate  $e$  in  $B$  if  $\Phi_e(e)[t] \downarrow = 1$ . Thus, if  $\Phi_e$  computes a set  $C$ , then either  $A \not\subseteq C$  or  $C \cap B \neq \emptyset$ .

**Solution 7.12.** Let  $C$  be a c.e. set. We are going to enumerate in order two sets  $A, B$  (which will therefore be computable) such that  $x \in C$  iff  $2^x \in D_{A,B}$ .

Suppose  $A_t, B_t$  enumerated in step  $t$ . Let  $x$  be a new element enumerated in  $C$  in step  $t + 1$ . Let  $k$  be the smallest integer such that  $2^k > x$  and such that  $2^k$  is greater than all the elements of  $A_t$  and  $B_t$ . We then enumerate  $2^{k+1}$  in  $A$  and  $2^{k+1} - x$  in  $B$ . We have  $2^{k+1} - (2^{k+1} - x) = x$  in the set  $D_{A,B}$ . Moreover the other elements added in  $D_{A,B}$  on this occasion are of the form  $2^{k+1} - b$  for  $b \in B_t$  and of the form  $a - (2^{k+1} - x)$  for  $a \in A_t$ . As  $b < 2^k$  we have  $2^k < 2^{k+1} - b < 2^{k+1}$ . So  $2^{k+1} - b$  is not a power of 2. As  $a, x < 2^k$ , then  $a - (2^{k+1} - x)$  is negative and therefore does not belong to  $D_{A,B}$ . The only elements of  $D_{A,B}$  which are powers of 2 then encode the elements of  $C$ . So  $D_{A,B}$  computes  $C$ , and so if  $D_{A,B}$  were computable,  $C$  would be computable too.

**Solution 7.13.** Let us show that the uniform process given in the indication exists. If  $W_e$  enumerates a first element  $a_0$  then we enumerate  $[0, a_0[$  in  $W_d$ . Then for all  $n$ , if  $W_e$  enumerates an element  $a_{n+1} > a_n$ , we enumerate  $]a_n, a_{n+1}[$  in  $W_d$ . This ends the description of  $W_d$ . If  $W_e$  is finite, the set  $W_d$  is also finite

and therefore of infinite complement. If  $W_e$  is infinite, the sequence  $(a_n)_{n \in \mathbb{N}}$  such that  $a_n < a_{n+1}$  is well defined, and the set  $W_d$  then consists exactly of the complement of this sequence (the complement of  $W_d$  is therefore infinite). It is clear that if  $W_d \subseteq W_e$  and  $W_e$  is infinite, then  $W_e = \mathbb{N}$ .

We will say that “ $\mathcal{R}_e$  is satisfied” if the enumerated set  $W_d$  satisfies  $W_d \subseteq W_e \rightarrow (|\mathbb{N} \setminus W_e| < \infty \vee |W_e \setminus W_d| < \infty)$ . The difficulty now is in coordinating different strategies so that each  $\mathcal{R}_e$  is satisfied for *the same set*  $W_d$ . The idea is as follows: we carry out the process described above to build  $W_d$  so as to satisfy  $\mathcal{R}_0$ . This process enumerates a sequence of integers  $a_0^0 < a_1^0 < \dots <$  which will constitute the complement of  $W_d$  for the moment. There are two possibilities. Case 1: the sequence  $(a_n^0)_{n \in \mathbb{N}}$  is infinite, in which case we satisfy  $\mathcal{R}_1$  in the same way by considering  $(a_n^0)_{n \in \mathbb{N}}$  as our new workspace. Case 2: there is a maximal element  $a_k^0$ , in which case we can satisfy  $\mathcal{R}_1$  by considering  $\{m : m > a_k^0\}$  as our new workspace. The problem is that we must satisfy  $\mathcal{R}_1$  uniformly, and that we do not know whether the satisfaction of  $\mathcal{R}_0$  will lead to the “infinite” case or to the “finite” case. The idea is then to deal with the two possibilities simultaneously. To satisfy  $\mathcal{R}_1$ , we will consider the “finite” possibility, and so for each  $a_k^0$  which is the largest element for the moment, we construct, to satisfy  $\mathcal{R}_1$ , elements  $a_0^1 < a_1^1 < \dots < a_n^1$  inside  $\{m : m > a_k^0\}$ . Note that this impacts the work to satisfy  $\mathcal{R}_0$ , because the satisfaction of  $\mathcal{R}_1$  also contributes to the enumeration of  $W_d$ ; in practice this is not a problem. At the same time, in order to satisfy  $\mathcal{R}_1$ , we will consider the “infinite” possibility and try, as we enumerate  $a_0^0 < a_1^0 < \dots < a_n^0$ , to work in this space. We must then iterate this idea to satisfy each  $\mathcal{R}_e$ .

The formalization of the above argument is done naturally via an infinite injury priority construction, as detailed in Chapter 13. However for this precise construction, it is possible to give a surprisingly simple algorithm which implements “automatically” what has been explained. Given  $x$ , we denote by  $\sigma_e(x)[s]$  the number  $\sum_{i \leq e, x \in W_i[s]} 2^{e-i}$ . The idea is that  $\sigma_e(x)[s]$  is a weight given to  $x$  which increases as  $x$  belongs to more and more sets  $W_i$  for  $i \leq e$ . Moreover the membership of  $x$  to  $W_i$  has more value than the membership of  $x$  to the set of  $W_j$  for  $i < j \leq e$ . The algorithm is as follows. In step 0, let  $a_n^0 = n$  for all  $n$ . In step  $s + 1$ , we define  $a_0^{s+1}$  as the smallest integer  $x \in [0, s + 1] \cap \overline{W_d}$  such that  $\sigma_e(x)[s]$  is maximal — here  $\overline{W_d}$  denotes the complement of  $W_d$  at this computation step — and one enumerates  $[0, a_0^{s+1}[$  in  $W_d$ . Then inductively, if  $a_e^{s+1}$  is defined, one defines  $a_{e+1}^{s+1}$  as the smallest integer  $x \in ]a_e^{s+1}, s + 1] \cap \overline{W_d}[s]$  — if it exists — such that  $\sigma_e(x)[s]$  is maximal, then one enumerates  $]a_e^{s+1}, a_{e+1}^{s+1}[$  in  $W_d$ . This completes the construction. Note that for every  $e, x$ , we have  $\sigma_e(x)[s] \leq \sigma_e(x)[s + 1]$ . Moreover, there are  $e$  possible values for  $\sigma_e(x)[s]$ . So  $\lim_s \sigma_e(x)[s]$  exists. One deduces that  $a_0 = \lim_s a_0^s$  exists as well, and by induction that  $a_e = \lim_s a_e^s$  exists for every  $e$ . It is then clear that  $W_d$  is the complement of  $(a_e)_{e \in \mathbb{N}}$ , hence that  $\overline{W_d}$  is infinite. Suppose for contradiction that there exists  $e$  such that  $W_d \subseteq W_e$ ,  $|W_e \setminus W_d| = \infty$  and  $|\mathbb{N} \setminus W_e| = \infty$ . Let  $e$  be the smallest of these integers. In particular, there exists an integer  $a_n$  from which for every  $i < e$ , either  $a_m \in W_i$  for every  $m > n$ , or  $a_m \notin W_i$  for every  $m > n$ .

Let  $a_{m_3} > a_{m_2} > a_{m_1} > a_n$  be such that  $a_{m_2} \notin W_e$  and  $a_{m_1}, a_{m_3} \in W_e$ . Then,  $\sigma_{m_2}(a_{m_2}) < \sigma_{m_2}(a_{m_3})$  (remember that the membership of  $a_{m_3}$  to  $W_e$  matters more than its membership or not to  $W_i$  for  $e < i \leq m_2$ ). In particular, at some time, one must enumerate  $]a_{m_1}, a_{m_3}[$  in  $W_d$ , which is a contradiction.

## Chapter 4

**Solution 4.3.** If  $Y(n) = \Phi_a(X, n)$  for all  $n$  and  $Z(n) = \Phi_b(Y, n)$  for all  $n$ , then we can create the functional  $\Phi_e$  which, from the oracle  $X$ , searches, for any integer  $n$ , a string  $\sigma$  such that  $\sigma(i) = \Phi_a(X, i)$  for  $i < |\sigma|$  and such that  $\Phi_b(\sigma, n) \downarrow$ .

**Solution 5.3.** Trivial according to Exercize 4.3.

**Solution 5.4.** Suppose  $Y =^* X$ . To compute  $Y$  from  $X$ , it suffices to “hardcode” the variations of  $X$  in a program (a finite sequence of instructions), which on the oracle  $X$  will repeat its oracle on the output, except for the finite number of elements specified in the program itself.

**Solution 5.8.** Let  $e_1, e_2$  be such that  $\Phi_{e_1}(Y, n) \downarrow = X(n)$  and  $\Phi_{e_2}(B, n) \downarrow = A(n)$ . Then, we define the functional  $\Phi_e$  such that  $\Phi_e(Z_0 \oplus Z_1, 2n) = \Phi_{e_1}(Z_0, n)$  and such that  $\Phi_e(Z_0 \oplus Z_1, 2n+1) = \Phi_{e_2}(Z_1, n)$ .

**Solution 6.2.** Let  $\Psi$  be the functional such that  $\Psi(Y, n) \downarrow = X(n)$  for all  $n$ . We note  $\Psi(Y)$  for the set  $\{n \in \mathbb{N} : \Psi(Y, n) \downarrow = 1\}$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the computable function such that  $\Phi_{f(n)}(Y, m) = \Phi_n(\Psi(Y), n) \downarrow$  for all  $m$ . In particular  $f(n) \in Y'$  iff  $n \in X'$ . So  $X' \leq_T Y'$ .

**Solution 6.4.** Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be defined as in the proof of Proposition 6.3. The functional  $\Phi_e$  on the oracle  $Z$  and the input  $n$ , just checks if  $g(n) \in Z$ . If so then it returns 1, otherwise it returns 0.

**Solution 7.5.**  $\emptyset''$  is  $\emptyset'$ -c.e. and  $\emptyset'$  is  $\emptyset$ -c.e., but  $\emptyset''$  is not  $\emptyset$ -c.e. because all  $\emptyset$ -c.e. sets are limit-computable, and therefore computable in  $\emptyset'$ , but according to Proposition 6.3,  $\emptyset''$  is not computable in  $\emptyset'$ .

**Solution 7.8.** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be such that  $f(n) \geq \mu_A(n)$ . To compute  $A(n)$  we launch the enumeration of  $A$  up to step  $f(n)$ . We then have  $A_{f(n)}(n) = A(n)$ .

**Solution 10.5.** It suffices to repeat the same proof, but changing the condition (P2) as follows: for all  $(x, y) \in \text{dom } h \setminus \text{dom } g$ , if  $x < m$  then  $h(x, y) = B(x)$ .

**Solution 10.6.** The construction of Proposition 10.2 is computable in  $\emptyset''$ , we can however use the fact that  $\emptyset''$  is computably enumerable in  $\emptyset'$  so that the

construction becomes effective in  $\emptyset'$ . At a step  $t$ , we satisfy the requirement  $S_e$  from a pair  $(g, e)$  for a partial function  $g$ , on the basis of what we believe to be the correct prefix of  $\emptyset''$  of size  $e$  in step  $t$ .

When  $\emptyset'$  detects that  $k$  is listed in  $\emptyset''$  for some  $k$ , the requirements  $S_e$  for  $e \geq k$  lapse, because they were satisfied on the basis of a wrong prefix. One can however re-satisfy these requirements as many times as necessary, by using the fact that the enumeration of  $\emptyset''$  converges little by little.

**Solution 10.7.** It suffices to repeat the proof of Proposition 8.1, but with a countable quantity of sets: suppose that in step  $t$ , we have defined prefixes  $\sigma_{0,t}, \dots, \sigma_{t,t}$  of  $A_0, \dots, A_t$ , while making sure that no functional  $\Phi_e$  for  $e \leq t$  can compute  $\sigma_{i,t}$  from  $\sigma_{j,t}$  for  $i \neq j \leq t$ . In step  $t+1$ , we define extensions  $\sigma_{i,t+1}$  of each  $\sigma_{i,t}$ , as well as an extension  $\sigma_{t+1,t+1}$  of the empty word, so that for all  $e \leq t+1$ , the functional  $\Phi_e$  cannot compute  $\sigma_{i,t+1}$  from  $\sigma_{j,t+1}$  for  $i \neq j \leq t+1$ . The extensions are done little by little by diagonalizing on the functional  $\Phi_e$  for  $e \leq t+1$  and on all the possible pairs  $(i, j)$  for  $i, j \leq t+1$ .

## Chapter 5

**Solution 1.8.** There is only a countable number of sets in the arithmetic hierarchy, so there are non- $\Sigma_n^0$  sets. Let  $A$  be such a set. In particular  $A = \bigcup_{n \in A} \{n\}$ . Each singleton being computable, and *a fortiori*  $\Sigma_n^0$ ,  $A$  is a countable union of  $\Sigma_n^0$  sets, but is not  $\Sigma_n^0$ .

**Solution 5.7.** For the  $X \equiv_T Y \rightarrow X' \equiv_m Y'$  direction, it suffices to take Exercise 4-6.2 and notice that the defined reduction is in fact a many-one reduction. Suppose now  $X' \geq_m Y'$ . Let  $e_{n,0}$  and  $e_{n,1}$  be the codes of the programs which halt on their inputs if respectively  $Z(n) = 0$  or  $Z(n) = 1$  for the current oracle  $Z$ . In particular, for any oracle  $Z$ , we have  $Z'(e_{n,0}) \neq Z'(e_{n,1})$  and at least one of the two values is 1. Now, from  $X$  and from the function  $f$  such that  $x \in Y'$  iff  $f(x) \in X$ , on an input  $n$ , we enumerate  $X'$  until we find  $f(e_{n,0}) \in X'$  or  $f(e_{n,1}) \in X'$ . If the first event occurs, then  $Y(n) = 0$ , and otherwise  $Y(n) = 1$ .

**Solution 7.2.** We have  $e \in \text{TOT}$  iff  $\forall n \exists t \Phi_e(n)[t] \downarrow$  which is a  $\Pi_2^0$  predicate. Let now  $A = \{n \in \mathbb{N} : \forall x_1 \exists x_2 R(n, x_1, x_2)\}$  be a  $\Pi_2^0$  set, where  $R$  is a computable predicate. Given  $n$ , we define the functional  $\Phi_{e_n}$  such that  $\Phi_{e_n}(m)$  halts if  $\exists t R(n, m, t)$ . It is clear that  $e_n \in \text{TOT}$  iff  $n \in A$ .

**Solution 7.5.** If this were the case, the halting problem would be computable.

**Solution 7.9.** Trivial.

**Solution 7.10.** It suffices to create, using the fixed point theorem, the code  $e$  of a functional which computes a set  $X$  by choosing only 0 for the first bits of  $X$ , until  $f(e)$  returns a value: if this value is 0 the code  $e$  decides to put one of the bits not yet computed to 1, otherwise it continues to only put 0. More formally:

Let  $g$  be the computable function such that:

1.  $\Phi_{g(e)}(t) = 1$  if  $t$  is the smallest such that  $f(e)[t] \downarrow$  and if  $f(e)[t] \downarrow = 0$
2.  $\Phi_{g(e)}(t) = 0$  otherwise

According to the fixed point theorem, there exists a code  $a$  such that  $\Phi_a = \Phi_{g(a)}$ . Note that  $g(e)$  is always the  $\Delta_1^0$  code of a finite set. This is therefore the case for  $g(a)$ , and therefore also for  $a$ . By hypothesis on  $f$ , we must therefore have  $f(a) \downarrow = n$  for a certain  $n$ . If  $n = 0$ , then  $e$  codes for a one-element set. Otherwise  $e$  codes for the empty set.

**Solution 7.11.** Let  $b$  be an integer such that for any oracle  $X$ , we have  $\Phi_b(X, b) \downarrow$  if  $X$  is not empty, and such that  $\Phi_b(X, b) \uparrow$  otherwise. We define the function  $g : \mathbb{N} \rightarrow \mathbb{N}$  such that:

1.  $\Phi_{g(e)}(\emptyset', t) = 0$  if  $\Phi_{f(e)}(\emptyset', b)[t] \uparrow$
2.  $\Phi_{g(e)}(\emptyset', t) = 1$  if  $\Phi_{f(e)}(\emptyset', b)[t] \downarrow = 0$
3.  $\Phi_{g(e)}(\emptyset', t) = 0$  if  $\Phi_{f(e)}(\emptyset', b)[t] \downarrow = 1$

According to the fixed point theorem, there exists a code  $a$  such that  $\Phi_{g(a)} = \Phi_a$ . Note that  $g(e)$  always codes for a total function in  $\emptyset'$  and is therefore a  $\Delta_2^0$  code. Therefore  $g(a)$  and  $a$  are also  $\Delta_2^0$  codes. Finally, note that the set of code  $a$  is either empty or has only a finite number of 0. It is therefore computable and in particular low. So by hypothesis on  $f$ ,  $f(a)$  is a  $\Delta_2^0$  code and therefore  $\Phi_{f(a)}$  is total on the oracle  $\emptyset'$  and there exists  $t$  such that  $\Phi_{f(a)}(\emptyset', b)[t] \downarrow = i$ . If  $i = 0$ , then  $a$  codes for a non-empty set  $X$  and we should have  $X'(b) = 1 \neq \Phi_{f(a)}(\emptyset', b)$ . If  $i = 1$ , then  $a$  codes for an empty  $X$  set and we should have  $X'(b) = 0 \neq \Phi_{f(a)}(\emptyset', b)$ .

Hidden Exercise: Now, give an informal description of the procedure described above, like the one in the previous solution.

## Chapter 6

**Solution 3.10.** Let  $m$  be such that our program uses at most the registers  $R_0, \dots, R_m$ . All you have to do is copy the registers from  $R_1$  to  $R_m$  to the registers from  $R_{m+1}$  to  $R_{m+m}$  at the start of the program. Then the program is the same by replacing all the registers  $R_i$  by  $R_{i+m}$  for  $i > 0$ , then finally by adding instructions at the end which reset all the registers from  $R_{m+1}$  to  $R_{m+m}$ .

**Solution 3.12.** Addition is defined using the primitive composition and recursion schemes. Multiplication and exponential using the primitive recursion scheme.

Addition:

$$\begin{aligned}\text{add}(a, 0) &= p_1^2(a, 0) \\ \text{add}(a, b + 1) &= \text{succ}(p_2^2(a, \text{add}(a, b)))\end{aligned}$$

Multiplication:

$$\begin{aligned}\text{mult}(a, 0) &= 0 \\ \text{mult}(a, b + 1) &= \text{add}(a, \text{mult}(a, b))\end{aligned}$$

Exponential:

$$\begin{aligned}\text{exp}(a, 0) &= 1 \\ \text{exp}(a, b + 1) &= \text{mult}(a, \text{exp}(a, b))\end{aligned}$$

**Solution 3.13.** Functions are defined using the primitive recursion scheme.

<p>Predecessor:</p> $\begin{aligned}\text{pred}(0) &= 0 \\ \text{pred}(n + 1) &= p_1^2(n, \text{pred}(n))\end{aligned}$	<p>Subtraction:</p> $\begin{aligned}-(a, 0) &= p_1^2(a, 0) \\ -(a, b + 1) &= \text{pred}(p_2^2(a, -(a, b)))\end{aligned}$
---	---

**Solution 3.14.** Functions are defined using the primitive recursion scheme.

<p>sg:</p> $\begin{aligned}\text{sg}(0) &= 0 \\ \text{sg}(x + 1) &= c_1^2(x, \text{sg}(x))\end{aligned}$	<p><math>\overline{\text{sg}}</math>:</p> $\begin{aligned}\overline{\text{sg}}(0) &= 1 \\ \overline{\text{sg}}(x + 1) &= c_0^2(x, \overline{\text{sg}}(x))\end{aligned}$
--	--

**Solution 3.19.** Let  $P_1, P_2 \subseteq \mathbb{N}^n$  be primitive recursive predicates. Then,

$$\begin{aligned}P_1(a_1, \dots, a_n) \wedge P_2(a_1, \dots, a_n) &= P_1(a_1, \dots, a_n) \times P_2(a_1, \dots, a_n) \\ P_1(a_1, \dots, a_n) \vee P_2(a_1, \dots, a_n) &= \text{sg}(P_1(a_1, \dots, a_n) + P_2(a_1, \dots, a_n)) \\ \neg P_1(a_1, \dots, a_n) &= \overline{\text{sg}}(P_1(a_1, \dots, a_n)).\end{aligned}$$

Let  $P \subseteq \mathbb{N}^{n+1}$  be a primitive recursive predicate. Then,

$$\begin{aligned}\exists y \leq z \, P(x_1, \dots, x_n, y) &= Q_1(x_1, \dots, x_n, z) \\ \forall y \leq z \, P(x_1, \dots, x_n, y) &= Q_2(x_1, \dots, x_n, z)\end{aligned}$$

or

$$\begin{aligned}Q_1(x_1, \dots, x_n, 0) &= P(x_1, \dots, x_n, 0) \\ Q_1(x_1, \dots, x_n, z + 1) &= P(x_1, \dots, x_n, \text{succ}(z)) \vee Q_1(x_1, \dots, x_n, z) \\ &\text{and} \\ Q_2(x_1, \dots, x_n, 0) &= P(x_1, \dots, x_n, 0) \\ Q_2(x_1, \dots, x_n, z + 1) &= P(x_1, \dots, x_n, \text{succ}(z)) \wedge Q_2(x_1, \dots, x_n, z).\end{aligned}$$

**Solution 3.20.** We use the primitive recursion scheme, the definition by case, and the closure of the primitive recursive predicates under conjunction and bounded

quantification:

$$\begin{aligned}
 g(x_1, \dots, x_p, 0) &= 0 \\
 g(x_1, \dots, x_p, n+1) &= \text{succ}(n) && \text{if } f(x_1, \dots, x_p, \text{succ}(n)) = 0 \\
 & && \text{and } \forall t \leq n \ f(x_1, \dots, x_p, t) \neq 0 \\
 &= g(x_1, \dots, x_p, n) && \text{otherwise}
 \end{aligned}$$

**Solution 3.23.** The set  $A$  can be defined in a computable way by recursion.  $A_0 = \{e_0\}$ , where  $e_0$  is the code of the successor function. Suppose  $A_n$  defined with  $n = \langle k, a \rangle$ . Then,  $A_{n+1}$  is the set  $A_n$  to which we add a code for all the possibilities of application of the composition scheme and of primitive recursion of the codes of  $A_n$ , as well as the codes for the constant function  $f(x_1, \dots, x_k) = a$  and for the projection  $f(x_1, \dots, x_k) = x_a$  in the case where  $a \leq k$ . Care should be taken that the new codes of  $A_{n+1}$  are all greater than the codes of  $A_n$ , via the padding lemma. The set  $A = \bigcup_n A_n$  is then computable.

By diagonalization, by using the fact that a primitive recursive function is always total, we then compute a function  $\Phi_e$  which has no code in  $A$ . Let  $a_0 < a_1 < a_2 < \dots$  be the elements of  $A$  encoding functions taking exactly one parameter. We can then for example define  $\Phi_e(n) = 0$  if  $\Phi_{a_n}(n) \neq 0$  and  $\Phi_e(n) = 1$  otherwise. Note that it is necessary for all  $n$  to start the execution of  $\Phi_{a_n}(n)$  until it halts (which necessarily happens because  $a_n$  codes for a primitive recursive function).

**Solution 3.25.**

- (1) We have  $A_0(x) = 2^x > x$ , so (1) is true for  $A_0$ . Suppose (1) for  $A_n$  in order to prove (1) for  $A_{n+1}$ . We proceed by induction on  $x$ . We have  $A_{n+1}(0) = 1 > 0$ . Suppose  $A_{n+1}(x) > x$  for  $x$ . Then,  $A_{n+1}(x+1) = A_n(A_{n+1}(x))$ . Since (1) is true for  $A_n$  then  $A_n(A_{n+1}(x)) > A_{n+1}(x)$ . By induction hypothesis  $A_{n+1}(x) > x$ . So  $A_{n+1}(x+1) > x+1$ .
- (2) The function  $A_0$  is clearly strictly increasing. Then for all  $n$ , we have  $A_{n+1}(x+1) = A_n(A_{n+1}(x))$ . Using (1) we have  $A_n(A_{n+1}(x)) > A_{n+1}(x)$  and therefore  $A_{n+1}(x+1) > A_{n+1}(x)$ . So for all  $n$ , the function  $A_{n+1}$  is strictly increasing.
- (3) For  $x = 0$ , for all  $n$ , we have  $A_{n+1}(0) = 1$  and  $A_n(0) = 1$ . So  $A_{n+1}(0) \geq A_n(0)$ . For  $x > 0$ , we have  $A_{n+1}(x) = A_n(A_{n+1}(x-1))$ . According to (1), we have  $A_{n+1}(x-1) > x-1$  therefore  $A_{n+1}(x-1) \geq x$ . Also, as the function  $A_n$  is increasing, we have  $A_n(A_{n+1}(x-1)) \geq A_n(x)$ . We therefore have  $A_{n+1}(x) \geq A_n(x)$ , which implies that the function  $n \mapsto A_n(x)$  is increasing.
- (4) By induction on  $y$ . For  $y = 0$ , we have  $A_{n+1}(x) \geq x$  from (1). Suppose (4) for  $y$  in order to show it for  $y+1$ . We have  $A_{n+1}(x+y+1) = A_n(A_{n+1}(x+y))$ . By the induction hypothesis  $A_{n+1}(x+y) \geq A_n^{(y)}(x)$ . We therefore have  $A_{n+1}(x+y+1) \geq A_n(A_n^{(y)}(x)) = A_n^{(y+1)}(x)$  (because  $A_{n+1}$  is increasing).

- (5) From (4), we have  $A_{n+1}(2(x+1)) \geq A_n^{(x+1)}(x+1)$ . It suffices to show  $A_{n+1}(x) > A_n(2(x+1))$  for almost every  $x$  in order to have (5). By (3), for all  $n \in \mathbb{N}$ , we have  $A_{n+1}(x) \geq A_0(x) = 2^x > 2x + 3$  for almost all  $x$ . By definition, for all  $n \in \mathbb{N}$ , we have  $A_{n+1}(x) = A_n(A_{n+1}(x-1)) \geq A_n(2(x-1)+3)$  for almost all  $x$  (because  $n \mapsto A_n$  is increasing by (3)).
- (6) For basic primitive recursive functions (projections, constants and successor functions),  $P(f)$  is clearly verified. Suppose we have  $P(h)$  and  $P(g_1), \dots, P(g_n)$  for the primitive recursive functions  $h : \mathbb{N}^n \rightarrow \mathbb{N}$  and  $g_1, \dots, g_n : \mathbb{N}^m \rightarrow \mathbb{N}$ . Let us show  $P(f)$  for the function

$$f(x_1, \dots, x_m) = h(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m)).$$

There are  $k \in \mathbb{N}$  such that

$$h(x_1, \dots, x_n) < A_k(\max(x_1, \dots, x_n)) \text{ and } g_i(x_1, \dots, x_m) < A_k(\max(x_1, \dots, x_m))$$

almost everywhere and for every  $i \leq n$ . As  $A_k$  is increasing, we have

$$h(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m)) \leq A_k(A_k(\max(x_1, \dots, x_m))).$$

According to (5), for almost all  $x_1, \dots, x_m$ , we have  $A_k^{(2)}(\max(x_1, \dots, x_m)) < A_{k+2}(\max(x_1, \dots, x_m))$ . So  $P(f)$  is checked.

Suppose now that  $P(h)$  and  $P(g)$  are true for the primitive recursive functions  $h : \mathbb{N}^{n+2} \rightarrow \mathbb{N}$  and  $g : \mathbb{N}^n \rightarrow \mathbb{N}$ . Let us show that  $P(f)$  is true for the function

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n) \\ f(x_1, \dots, x_n, x+1) &= h(x_1, \dots, x_n, x, f(x_1, \dots, x_n, x)). \end{aligned}$$

In particular, there are  $k$  such that  $g(x_1, \dots, x_n) < A_k(\max(x_1, \dots, x_n))$  and  $h(x_1, \dots, x_n, z, x) < A_k(\max(x_1, \dots, x_n, z, x))$  for almost all  $x_1, \dots, x_n, z, x$ . We leave it to the reader to show by induction on  $x$  that we have

$$f(x_1, \dots, x_n, x) < A_k^{(x+1)}(\max(x_1, \dots, x_n))$$

for all  $x$ . We also have

$$A_k^{(x+1)}(\max(x_1, \dots, x_n)) \leq A_k^{(\max(x_1, \dots, x_n, x+1))}(\max(x_1, \dots, x_n, x+1))$$

or

$$A_k^{(\max(x_1, \dots, x_n, x+1))}(\max(x_1, \dots, x_n, x+1)) \leq A_{k+2}(\max(x_1, \dots, x_n, x))$$

for almost every  $x_1, \dots, x_n, x$  (from (5)). So

$$\forall^\infty x \ f(x_1, \dots, x_n, x) < A_{k+2}(\max(x_1, \dots, x_n, x))$$

and therefore  $P(f)$ . We therefore have  $P(f)$  for any primitive recursive function  $f$ . We deduce that the Ackermann function almost everywhere dominates all recursive functions.

## Chapter 7

**Solution 1.5.** If a set  $A$  is not immune, it contains an infinite c.e. set  $x_0, x_1, \dots$ . We can assume without loss of generality that  $x_i \neq x_j$  for  $i \neq j$  and by setting  $F_i = \{x_i\}$ , we obtain a c.e. array  $F_0, F_1, \dots$  for which  $A \cap F_n \neq \emptyset$  for all  $n$ . So  $A$  is not hyperimmune.

**Solution 2.2.** Let  $X$  be a set computing a DNC function  $f : \mathbb{N} \rightarrow \mathbb{N}$ . By the padding lemma, there exists an infinite computable set  $A = \{e_0 < e_1 < \dots\}$  of codes of the nowhere-defined function. Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be the function defined by  $g(x) = f(x)$  if  $x \notin A$ , and  $f(e_n) = X(n)$  otherwise. The function  $g$  is  $X$ -computable. Conversely, we can re-compute  $X$  from  $g$  by looking at the values of  $g$  at the positions of  $A$ . Thus,  $g$  is in the degree of  $X$ . Let us show that  $g$  is DNC. If  $x \notin A$ , then  $g(x) = f(x) \neq \Phi_x(x)$ . If  $x \in A$ , then  $\Phi_x$  is not nowhere-defined, so  $g(x) \neq \Phi_x(x)$ .

**Solution 2.9.** We construct a set  $A$  as a limit of strings  $\sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$  via a construction computable in  $\emptyset'$ . The way to make  $A$  non-computable is identical to that of the proof of Proposition 4-9.1.

To get  $A$  non-DNC, given a functional  $\Phi$  and a string  $\sigma_n$ , either there exists  $m \in \mathbb{N}$  such that  $\Phi(\sigma_n \tau, m) \uparrow$  for any string  $\tau$ , in which case  $\Phi(A, m)$  will be partial, or for all  $m$ , we have  $\Phi(\sigma_n \tau, m) \downarrow$  for a string  $\tau$ , in which case we can compute the total function  $f(m) = \Phi(\sigma_n \tau, m)$  for the first string  $\tau$  found such that  $\Phi(\sigma_n \tau, m) \downarrow$ . There is then some  $a$  such that  $\Phi_a(a) \downarrow = f(a)$ . We choose  $\sigma_{n+1} = \sigma_n \tau$  for  $\tau$  such that  $\Phi(\sigma_n \tau, a) \downarrow = f(a) = \Phi_a(a)$  and we will have  $\Phi(A, a) \downarrow = \Phi_a(a) \downarrow$ .

**Solution 3.2.** We proceed as in the indication. Either there exists  $n$  such that the function  $m \mapsto g(a_{n,m})$  is DNC relative to  $C$ , or for all  $n$ , there exists  $m$  such that  $\Phi_m(C, m) \downarrow = g(a_{n,m})$ . We then define the  $g \oplus C$ -computable function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $f(n)$  returns the smallest  $s$  such that  $\Phi_m(C, m)[s] \downarrow = g(a_{n,m})$  for  $m \leq s$ . Let us assume absurdly that there exists  $n$  such that  $\emptyset'(n) = 1$  and such that  $\emptyset'(n)[f(n)] = 0$ . Then,  $g(a_{n,m}) = \Phi_{a_{n,m}}(a_{n,m})$  which contradicts that  $g$  is DNC. So  $f$  limits the entry time of  $n$  into  $\emptyset'$ .

**Solution 4.2.** A direction is trivial. For the other direction, suppose that there exists a computable function  $g$  and an integer  $y$  for which  $g(x) > f(x)$  for all  $x \geq y$ . Then, the function  $g'(x)$  defined by  $g'(x) = f(x) + 1$  if  $x < y$  and by  $g'(x) = g(x)$  if not, is computable and dominates  $f$  everywhere. So  $f$  is not hyperimmune.

**Solution 4.5.** Let  $X \geq_T f$  with  $f$  hyperimmune. Then,  $X \geq_T g$  with  $g : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $g(n) = \langle f(n), X(n) \rangle$ . As we have  $g > f$  then  $g$  is hyperimmune, and it is clear that  $g$  computes  $X$ .

**Solution 4.6.** Let  $(f_n)_{n \in \mathbb{N}}$  be an enumeration of total computable functions. We define  $\sigma_0 = \epsilon$ . Suppose  $\sigma_n$  defined. Then, we define  $\sigma_{n+1}$  as being  $\sigma_n 0^{f_n(n)+1} 1$ . The set  $\{X\} = \bigcap_n [\sigma_n]$  is hyperimmune via the function which to  $n$  associates the  $n$ -th element of  $X$ .

**Solution 5.8.** Suppose  $T(\sigma)$  defined for any string  $\sigma$  of size  $n$ , such that for any  $e \leq n$ , we have  $\Phi_e(T(\sigma)) \not\leq T(\tau)$  for any  $\sigma, \tau$  of size  $n$ . Let  $(\sigma_{j,0})_{j \leq 2^{n+1}}$  be an enumeration of strings of the form  $T(\sigma)i$  for a string  $\sigma$  of size  $n$  and  $i \in \{0, 1\}$ . Let  $((a_j, b_j))_{j \in k}$  be an enumeration of distinct pairs of elements less than  $2^{n+1}$ . Suppose  $(\sigma_{j,t})_{j \leq 2^{n+1}}$  defined in step  $t$ . In step  $t+1$ , we define  $\sigma_{a_{t+1}, t+1}$  and  $\sigma_{b_{t+1}, t+1}$  as being extensions of  $\sigma_{a_{t+1}, t}$  and  $\sigma_{b_{t+1}, t}$  such that  $\Phi_{n+1}(\sigma_{a_{t+1}, t+1}) \not\leq \sigma_{b_{t+1}, t+1}$ . We define  $\sigma_{c, t+1} = \sigma_{c, t}$  for  $c \neq a_{t+1}$  and  $c \neq b_{t+1}$ . Once all the pairs  $((a_j, b_j))_{j \in k}$  have been considered, we finally define  $T(\sigma i)$  as being the string  $\sigma_{j,k}$  which extends  $T(\sigma)i$  for  $i \in \{0, 1\}$ .

**Solution 6.3.** Let us show  $(1) \Rightarrow (2)$ . Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be a function which eventually dominates every computable function. We proceed as in  $(2) \Rightarrow (3)$  of Theorem 6.2, but for functionals with values in  $\mathbb{N}$ .

Let us show  $(2) \Rightarrow (3)$ . We transform the list  $(f_n)_{n \in \mathbb{N}}$  by replacing each function  $f$  such that  $f(n) < f(n+1)$  by the set  $X$  such that  $f(n)$  is the  $n$ -th element of  $X$ . If, for a function  $f$ , we notice at a time  $f(n+1) \leq f(n)$ , we complete the corresponding set  $X$  by an infinite computable set fixed in advance.

Let us show  $(3) \Rightarrow (1)$ . It suffices to compute the function  $g : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $g(n)$  as being the sum of the  $n$ -th elements of the sets  $X_i$  for  $i \leq n$ .

**Solution 6.5.** Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the  $X$ -computable function which to  $n$  associates the smallest  $m$ , such that  $\mathbb{N} \setminus X$  contains  $2n$  elements smaller than  $m$ . Let us assume absurdly that there exists a computable function  $g$  such that  $\exists^\infty n \ g(n) > f(n)$ . We can assume without loss of generality that  $g(n) < g(n+1)$ .

Let us first show that there is an infinity of values  $n$ , such that  $\mathbb{N} \setminus X$  contains at least two elements in  $[g(n), g(n+1)[$ . Indeed, in the opposite case, let  $n_0$  be the smallest integer such that for all  $n \geq n_0$ ,  $[g(n), g(n+1)[$  contains at most one element of  $\mathbb{N} \setminus X$ . Let  $k$  be the number of elements of  $\mathbb{N} \setminus X$  less than  $g(n_0)$ . For any  $n > n_0$ , the number of elements of  $\mathbb{N} \setminus X$  less than  $g(n)$  is at most  $k + n - n_0$ . For  $n$  large enough, we have  $k + n - n_0 < 2n$ , in other words, there are less than  $2n$  elements smaller than  $g(n)$ , therefore  $g(n) < f(n)$ , which contradicts our hypothesis.

Let us now show that there exists a c.e. set  $Y \supseteq X$  such that  $Y \setminus X$  and  $\mathbb{N} \setminus Y$  are both infinite. We copy  $X$  into  $Y$ , and in addition to that, at each computation time  $s$ , we enumerate in  $Y$  the smallest integer of the interval  $[g(n), g(n+1)[$  (for  $n \leq s$ ) which is not listed in  $X$  at step  $s$ . Note that for each interval  $[g(n), g(n+1)[$  containing at least two elements of  $\mathbb{N} \setminus X$ , we will enumerate an element in  $Y$  which is not in  $X$ , and we will keep an element of  $\mathbb{N} \setminus X$  out of  $Y$ . The complement of  $Y$  will therefore be infinite, and an infinity of elements of  $Y$  will not be in  $X$ , which contradicts the hypothesis of maximality on  $X$ . So  $X$  is high.

## Chapter 8

**Solution 1.7.** Trivial.

**Solution 1.8.** Once a node  $\sigma_n \in T$  has been computed, this node has at least one extension in  $T$ . We then compute  $\sigma_{n+1}$  as being the leftmost extension of  $\sigma_n$  in  $T$  (i.e.  $\sigma_n 0$  if  $\sigma_n 0 \in T$  and  $\sigma_n 1$  otherwise).

**Solution 4.2.** Using  $\emptyset'$ , we can compute the subtree  $T' \subseteq T$  of the extendible nodes of  $T$ . According to Exercize 1.8, this tree contains a computable path in the tree and therefore  $\emptyset'$ -computable.

**Solution 5.2.** No particular difficulty: it consists of duplicating the construction made in the proof of Theorem 7-5.6, in the same way as the construction of a computably dominated degree in a  $\Pi_1^0$  class is duplicated in the proof of Theorem 5.1.

**Solution 5.3.** By taking the elements of the proof of Theorem 5.1, at a step  $n$ , we have, for each string  $\sigma \in 2^{<\mathbb{N}}$  of size  $n$ , non-empty pairwise disjoint  $\Pi_1^0$  classes  $\mathcal{P}_\sigma \subseteq \mathcal{P}$ . Let  $\Phi_n$  be the functional of code  $n$ . Let  $\sigma_0, \sigma_1$  be two distinct strings of size  $n$ . Either there exists  $X$  such that  $\Phi_n(Y) = X$  for all  $Y \in \mathcal{P}_{\sigma_0}$ , in which case  $X$  is computable, and therefore  $\Phi_n(Y) \notin \mathcal{P}_{\sigma_1}$  for all  $Y \in \mathcal{P}_{\sigma_0}$ . Or there exists  $m$  such that  $\{Y \in \mathcal{P}_0 : \Phi_n(Y, m) \uparrow \neq 0\} \neq \emptyset$  and  $\{Y \in \mathcal{P}_0 : \Phi_n(Y, m) \uparrow \neq 1\} \neq \emptyset$ . At least one of the two classes among  $\{Y \in \mathcal{P}_1 : Y(m) = 0\}$  and  $\{Y \in \mathcal{P}_1 : Y(m) = 1\}$  is not empty. We can therefore always find two non-empty  $\Pi_1^0$  classes  $\mathcal{P}'_{\sigma_0} \subseteq \mathcal{P}_{\sigma_0}$  and  $\mathcal{P}'_{\sigma_1} \subseteq \mathcal{P}_{\sigma_1}$  such that  $\Phi_n(Y) \notin \mathcal{P}'_{\sigma_1}$  for all  $Y \in \mathcal{P}'_{\sigma_0}$ .

It suffices to iterate this idea for all the pairs  $\sigma_0, \sigma_1$  of size  $n$ , and to start again by alternating these steps with those of the proof of Theorem 5.1, to diagonalize against all the functionals  $\Phi_n$ .

**Solution 5.4.** The difference is that we have no effectiveness here. We show the following lemma: given two strings  $\sigma$  and  $\tau$  which are the first branching node of  $f$ -trees  $S_\sigma$  and  $S_\tau$ , respectively, given a functional  $\Phi_e$ , there are extensions  $\sigma' \succeq \sigma$ ,  $\tau' \succeq \tau$  and a sub- $f$ -tree  $S'_\sigma \subseteq S_\sigma$  of which  $\sigma'$  is the first branching node, and such that  $\Phi_e(X) \not\preceq \tau'$  for all  $X \in [S'_\sigma]$ .

Let  $\rho_0, \rho_1$  be such that  $\tau\rho_0, \tau\rho_1 \in \text{Im } S_\tau$  are incompatible. Then, we have two possibilities: either there exists  $\sigma' \succeq \sigma$  with  $\sigma' \in \text{Im } S_\sigma$  such that  $\Phi_e(\sigma') \succeq \tau\rho_0$ . In this case, we find a subtree of  $S_\sigma$  such that none of its nodes which extend  $\sigma'$  is sent to  $\tau\rho_1$ . Or for all  $\sigma' \succeq \sigma$ , we have  $\Phi_e(\sigma') \not\preceq \tau\rho_0$ . In this case, no node from  $S_\sigma$  that extends  $\sigma$  is sent to  $\tau\rho_0$ .

Now, given any  $f$ -tree  $T$ , it suffices to construct a sub- $f$ -tree  $S$  of  $T$  by gradually iterating the lemma over each leaf of the finite chunk of  $S$  that we are in the process of building.

**Solution 5.5.** Let  $T \subseteq 2^{<\mathbb{N}}$  be a computable tree such that  $\mathcal{P} = [T]$ . Using the halting problem, one computes the set  $T' \subseteq T$  of the extendible nodes of  $T$ . We

can then find a path of  $[T']$  by diagonalizing against all computable sets (knowing that  $\emptyset'$  can uniformly list all computable sets).

**Solution 6.10.** Given  $n$ , we define the code  $e_n$  of the functional such that  $\Phi_{e_n}(e_n) = 0$  if  $n \in A$  and  $\Phi_{e_n}(e_n) = 1$  if  $n \in B$ . If neither case occurs, then  $\Phi_{e_n}(e_n) \uparrow$ . Let now  $f : \mathbb{N} \rightarrow \{0, 1\}$  be an  $X$ -computable function such that  $f(n) \neq \Phi_n(n)$  for all  $n$ . We then compute  $C(n) = f(e_n)$ .

**Solution 7.6.** If  $Y$  a  $\text{PA}(X)$  set, it can compute a path in any non-empty  $\Pi_1^0(X)$  class, and therefore in particular in a  $\Pi_1^0(X)$  class having  $X$  for only path.

**Solution 7.11.** The prefix closure of a set of strings is a tree. If this set of strings is infinite, it is an infinite tree, which admits a path by weak König's lemma.

**Solution 7.12.** Let  $\sigma_t = \emptyset'[t] \upharpoonright_t$  and  $S = \{\sigma_t : t \in \mathbb{N}\}$ . Then, the only path through the prefix closure of  $S$  is  $\emptyset'$ .

**Solution 7.13.** Let  $f : 2^{<\mathbb{N}} \times \mathbb{N} \rightarrow \{0, 1\}$  be a  $\Delta_2^0$  approximation of  $T$ , that is, such that for all  $\sigma \in 2^{<\mathbb{N}}$ ,  $\lim_y f(\sigma, y)$  exists, and  $\lim_y f(\sigma, y) = 1$  iff  $\sigma \in T$ . We will further assume that for all  $y$ ,  $\{\sigma \in 2^{<\mathbb{N}} : f(\sigma, y) = 1\}$  is closed under prefix and infinite.

We will define a computable sequence of strings  $\sigma_0, \sigma_1, \dots$  such that  $|\sigma_n| = n$  and such that the set of infinite paths of its prefix closure is  $[T]$ . Initially,  $\sigma_0 = \epsilon$ . Suppose we have defined  $\sigma_0, \dots, \sigma_n$ .

For any string  $\tau$  of length at most  $n$ , we define its *seniority*  $\mu_n(\tau)$  in step  $n$  as the largest integer  $s \leq n$  such that  $\tau \preceq \sigma_s$ , if it exists. Otherwise,  $\mu_n(\tau) = 0$ . Let  $\sigma_{n+1}$  be a string of length  $n+1$  such that  $f(\sigma_{n+1}, n+1) = 1$ , and such that for all  $s \leq n$ ,  $\sigma_{n+1} \upharpoonright_s$  is of minimum seniority among  $\{\rho \in 2^{\mathbb{N}} : |\rho| = s \wedge \rho \upharpoonright_{s-1} = \sigma_{n+1} \upharpoonright_{s-1}\}$ .

Let  $S = \{\sigma_n : n \in \mathbb{N}\}$ . Let us show that  $[\hat{S}] \subseteq [T]$ . Let  $P \in [\hat{S}]$  and  $\tau \prec P$ . Then, there exists an infinity of  $n$  such that  $\tau \prec \sigma_n$ . As  $f(\sigma_n, n) = 1$  and  $\{\rho \in 2^{<\mathbb{N}} : f(\rho, n) = 1\}$  are closed under prefix, there is an infinity of  $n$  such that  $f(\tau, n) = 1$ , so  $\lim_n f(\tau, n) = 1$ , in other words  $\tau \in T$ . Since every initial segment of  $P$  is in  $T$ ,  $P \in [T]$ .

Let us show that  $[T] \subseteq [\hat{S}]$ . Let  $P \in [T]$ . Let us assume absurdly that  $P \notin [\hat{S}]$ , and let  $\tau \prec P$  of minimum length such that  $[\tau] \cap [\hat{S}] = \emptyset$ . Let  $n_0$  be such that

- (1) for all  $n > n_0$ ,  $\sigma_n \upharpoonright_{|\tau|}$  is extendible into an infinite path in  $\hat{S}$
- (2) for any  $\rho \in 2^{<\mathbb{N}}$  of length  $|\tau|$  extendible into an infinite path in  $\hat{S}$ , there exists  $n < n_0$  such that  $\rho \prec \sigma_n$
- (3) for all  $n > n_0$ ,  $f(\tau, n) = 1$ .

By minimality of  $\tau$ , there exists  $n > n_0$  such that  $\tau \restriction_{|\tau|-1} \prec \sigma_n$ . By the choice of  $\sigma_n$ ,  $\sigma \restriction_{|\tau|}$  is a string extending  $\sigma \restriction_{|\tau|-1}$  of minimum seniority. By (2), all strings of length  $|\tau|$  extendible in  $\hat{S}$  are of strictly positive seniority, by (3), there exists a string extending  $\sigma \restriction_{|\tau|-1}$  of seniority 0, and by (1),  $\sigma \restriction_{|\tau|}$  is extendible into an infinite path. We get a contradiction. So  $P \notin [\hat{S}]$ .

## Chapter 10

**Solution 2.8.** Suppose that  $U^\prec$  is dense in  $2^{<\mathbb{N}}$ . Let  $\sigma \in 2^{<\mathbb{N}}$ . Let us show that  $[\sigma] \cap \bigcup_{\tau \in U} [\tau] \neq \emptyset$ . By density of  $U^\prec$ , there exists  $\rho \succeq \sigma$  such that  $\rho \in U^\prec$ . There is therefore a prefix  $\rho' \preceq \rho$  such that  $\rho' \in U$ . Either  $\sigma \preceq \rho'$ , in which case  $[\rho'] \subseteq [\sigma]$ , or  $\rho' \preceq \sigma$ , in which case  $[\sigma] \subseteq [\rho']$ . In all cases,  $[\rho'] \cap [\sigma] \neq \emptyset$  and therefore  $[\sigma] \cap \bigcup_{\tau \in U} [\tau] \neq \emptyset$ .

Suppose now that the open class  $\bigcup_{\tau \in U} [\tau]$  is dense in Cantor space. Let  $\sigma \in 2^{<\mathbb{N}}$ . In particular,  $[\sigma] \cap \bigcup_{\tau \in U} [\tau] \neq \emptyset$ , so there is some  $A \in [\sigma] \cap \bigcup_{\tau \in U} [\tau]$ . Let  $\tau \in U$  be such that  $A \in [\tau]$ . Let  $n > \max(|\sigma|, |\tau|)$ . In particular,  $A \restriction_n \succeq \tau$ , so by the suffix closure of  $U^\prec$ ,  $A \restriction_n \in U^\prec$ , and  $A \in [\sigma]$ , therefore  $A \restriction_n \succeq \sigma$ . The string  $\sigma$  therefore has an extension in  $U^\prec$ .

**Solution 2.12.** By Exercise 2.8, for all  $n$ , as  $W_n^\prec$  is dense in  $2^{<\mathbb{N}}$  and closed under suffix, then  $[W_n]$  is dense in Cantor space. Thus,  $\bigcap_n [W_n]$  is a countable intersection of dense open classes, so the class is co-meager.

**Solution 2.13.** Let  $G$  be a  $\vec{W}$ -generic set. In particular,  $G$  meets  $W_n$  for all  $n$ , so there exists  $\sigma \prec G$  such that  $\sigma \in W_n$ , thus  $G \in [W_n]$ , and therefore  $G \in \bigcap_n [W_n]$ . Conversely, let  $G \in \bigcap_n [W_n]$ . Then, for all  $n$ ,  $G \in [W_n]$ , so there exists  $\sigma \prec G$  such that  $\sigma \in W_n$ . In other words,  $G$  meets  $W_n$  for all  $n$ , so  $G$  is  $\vec{W}$ -generic.

**Solution 3.5.** As in (3)  $\rightarrow$  (2) of Theorem 3.2, but relativized to  $A$ .

**Solution 3.6.** Let  $f \leq_T G$  be a function infinitely equal to any  $A$ -computable function. Then,  $f + 1$  is infinitely often above any  $A$ -computable function.

**Solution 3.11.** The construction is similar to that of the proof of Theorem 3.9. We construct, for all  $e$ , a  $\emptyset''$ -computable function  $f_e$  which will differ from  $\Phi_e(X)$  for all  $X$  in a tree  $T : \mathbb{N}^{<\mathbb{N}} \rightarrow \mathbb{N}^{<\mathbb{N}}$  that we build little by little.

Suppose in step  $s - 1$  that  $T$  is already defined on the strings  $\sigma_0, \dots, \sigma_s$ , and that we have infinite c.e. reservoirs  $V_{t,s} \subseteq \mathbb{N}^{<\mathbb{N}}$  for  $t \leq s$ . In step  $s = \langle e, i \rangle$ , in sub-step  $t \leq s$ , we ask  $\emptyset''$  if there is an infinity of strings  $\mu \in V_{t,s}$  having an extension  $\mu' \succeq \mu$  for which  $\Phi_e(\mu', i) \downarrow$  is equal to a number whose  $t$ -th bit is 0. If the answer is yes then  $V_{t,s+1}$  is the set of these extensions, and otherwise  $V_{t,s+1}$  is the set  $V_{t,s}$  from which we remove a finite number of elements, so that all

possible extensions  $\mu'$  on which  $\Phi_e(\mu', i)$  halts are sent to a number whose  $t$ -th bit is 1. Once the answer has been obtained for all  $t \leq s$ , we construct a number of  $s$  bits which differs from all the possible computations by an oracle of our tree. We then define  $f_e(i)$  as being this number.

**Solution 3.22.** Let  $A$  be a computable set. Let  $W$  be the  $\Sigma_1^0$  set  $\{(A \upharpoonright_n) \cap (1 - A(n)) : n \in \mathbb{N}\}$ . By construction,  $A$  has no initial segment in  $W$ , but no initial segment of  $A$  avoids  $W$  either. So  $A$  does not have any initial segment in  $W \cup W^\perp$ , hence  $A$  is not 1-generic.

**Solution 3.23.** Given a string  $\sigma$  and a c.e. set  $U \subseteq 2^{<\mathbb{N}}$ ,  $\emptyset'$  may answer the question of whether  $\sigma$  has an extension in  $U$ , which is a  $\Sigma_1^0$  question. We thus easily construct a generic  $\emptyset'$ -computable set  $G$ . Since  $G$  is 1-generic, then  $G' \leq_T G \oplus \emptyset' \equiv \emptyset'$ . So  $G$  is low.

**Solution 3.25.** Recall that the principal function of an infinite set  $X = \{a_0 < a_1 < \dots\}$  is the function  $p_X$  which to  $i$  associates  $a_i$ . Let  $\Phi_e$  be a functional, and  $i$  an integer. Consider the open class

$$\mathcal{U}_{e,i} = \left\{ \bigoplus_{n \in \mathbb{N}} X_n : \exists m \ \Phi_e\left(\bigoplus_{j \neq i} X_j, m\right) \downarrow < p_{X_i}(m) \right\}.$$

Suppose there is a string  $\sigma$  such that  $[\sigma] \cap \mathcal{U}_{e,i} = \emptyset$ . Let us show that we necessarily have an integer  $m$  such that  $\Phi_e(\bigoplus_{j \neq i} X_j, m) \uparrow$  for all  $X = \bigoplus_{n \in \mathbb{N}} X_n$  such that  $\sigma \prec X$ . Suppose, for the absurd, the opposite. Let  $m > |\sigma|$ . Then, there exists  $X = \bigoplus_{n \in \mathbb{N}} X_n$  with  $\sigma \prec X$  such that  $\Phi_e(\bigoplus_{j \neq i} X_j, m) \downarrow = k \in \mathbb{N}$ . As  $[\sigma] \cap \mathcal{U}_{e,i} = \emptyset$ , we necessarily have  $p_{X_i}(m) \leq k$ . Now it suffices to consider  $Y = \bigoplus_{n \in \mathbb{N}} Y_n$  such that  $Y_j = X_j$  for  $j \neq i$ ,  $Y_i(m') = 0$  if  $m \leq m' \leq k$ , and  $Y_i(m') = X_i(m')$  otherwise. We then have  $\Phi_e(\bigoplus_{j \neq i} Y_j, m) \downarrow = k < p_{Y_i}(m)$ , hence  $Y \in \mathcal{U}_{e,i}$ , and as  $Y \succ \sigma$ , this contradicts our hypothesis on  $\sigma$ .

So if  $G = \bigoplus_{n \in \mathbb{N}} G_n$  is 1-generic, either  $G \in \mathcal{U}_{e,i}$ , or there is a prefix  $\sigma \prec G$  such that  $[\sigma] \cap \mathcal{U}_{e,i} = \emptyset$ , in which case there is  $m$  such that  $\Phi_e(\bigoplus_{j \neq i} G_j, m) \uparrow$ .

**Solution 3.26.** The proof is an adaptation of the direction (1)  $\rightarrow$  (3) of Theorem 3.2. For any  $i \in \mathbb{N}$ , let  $f_i$  be the principal function of  $X_i$ , that is to say the function which to  $n$  associates the  $n$ -th element of  $X_i$ . Note that  $f_i$  is increasing, and hyperimmune with respect to  $\bigoplus_{j \neq i} X_j$ . We compute from  $X$  a 1-generic set  $G \in 2^{\mathbb{N}}$ . Let  $(W_e)_{e \in \mathbb{N}}$  be an enumeration of the  $\Sigma_1^0$  subsets of  $2^{<\mathbb{N}}$ . We construct  $G$  by successive approximation  $\sigma_0 \preceq \sigma_1 \preceq \sigma_2 \preceq \dots$ .

We first describe a recursive procedure to be performed each time we want to concatenate a string  $\tau$  to a string  $\sigma$  that we have so far computed. This procedure, which we will name  $R$ , takes a third parameter: an integer  $e$ , which corresponds to the smallest integer such that  $\sigma\tau$  is enumerated in  $W_e$  at the computation step  $f_e(|\sigma|)$ . We will note  $R(\sigma, \tau, e)$  the result of the call to this procedure. Finally, note that some integers are marked as “satisfied” when the procedure is

called: these are the integers  $e$  such that  $\sigma$  extends a string of  $W_e$  at the current computation step.

The procedure  $R(\sigma, \tau, e)$  does the following: for each prefix  $\tau' \preceq \tau$  in order, it searches for the smallest integer  $e' < e$  that is not satisfied, and such that a string of the form  $\sigma\tau'\rho$  is listed in  $W_{e'}[f_{e'}(|\sigma\tau'|)]$ . If such an integer is found, then the procedure returns the result of the recursive call to  $R(\sigma\tau', \rho, e')$ . Otherwise, it returns  $\sigma\tau$ . Note that reducing the value of the last parameter in recursive calls causes the procedure to stop necessarily.

In step 0, we define  $\sigma_0 = \epsilon$ . Suppose  $\sigma_t$  defined in step  $t$ . In step  $t+1$ , we look for the smallest unsatisfied integer  $e \leq t+1$  such that a string of the form  $\sigma_t\tau$  is listed in  $W_e[f_e(|\sigma_t|)]$ . If we find such an integer  $e$ , we define  $\sigma_{t+1}$  as being  $R(\sigma_t, \tau, e)$ . Otherwise  $\sigma_{t+1}$  is  $\sigma_0$ . This concludes the construction.

Let us assume absurdly that  $G$  is not 1-generic. Let  $e$  be the smallest integer such that  $W_e$  is a dense set along  $G$ . In this case, the function  $f_e$  has no impact on the construction, which is then  $\bigoplus_{j \neq e} X_j$ -computable. In particular,  $f_e$  is  $G$ -hyperimmune. Let  $h$  be the total  $G$ -computable function which to  $n$  associates the smallest computation time  $t$  such that  $W_e$  enumerates a string extending  $G \upharpoonright_n$ . Since  $f_e$  is  $G$ -hyperimmune, there is an infinite number  $n$  of values such that  $f_e(n) > h(n)$ .

Let  $t$  be such that all integers  $e' < e$  which are satisfied at a time of the construction are satisfied at time  $t$ . Let  $n$  be the smallest integer greater than or equal to  $|\sigma_t|$ , such that  $f_e(n) > h(n)$ . Let  $s \geq t$  be the smallest integer such that  $|\sigma_s| \leq n < |\sigma_{s+1}|$ . If  $|\sigma_s| = n$ , then by minimality of  $e$ , the algorithm defines  $\sigma_{s+1} = \sigma_s\tau$  with  $\sigma_s\tau \in W_e[f_e(n)]$ . If not, then by construction, when defining  $\sigma_{s+1} = \sigma_s\tau$  for a certain string  $\tau$ , the algorithm checks for any prefix  $\tau' \preceq \tau$ , that we do not have an extension of  $\sigma_s\tau'$  listed in  $W_e[f_e(|\sigma_s\tau'|)]$ , and in particular for the prefix  $\tau'$  such that  $|\sigma_s\tau'| = n$ . If this is the case, the algorithm is restarted on this extension. As it is indeed the case by hypothesis, and by minimality of  $e$ , we will in fact have  $\sigma_s\tau \in W_e[f_e(n)]$  for  $\sigma_{s+1} = \sigma_s\tau$ . We conclude that  $W_e$  is not dense along  $G = \sigma_0 \prec \sigma_1 \prec \sigma_2 \prec \dots$ , hence that the set  $G$  is indeed 1-generic.

**Solution 3.30.** Let us show that there exists a weakly 1-generic set relative to  $A$  and left-c.e. relative to  $A$ . Let  $(W_n)_{n \in \mathbb{N}}$  be an enumeration of the  $\Sigma_1^0(A)$  sets of strings.

In step  $s$ , we compute with the help of  $A$  a string  $\sigma_{n,s}$  for all  $n \leq s$ , such that  $\sigma_{n,s} \prec \sigma_{n+1,s}$  for all  $n < s$ , and such that  $\sigma_{n,s-1}$  is lexicographically less than or equal to  $\sigma_{n,s}$ . In step  $s = 0$ , we define  $\sigma_{0,0}$  as being the string 0. Suppose  $\sigma_{n,s}$  defined in step  $s$  for all  $n \leq s$ . In step  $s+1$ , let  $n \leq s$  be the smallest integer such that  $\sigma_{n,s}$  is of the form  $\tau 0$ , and such that there is an extension  $\tau 1 \rho$  of  $\tau 1$  in  $W_n[s+1]$ . If no such integer  $n$  exists, then  $\sigma_{n,s+1} = \sigma_{n,s}$  for  $n \leq s$ , and we define  $\sigma_{s+1,s+1} = \sigma_{s,s+1}0$ . Otherwise, we define  $\sigma_{i,s+1} = \sigma_{i,s+1}$  for  $i < n$ ,  $\sigma_{n,s+1} = \tau 1 \rho$  and  $\sigma_{i+1,s+1} = \sigma_{i,s+1}0$  for  $s+1 \geq i \geq n$ . We leave it to the reader to check that  $\lim_{s \rightarrow +\infty} \sigma_{n,s} = \sigma_n$  is well defined and that the unique element  $G \in \bigcap_n [\sigma_n]$  is left-c.e. relative to  $A$  and weakly 1-generic relative to  $A$ .

Let us show that no weakly 1-generic set relative to  $A$  is left-c.e. relative to  $A$ . Let  $(X_s)_{s \in \mathbb{N}}$  be a left-c.e. approximation relative to  $A$  of a set  $X$ . We can assume without loss of generality that  $X_s \neq X_{s+1}$ . For any  $s$ , let  $n_s$  be the smallest integer such that  $X_s \upharpoonright_{n_s+1} \neq X_{s+1} \upharpoonright_{n_s+1}$ . Note that we necessarily have  $X_s(n_s) = 0$  and  $X_{s+1}(n_s) = 1$ . We enumerate in  $W$  — an  $A$ -c.e. set — the string  $X_s \upharpoonright_{n_s+1}$  for all  $s$ . We leave it to the reader to verify that  $W$  is dense along  $G$  without ever meeting  $G$ .

## Chapter 11

**Solution 1.7.** Let  $Q = \{d \in \mathbb{P} : d \text{ and } c \text{ are incompatible}\}$ . Let us show that  $D \cup Q$  is dense. Let  $d \in \mathbb{P}$ . Case 1:  $d$  and  $c$  are incompatible, in which case  $d \in Q$ . Case 2: there is  $d' \leq d, c$ . Knowing that  $D$  is dense below  $c$ , there exists  $e \leq d'$  such that  $e \in D$ . In particular,  $e \leq d$  and  $e \in D \cup Q$ . Thus,  $D \cup Q$  is dense. Knowing that  $F$  is sufficiently generic, it intersects  $D \cup Q$ . If it contains  $c$ , then by definition of a filter, all the elements of  $F$  are compatible with  $c$ , so  $F \cap Q = \emptyset$ . It follows that  $F$  intersects  $D$ .

**Solution 2.4.** Let  $F$  be a sufficiently generic filter containing  $c$ , and  $D = \{d \in \mathbb{P} : d \text{ forces } \mathcal{R}\}$ . By Exercise 1.7,  $F$  intersects the set  $D$ , therefore  $\dot{F}$  satisfies  $\mathcal{R}$ . It follows that  $c$  forces  $\mathcal{R}$ .

**Solution 2.8.** By induction on the complexity of  $\mathcal{R}$ .

Case 1:  $\mathcal{R}$  is  $\Sigma_1^0$  or  $\Pi_1^0$ . By definition, for all  $X \in [c]$ ,  $\mathcal{R}(X)$  is true. As  $d \leq c$ , then  $[d] \subseteq [c]$ , so for all  $X \in [d]$ ,  $\mathcal{R}(X)$  is true. Thus,  $d \Vdash^* \mathcal{R}$ .

Case 2:  $\mathcal{R}$  is of the form  $\exists x \mathcal{S}(x)$ , where  $\mathcal{S}(x)$  is  $\Pi_k^0$  for  $k \geq 1$ . By definition, there exists  $n \in \mathbb{N}$  such that  $c \Vdash^* \mathcal{S}(n)$ . By induction hypothesis,  $d \Vdash^* \mathcal{S}(n)$ , therefore  $d \Vdash^* \exists x \mathcal{S}(x)$ .

Case 3:  $\mathcal{R}$  is of the form  $\forall x \mathcal{S}(x)$  where  $\mathcal{S}(x)$  is  $\Sigma_k^0$  for  $k \geq 1$ . Then,  $c \Vdash^* \mathcal{R}$  implies  $\forall e \leq c \ e \Vdash^* \neg \mathcal{R}$  implies  $\forall e \leq d \ e \Vdash^* \neg \mathcal{R}$  implies  $d \Vdash^* \mathcal{R}$ .

**Solution 2.9.** By induction on the complexity of  $\mathcal{R}$ . Let  $U = \{c \in \mathbb{P} : c \Vdash^* \mathcal{R} \text{ or } c \Vdash^* \neg \mathcal{R}\}$ . We can assume that  $\mathcal{R}$  is  $\Sigma_n^0$  (otherwise we repeat the argument with  $\neg \mathcal{R}$  instead of  $\mathcal{R}$ ). Given a condition  $c$ , either there is an extension of  $d \leq c$  such that  $d \Vdash^* \mathcal{R}$ , or  $\forall d \leq c \ d \Vdash^* \neg \mathcal{R}$ , in which case  $c \Vdash^* \neg \mathcal{R}$ .

**Solution 2.10.** By induction on the complexity of  $\mathcal{R}$ .

Case 1:  $\mathcal{R}$  is  $\Sigma_1^0$  or  $\Pi_1^0$ . Suppose that  $c \Vdash^* \mathcal{R}$ . For any maximal filter  $F$  containing  $c$ , as  $\dot{F} \in [c]$ , then  $\dot{F}$  satisfies  $\mathcal{R}$ , so  $c \Vdash \mathcal{R}$ .

Case 2:  $\mathcal{R}$  is of the form  $\exists x \mathcal{S}(x)$ . Suppose that  $c \Vdash^* \exists x \mathcal{S}(x)$ . By definition, there exists an  $n \in \mathbb{N}$  such that  $c \Vdash^* \mathcal{S}(n)$ . By induction hypothesis,  $c$  forces  $\mathcal{S}(n)$ , therefore  $c \Vdash \exists x \mathcal{S}(x)$ .

Case 3:  $\mathcal{R}$  is of the form  $\forall x \mathcal{S}(x)$ . Suppose that  $c \Vdash^* \forall x \mathcal{S}(x)$ . By definition, for all  $d \leq c$  and  $n \in \mathbb{N}$ ,  $d \nVdash^* \neg \mathcal{S}(n)$ . By Exercize 2.9, for all  $n$ , the set  $\{d \in \mathbb{P} : d \Vdash^* \mathcal{S}(n)\}$  is dense below  $c$ . Let  $F$  be a sufficiently generic filter containing  $c$ . In particular, for all  $n \in \mathbb{N}$ , there exists  $d \in F$  such that  $d \Vdash^* \mathcal{S}(n)$ . By induction hypothesis,  $d \Vdash \mathcal{S}(n)$ , therefore  $\dot{F}$  satisfies  $\mathcal{S}(n)$ . It follows that  $\dot{F}$  satisfies  $\forall x \mathcal{S}(x)$ , so  $c \Vdash \forall x \mathcal{S}(x)$ .

**Solution 3.3.** It suffices to show that  $G$  is different from any set  $A$  fixed in advance. Let  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  be a computable f-tree. In particular,  $T(0)$  and  $T(1)$  are incomparable, so there exists  $i < 2$  such that  $T(i)$  is incomparable with  $A$ . The f-tree  $S : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  defined by  $S(\sigma) = T(i\sigma)$  is a computable sub-f-tree of  $T$  such that for all  $P \in [S]$ ,  $P \neq A$ .

**Solution 3.5.** Let  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  be a Sacks forcing condition and  $\Phi_e$  a functional. Suppose that for all  $n$ , there exists  $i_n$  such that for all  $\sigma \in 2^{<\mathbb{N}}$ , we have  $\Phi_e(T(\sigma), n) \downarrow \rightarrow \Phi(T(\sigma), n) \downarrow = i_n$ . Then, if  $\Phi_e$  is total over  $X \in [T]$ , we necessarily have  $\Phi_e(X) = Y$  with  $Y(n) = i_n$  for all  $n$ . Such a set  $Y$  is computable because it suffices to know  $Y(n)$  to look for  $\sigma \in 2^{<\mathbb{N}}$  such that  $\Phi_e(T(\sigma), n) \downarrow$ . So  $\Phi_e(X) \neq A$ . In particular,  $\Phi_e(X) \neq A$  for all  $X \in [T]$ .

Otherwise there are  $n$  and  $\sigma_1, \sigma_2 \in 2^{<\mathbb{N}}$  such that  $\Phi_e(T(\sigma_1), n) \nmid \neq \Phi_e(T(\sigma_2), n) \downarrow$ . We can consider without loss of generality that  $\Phi_e(T(\sigma_1), n) \nmid \neq A(n)$ . We then take the extension  $S \leq T$  defined by  $S(\tau) = T(\sigma_1\tau)$ .

The set of conditions  $T$  such that  $T \Vdash^* \exists n \Phi_e(G, n) \uparrow$  or  $\exists n T \Vdash^* \Phi_e(G, n) \nmid \neq A(n)$  is dense. So if  $G$  is sufficiently generic for Sacks forcing, it does not compute  $A$ .

**Solution 3.6.**

1. Let  $T : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}}$  be a Sacks forcing condition and  $\Phi_e$  a functional. If  $\exists n \exists \sigma \forall \tau \succeq \sigma \Phi_e(T(\tau), n) \uparrow$ , we consider the extension  $S \leq T$  defined by  $S(\rho) = T(\sigma\rho)$ . We thus force the partiality of  $\Phi_e$ .

Otherwise  $\forall n \forall \sigma \exists \tau \succeq \sigma \Phi_e(T(\tau), n) \downarrow$ , we construct a condition  $S \leq T$  such that  $\Phi_e(S(\sigma), |\sigma|) \downarrow$  for all  $\sigma \in 2^{<\mathbb{N}}$ . The size of the set  $T_n$  of values computed by strings  $S(\sigma)$  for  $\sigma$  of size  $n$ , is bounded by  $2^n$ .

2. Let  $X$  be computably traceable with a computable bound  $h$ . Let  $h'$  as in the statement. Let  $(A_n)_{n \in \mathbb{N}}$  be a sequence of consecutive intervals, forming a partition of  $\mathbb{N}$ , and such that  $h(\min A_n) < h'(\max A_n)$ . We denote by  $\langle f(x) \rangle_{x \in A_n}$  for the integer  $\langle f(x_0), \dots, f(x_k) \rangle$ , where  $x_0, \dots, x_k$  are the elements of  $A_n$ .

Given  $f \leq_T X$ , we consider  $g \leq_T f$  defined by  $g(n) = \langle f(x) \rangle_{x \in A_n}$ . Let  $(T_n)_{n \in \mathbb{N}}$  be a trace of  $g$  with  $|T_n| \leq h(n)$ . We can decompose the trace  $(T_n)_{n \in \mathbb{N}}$  into a trace  $(S_n)_{n \in \mathbb{N}}$  for the function  $f$  which will be such that  $|S_n| \leq h'(n)$  for  $n > 0$  (because we have  $h(\min A_n) < h'(\max A_n)$ ).

3. For each condition  $T$ ,  $[T]$  contains functions  $f : \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) < 2^n$ . The objective is to show that if such a function  $g$  is sufficiently generic, then it will be computably dominated, and for any trace  $(T_n)_{n \in \mathbb{N}}$  such that  $|T_n| \leq n$ , we will have  $g(m) \notin T_m$  for a certain  $m$ . According to (2), this is sufficient to show that the set  $X_g \in 2^{\mathbb{N}}$  which encodes  $g$  is not computably traceable, because  $X_g$  computes  $g$  which does not admit any trace  $(T_n)_{n \in \mathbb{N}}$  with  $|T_n| \leq n$ .

The way of forcing the set to be computably dominated is similar to that of Sacks forcing: either there exists  $n \in \mathbb{N}$  and there is  $\sigma \in T$  such that  $\forall \tau \succeq \sigma$  with  $\tau \in T$ , we have  $\Phi_e(\tau, n) \uparrow$ , in which case  $S = T \restriction_\sigma$  is an extension which forces partiality, i.e., for all  $n \in \mathbb{N}$  and for all  $\sigma \in T$  there exists  $\tau \succeq \sigma$  with  $\tau \in T$  such that  $\Phi_e(\tau, n) \downarrow$ . In this case, for any node  $\sigma \in T$  with maximum branching, one can find for any  $i < 2^{|\sigma|}$  extensions  $\tau_i \succeq \sigma i$  such that  $\Phi_e(\tau_i, n) \downarrow$ . We use this to construct an extension  $S \subseteq T$  such that  $\Phi_e$  is total on the paths of  $[S]$ , as well as a computable function bounding  $n \mapsto \Phi(X, n)$  for all  $X \in [S]$ .

By nature of the forcing conditions, given a computable trace  $(T_n)_{n \in \mathbb{N}}$  with  $|T_n| \leq n$ , we have a string  $\tau \in T$  such that  $\tau i \in T$  and  $i \notin T_{|\tau|}$ . We can then extend  $T$  into the condition  $S \subseteq T$  defined as the set of elements of  $T$  compatible with  $\tau i$ , in order to force them not to be traced by  $(T_n)_{n \in \mathbb{N}}$ .

**Solution 4.4.** (a) Let  $T \in \mathbb{P}$  and  $\sigma \in 2^{<\mathbb{N}}$  such that  $[T] \cap [\sigma] \neq \emptyset$ . Let  $S = \{\tau \in T : \tau \succeq \sigma \vee \tau \prec \sigma\}$ . Then,  $S$  is an extension of  $T$  such that  $[S] \subseteq [\sigma]$ . Thus,  $(\mathbb{P}, \leq)$  is a Cantor forcing.

(b) Let us show that, for any filter  $F \subseteq \mathbb{P}$  sufficiently generic for  $\mathbb{P}$ , the filter  $\hat{F} = \{[T] : T \in F\}$  is sufficiently generic for Jockusch-Soare forcing. Let  $\hat{D}$  be a dense set in Jockusch-Soare forcing. Let us show that  $D = \{T \in \mathbb{P} : [T] \in \hat{D}\}$  is dense in  $\mathbb{P}$ . Let  $T \in \mathbb{P}$ . By density of  $\hat{D}$ , there exists a  $\Pi_1^0$  class  $\mathcal{C} \subseteq [T]$  such that  $\mathcal{C} \in \hat{D}$ . By Proposition 8-3.3, there exists a computable infinite tree  $S \subseteq 2^{<\mathbb{N}}$  such that  $[S] = \mathcal{C}$ . Note that  $S \cap T$  is also a computable infinite tree such that  $[S \cap T] = \mathcal{C}$ , and such that  $S \cap T$  is an extension of  $T$ . So  $D$  is dense in  $\mathbb{P}$ . It follows that any set sufficiently generic for  $\mathbb{P}$  is sufficiently generic for Jockusch-Soare forcing.

Let us show that for any filter  $\hat{F}$  sufficiently generic for Jockusch-Soare forcing,  $F = \{T \in \mathbb{P} : [T] \in \hat{F}\}$  is sufficiently generic for  $\mathbb{P}$ . Let  $D \subseteq \mathbb{P}$  be a dense set for  $\mathbb{P}$ . Let us show that  $\hat{D} = \{[T] : T \in D\}$  is dense for Jockusch-Soare forcing. Let  $\mathcal{C} \subseteq 2^{\mathbb{N}}$  be a non-empty  $\Pi_1^0$  class. By Proposition 8-3.3, there exists a computable infinite tree  $T \subseteq 2^{<\mathbb{N}}$  such that  $[T] = \mathcal{C}$ . By density of  $D$ , there exists an extension  $S$  of  $T$  in  $D$ . In particular,  $[S]$  is an extension of  $\mathcal{C}$  in  $\hat{D}$ , so  $\hat{D}$  is dense. It follows that any set sufficiently generic for Jockusch-Soare forcing is sufficiently generic for  $\mathbb{P}$ .

(c) Let  $T \in \mathbb{P}$  be an infinite computable tree and let  $S = \{\sigma \in T : \forall n < |\sigma| \Phi(\sigma, n) \uparrow\}$ . If  $S$  is infinite, then  $S$  is an extension of  $T$  such that  $S \Vdash^\circ$

$\forall n \Phi(G, n) \uparrow$ . If  $S$  is finite, let  $t$  be large enough such that no string of length  $t$  is in  $S$ . Then, for all  $\sigma \in T$  of length  $t$ ,  $\exists n < t \ \Phi(\sigma, t) \downarrow$ , so  $T \Vdash^\circ \exists n \Phi(G, n) \downarrow$ .

(d) Immediate.

#### Solution 4.7.

1. We will define, at each step  $s$ , a perfect tree  $T_s \subseteq 2^{<\mathbb{N}}$  computable uniformly in  $s$ , with  $T_{s+1} \subseteq T_s$ . Given a perfect tree  $T$ , we define  $T \upharpoonright_n$  as the prefix closure of the set of strings  $\sigma \in T$  having exactly  $n$  branching prefixes in  $T$ . Then,  $T \upharpoonright_n$  is a finite tree with  $2^n$  leaves.

Let  $(\mathcal{P}_e)_{e \in \mathbb{N}}$  be an enumeration of the  $\Pi_1^0$  classes. In step 0, we define  $T_0 = 2^{<\mathbb{N}}$ . Suppose  $T_s$  defined in step  $s$ . In step  $s+1$ , we consider the smallest  $e \leq s+1$  such that for a leaf  $\sigma \in T_s \upharpoonright_e$ , we have  $\mathcal{P}_e[s+1] \cap [\sigma] \neq \emptyset$  and  $\mathcal{P}_e[s+1] \cap [\sigma] \subsetneq [T_s] \cap [\sigma]$ . If such an  $e$  and such a string  $\sigma$  exist, we define  $\tau$  as being an extension of  $\sigma$  belonging to  $T_s$ , and such that  $\mathcal{P}_e[s+1] \cap [\tau] = \emptyset$ , then we define  $T_{s+1}$  as the union of all the strings of  $T_s$  incomparable with  $\sigma$ , and of all the strings of  $T_s$  comparable with  $\tau$ . We then say that “ $T_s$  is modified because of  $\mathcal{P}_e$ ”. This concludes the construction.

Given  $n$ , we denote by  $s_n$  the smallest computation time such that  $T_{s_n} \upharpoonright_n = T_t \upharpoonright_n$  for all  $t \geq s_n$ . Let us show by induction on  $n$  that  $s_n$  exists for all  $n$ . Only one modification can happen because of  $\mathcal{P}_0$ : that of the first branching node  $\sigma \in T_t \upharpoonright_0$ . If the modification does indeed occur at a stage  $t$ , then we have  $\mathcal{P}_0[t+1] \cap [\sigma] = \emptyset$  for  $\sigma$  the only leaf of  $T_{t+1} \upharpoonright_0$ , and by construction no other modification will happen because of  $\mathcal{P}_0$ , since  $\mathcal{P}_0[t] \cap [\sigma] = \emptyset$ . Moreover, a modification due to  $\mathcal{P}_e$  for  $e > 0$  is done on the leaves of  $T_t \upharpoonright_e$ , and therefore leaves  $T_t \upharpoonright_0$  unchanged. So  $s_0 = t+1$ . By iterating this idea, and using the fact that  $\mathcal{P}_e$  can only make  $2^e$  modifications on  $T_t \upharpoonright_e$  for  $t \geq s_{e-1}$ , we come to the conclusion. We deduce that  $T = \bigcap_s T_s$  is a perfect tree, and therefore  $[T]$  a perfect  $\Pi_1^0$  class.

Let us show that  $[T]$  is thin. Let  $\mathcal{P}_e$  be a  $\Pi_1^0$  class such that  $\mathcal{P}_e \subseteq [T]$ . Let us assume by the absurdity that  $\mathcal{P}_e \cap [\sigma_i] \neq \emptyset$  and  $\mathcal{P}_e \cap [\sigma_i] \subsetneq [T] \cap [\sigma_i]$  for some  $\sigma_i \in T \upharpoonright_e$ . So this has to be detected at some stage  $t > s_e$ , in which case by construction,  $T_{t+1} \upharpoonright_e \neq T_t \upharpoonright_e$ , which contradicts the definition of  $s_e$ .

2. Let  $\mathcal{P}$  be a thin  $\Pi_1^0$  class and  $X \in \mathcal{P}$ . Given a functional  $\Phi_e$ , let  $\mathcal{Q} = \{Y \in \mathcal{P} : \Phi_e(Y, e) \uparrow\}$ . Using  $\emptyset'$ , we first determine whether  $\mathcal{Q}$  is empty. If this is the case, then  $e \in Y'$ . Otherwise, we search with  $\emptyset''$  for strings  $\sigma_0, \dots, \sigma_n$  such that  $\mathcal{Q} = \mathcal{P} \cap ([\sigma_0] \cup \dots \cup [\sigma_n])$ . Once such strings have been found, we check if one of them is a prefix of  $Y$ . If this is the case, then  $e \notin Y'$ , otherwise  $e \in Y$ .

**Solution 4.8.** Let us note first that any class of containing only Martin-Löf random numbers has positive measure. The idea is to repeat the proof of Theorem 4.6, using Lemma 18-3.3 in order to construct in any  $\Pi_1^0$  class  $\mathcal{P}$  of positive measure a perfect  $\Pi_1^0$  subclass  $\mathcal{S} \subseteq \mathcal{P}$ , such that for any  $\sigma$  for which  $\mathcal{S} \cap [\sigma] \neq \emptyset$ , there exists  $\tau \succeq \sigma$  for which  $\mathcal{S} \cap [\tau] = \emptyset$  and  $\mathcal{P} \cap [\tau] \neq \emptyset$ .

**Solution 4.17.** Let us show that  $c \text{?} \vdash \mathcal{R}$  satisfies the properties (1) and (2) of the definition of a forcing question. (1) This property is verified by definition of  $c \text{?} \vdash \mathcal{R}$ . (2) If  $c \text{?} \not\vdash \mathcal{R}$ , then for any extension  $d \leq c$ ,  $d \not\vdash \mathcal{R}$ . As the set  $\{d : d \Vdash \mathcal{R} \text{ or } d \Vdash \neg \mathcal{R}\}$  is dense, there exists an extension  $d \leq c$  such that  $d \Vdash \neg \mathcal{R}$ . In other words,  $c \text{?} \vdash \neg \mathcal{R}$ .

**Solution 4.26.** The proof is exactly that of Proposition 4.25, where we are no longer restricted to functions with values in  $\{0, 1\}$ . We therefore avoid computing a DNC function with arbitrary values. In case 1, we end up with a countable quantity of properties  $\neg(\Phi_e^{G^{(n)}}(m) \downarrow = v)$  for  $v \in \mathbb{N}$  to be forced simultaneously, which is possible because the forcing question is  $\Pi$ - $\omega$ -merging.

## Chapter 14

**Solution 1.10.** It suffices to consider the leftmost path  $X$  of the class. This path can be approached from the left, and it makes it possible to compute the c.e. set  $\{\sigma \in 2^{<\mathbb{N}} : \sigma \text{ is lexicographically on the left of } X\}$ . Note that this set makes it possible to recompute  $X$ . If a  $\Pi_1^0$  class only contains sets of minimal degrees, then its left-most path is non-computable, and therefore allows us to compute a non-computable c.e. set, which is then not of minimal degree: contradiction.

**Solution 1.12.** It suffices to consider the class

$$\{X \oplus Y : \forall t \forall e \Phi_e(e)[t] \neq X(e) \wedge \Phi_e(Y, e)[t] \neq Y(e)\}.$$

The members  $X \oplus Y$  of this class exactly those such that  $X$  is a DNC<sub>2</sub> set and  $Y$  is a DNC<sub>2</sub> set relative to  $X$ . In particular, by Exercize 8-7.6,  $Y \geq_T X$  but  $X \not\geq_T Y$ .

## Chapter 16

**Solution 1.7.** This is a special case of Schoenfield's limit lemma 4-7.2, applied to functions rather than sets of integers.

**Solution 1.9.** Let  $f \geq \Sigma$ . Then,  $U(\sigma) \downarrow$  iff  $U(\sigma)[f(|\sigma|)] \downarrow$ . So  $f$  allows to compute  $U$  and therefore  $\emptyset'$ .

**Solution 1.10.** Let us show that for  $n$  sufficiently large, there exists a string of length  $2n$ , of complexity less than  $2n$  but such that no program of length less than  $2n$  can compute it in less than  $\Sigma(n)$  time steps. Given  $n$  let  $\sigma_n$  be the "busy beaver" for  $n$ , that is to say with  $|\sigma_n| \leq n$  and  $\Sigma(n) = t$  where  $t$  is the smallest computation time such that  $U(\sigma_n)[t] \downarrow$ . For  $n$  fixed we consider the smallest string  $\tau_n$  corresponding to the program which searches for the smallest computation time  $t$  such that  $U(\sigma_n)[t] \downarrow$ , which computes  $U(\rho)[t]$  for any string  $\rho$

of length strictly less than  $2n$ , and finally writes a string of length  $2n$  which is not produced by such a program. As  $\sum_{i < 2n} 2^i = 2^{2n} - 1$  such a string of length  $2n$  necessarily exists.

For  $n$  large enough the length of  $\tau_n$  is smaller than  $2n$ , since the information needed for this program is a string of length less than or equal to  $n$ , the knowledge of the number  $n$  itself, whose encoding has approximate length  $\log_2(n)$ , and the rest of the program which is constant for all  $n$ . In particular for  $\sigma$  such that  $U(\tau_n) \downarrow = \sigma$  we therefore have  $C(\sigma) < 2n$ . Also by construction no program of length less than  $2n$  can produce  $\sigma$  in a computation time less than  $\Sigma(n)$ . We can therefore compute a bound of  $\Sigma(n)$  from the knowledge of  $C$ , by looking for any sufficiently large  $n$ , the complexity of all the strings of length  $2n$ , by searching for each of them the smallest program allowing to produce them, then by considering the maximal computation time needed. This computation time necessarily bounds  $\Sigma(n)$ . By (1) we can therefore compute  $U$ .

**Solution 1.11.** Let  $c$  be the constant such that  $C(\sigma) \leq |\sigma| + c$  for any string  $\sigma$ . It is then enough to use the low basis theorem on the following  $\Pi_1^0$  class:

$$\mathcal{P} = \{I : 2^{<\mathbb{N}} \rightarrow 2^{<\mathbb{N}} : (\forall \sigma \ |I(\sigma)| \leq |\sigma| + c) \wedge (\forall t \ \forall \sigma \ |I(\sigma)| \leq C(\sigma)[t])\}.$$

Note the condition  $\forall \sigma \ |I(\sigma)| \leq |\sigma| + c$  is not itself absolutely necessary, but it is there to insist on the fact the computable tree whose infinite paths are the elements of  $\mathcal{P}$ , is computably bounded (see Definition 8-7.7).

**Solution 1.14.** Let  $M$  be the program which on a string  $\sigma$  returns the string  $\sigma'U(\sigma)$  where  $\sigma'$  is the binary representation of  $|U(\sigma)|$ . Let  $c$  be such that  $C(\sigma) < C_M(\sigma) + c$  for any string  $\sigma$ . Let  $d$  be such that  $C(\sigma) \leq |\sigma| + d$  for any string  $\sigma$ .

Let  $\sigma$  be a string larger than  $c + d + k + 2^{k+c+d}$ . Let  $\tau \prec \sigma$  be a prefix of length  $c + d + k$ . Let  $n$  be the integer encoded by  $\tau$ . Let  $\rho$  of length  $n$  be such that  $\tau\rho \preceq \sigma$ . Then,  $C_M(\tau\rho) = C(\rho) \leq |\rho| + d = (|\tau\rho| - k - c - d) + d = |\tau\rho| - k - c$ . So  $C(\tau\rho) \leq |\tau\rho| - k$ .

**Solution 1.15.** Fix  $k$  and let us find strings  $\sigma, \tau$  such that  $C(\tau\rho) \geq C(\tau) + C(\rho) + k$ . Let  $c$  be the constant such that  $C(\sigma) \leq |\sigma| + c$  for any string  $\sigma$ . Consider a string  $\sigma$  such that  $C(\sigma) \geq |\sigma|$  and for which we have a prefix  $\tau$  such that  $C(\tau) \leq |\tau| - k - c$ . Let  $\rho$  be such that  $\sigma = \tau\rho$ . We then have  $C(\tau\rho) \geq |\sigma| = |\tau| + |\rho| \geq (C(\tau) + k + c) + C(\rho) - c = C(\tau) + C(\rho) + k$ .

Finally it is clear that  $C(\langle \tau, \rho \rangle) \geq^+ C(\tau\rho)$  for all strings  $\sigma, \tau$ . So also for all  $k$  there are strings  $\tau, \rho$  such that  $C(\langle \tau, \rho \rangle) \geq C(\tau) + C(\rho) + k$ .

**Solution 2.3.** We define the machine  $M$  such that  $M(\sigma) = f(U(\sigma))$ . Thus  $K(f(\tau)) \leq^+ K_M(f(\tau)) \leq^+ K(\tau)$  for any string  $\tau$ .

**Solution 2.6.** We define the machine  $M$  such that  $M(\sigma)$  seeks  $\tau, \rho$  for which  $\tau\rho = \sigma$  and such that  $U(\tau) = |\rho|$ . The machine then returns  $\rho$ . Note that if such

strings  $\tau, \rho$  exist, they are unique because  $U$  is prefix-free. The machine  $M$  is also prefix-free: if  $M(\tau\rho) \downarrow$  that means  $U(\tau) = |\rho|$ . If now  $M(\tau\rho') \downarrow$  for  $\rho' \prec \rho$  or  $\rho \prec \rho'$  that would also mean  $U(\tau) = |\rho'| \neq |\rho|$  which is impossible. We finally obtain the inequality  $K(\sigma) \leq^+ |\sigma| + K(|\sigma|)$  via the machine  $M$ .

Let  $\log_2^{(n)}(\sigma)$  be the  $n$ -th iteration of  $\log_2$  first applied to  $\sigma$ . If we repeat the inequality recursively then for all  $n$  we have  $K(\sigma) \leq^+ |\sigma| + \sum_{k=1}^n \log^{(k)}(|\sigma|) + K(\log^{(n)}(|\sigma|))$  for all  $\sigma$ . For  $n = 1$  this gives us  $K(\sigma) \leq^+ |\sigma| + \log_2(|\sigma|) + K(\log_2(|\sigma|)) \leq^+ |\sigma| + \log_2(|\sigma|) + \log_2(\log_2(|\sigma|)) + 2\log_2(\log_2(\log_2(|\sigma|)))$  which is indeed an improvement.

**Solution 2.7.** Suppose  $K(\sigma) < |\sigma| + c$  for any string  $\sigma$ . For  $n$  fixed we have  $\sum_{|\sigma|=n} 2^{-K(\sigma)} > \sum_{|\sigma|=n} 2^{-|\sigma|-c} = 2^{-c} \sum_{|\sigma|=n} 2^{-|\sigma|} = 2^{-c}$ . For  $n = 2^c + 1$  we then have  $\sum_{|\sigma| \leq n} 2^{-K(\sigma)} = n2^{-c} > 1$ , which is a contradiction.

**Solution 2.11.** Let  $T$  be a computable tree such that  $[T]$  are the paths of  $\mathcal{P}$ . Then, the smallest path of  $[T]$  for the lexicographic order is approachable from the left: It is approximated by  $\sigma_t 0^\infty$  for  $\sigma_t$  the node of length  $t$  furthest on the left of  $T$ .

**Solution 4.7.** This follows simply from Proposition 2.4 which implies  $K(\sigma_0 \oplus \sigma_1) =^+ K(\langle \sigma_0, \sigma_1 \rangle) \leq^+ K(\sigma_0) + K(\sigma_1)$  for any string  $\sigma_0, \sigma_1$ .

## Chapter 17

**Solution 3.4.** We define the function  $f : \mathbb{N} \rightarrow \mathbb{N}$  which on  $\langle n, i, e \rangle$  for  $n \geq 2$  returns  $\langle n, 1 - i, e' \rangle$  where  $e'$  is such that  $W_{e'} = \{f(a) : a \in W_e\}$ , and which on  $\langle 1, i, e \rangle$  returns  $\langle 1, 1 - i, e \rangle$ .

**Solution 3.6.** A careful examination of the proof shows we have uniformity.

**Solution 3.9.** It suffices to see that any dense  $\Pi_2^0(\emptyset')$  class contains a low set, by the finite extension method.

**Solution 3.10.** Consider an open class  $\mathcal{U} = \bigcup_{\tau \in W} [\tau]$  containing all the elements which are not Turing jumps, and strings  $\sigma, \tau$  such that  $X' \succeq \sigma$  for all  $X \succeq \tau$ . Let us show there exists an extension  $\sigma^* \succeq \sigma$  with  $\sigma^* \in W$  and an extension  $\tau^* \succeq \tau$  such that  $X' \succeq \sigma^*$  for all  $X \succeq \tau^*$ .

Let  $\mathcal{V}_m$  be a decreasing sequence of  $\Sigma_1^0$  classes such that  $\bigcap_m \mathcal{V}_m$  is empty and such that  $\mathcal{V}_m \cap [\tau]$  is non-empty for all  $m$ . Let  $a_m > |\sigma|$  be the code of the functional which halts on the input  $a_m$  iff its oracle belongs to  $\mathcal{V}_m$ .

Let  $\sigma_0 = \sigma$  and  $\mathcal{W}_0 = [\tau]$ . Let  $e = |\sigma|$ . Suppose we have defined strings  $\sigma_n \succeq \dots \succeq \sigma_0 = \sigma$  such that  $|\sigma_i| = |\sigma| + i$ , as well as open classes  $\mathcal{W}_i$  for  $i < n$

such that  $\mathcal{W}_{i+1} = \{X : \Phi_{e+i}(X, e+i) \downarrow\}$  if  $\sigma_n(e+i) = 1$  and  $\mathcal{W}_{i+1} \subseteq \{X : \Phi_{e+i}(X, e+i) \uparrow\}$  if  $\sigma_n(e+i) = 0$ . Note that  $\bigcap_{k \leq n} \mathcal{W}_k$  only contains elements  $X$  such that  $\sigma_n \prec X'$ . Suppose also

(1)  $\bigcap_{k \leq n} \mathcal{W}_k \cap \mathcal{V}_m$  not empty for all  $m$ .

Note that (1) holds for  $n = 0$ . Let's build  $\sigma_{n+1}$  and  $\mathcal{W}_{n+1}$ . If  $e+n$  equals  $a_m$  for some  $m$  then  $\sigma_{n+1} = \sigma_n 1$  and  $\mathcal{W}_{n+1} = \mathcal{V}_m = \{X : \Phi_{e+n}(X, e+n) \downarrow\}$ . We easily verify that (1) holds. Otherwise we have two possibilities: if  $\{X : \Phi_{e+n}(X, e+n) \downarrow\} \cap \bigcap_{k \leq n} \mathcal{W}_k \cap \mathcal{V}_m$  is not empty for all  $m$  then  $\sigma_{n+1} = \sigma_n 1$ ,  $\mathcal{W}_{n+1} = \{X : \Phi_{e+n}(X, e+n) \downarrow\}$  and (1) always holds. Otherwise it means  $\bigcap_{k \leq n} \mathcal{W}_k \cap \mathcal{V}_m \subseteq \{X : \Phi_{e+n}(X, e+n) \uparrow\}$  for some  $m$ . We then define  $\sigma_{n+1} = \sigma_n 0$  and  $\mathcal{W}_{n+1} = \bigcap_{k \leq n} \mathcal{W}_k \cap \mathcal{V}_m$  and we still have (1).

Let  $Y = \sigma_1 \prec \sigma_2 \prec \dots$ . Let us show that  $Y$  is not a Turing jump. Suppose that  $Y = X'$  for a set  $X$ . Let  $m \in \mathbb{N}$ . As  $Y(a_m) = 1$ , then  $\Phi_{a_m}(X, a_m) \downarrow$ , so  $X \in \mathcal{V}_m$ . It follows that  $X \in \bigcap_m \mathcal{V}_m$ , but  $\bigcap_m \mathcal{V}_m = \emptyset$ , which leads to a contradiction. So  $Y \in \mathcal{U}$  by hypothesis on  $\mathcal{U}$ . We therefore have a prefix  $\sigma_n \prec Y$  such that  $\sigma_n \in W$ . By (1) the class  $\bigcap_{k \leq n} \mathcal{W}_k \subseteq [\tau]$  is not empty. It is also open and there is therefore an extension  $\tau_n \succeq \tau$  such that  $X' \succeq \sigma_n$  for all  $X \succeq \tau_n$ .

Finally, let  $\bigcap_n \mathcal{U}_n$  be a  $\mathbf{\Pi}_2^0$  class containing all the elements that are not Turing jumps. By using what precedes, one then easily constructs by finite extension elements  $X, Y$  such that  $Y \in \bigcap_n \mathcal{U}_n$  and such that  $Y = X'$ . So  $\bigcap_n \mathcal{U}_n$  also contains elements which are Turing jumps, and the class of all elements which are not jumps is therefore not  $\mathbf{\Pi}_2^0$ .

**Solution 3.11.** Let us introduce some notations. For a string  $\sigma$  we will write  $r(\sigma)$  to denote the ratio of 1 in  $\sigma$  with respect to its length. We denote by  $P_m(\sigma, \varepsilon)$  the predicate " $r(\sigma \upharpoonright_n) < \varepsilon$  for any  $n$  such that  $m < n \leq |\sigma|$ ".

Let  $\bigcup_n \mathcal{F}_n$  be a  $\mathbf{\Sigma}_2^0$  class containing all the elements of positive upper density. Let  $\bigcap_n \mathcal{U}_n$  be the complement of  $\bigcup_n \mathcal{F}_n$ . Let  $\sigma$  be any string and  $\varepsilon$  be a rational. Let us show there must exist an extension  $\tau \succeq \sigma$  and an integer  $n$  such that:

- (1)  $P_{|\sigma|}(\tau, \varepsilon)$
- (2)  $\{X \succeq \tau : \forall m > |\sigma| \ P_{|\sigma|}(X \upharpoonright_m, \varepsilon)\} \subseteq \mathcal{F}_n$

We fix for this proof another rational  $\varepsilon' < \varepsilon$ . Let  $\tau_0 = \sigma$ . Note that (1) holds with  $\tau_0$ . So if (2) holds with  $\tau_0$  and  $n = 0$  then we are done. Otherwise there must exist  $X \in \mathcal{U}_0$  with  $X \succeq \tau_0$  such that  $\forall m > |\sigma| \ P_{|\sigma|}(X \upharpoonright_m, \varepsilon)$ . Let  $\tau_0 \preceq \rho_0 \prec X$  be such that  $[\rho_0] \subseteq \mathcal{U}_0$ . Suppose  $\tau_n, \rho_n$  defined. Then, we find an extension  $\tau_{n+1} \succeq \tau_n$  such that  $P_{|\sigma|}(\tau_{n+1}, \varepsilon)$  and such that  $r(\tau_{n+1}) > \varepsilon'$ . If we have (2) with  $\tau_{n+1}$  and  $n+1$  then we are done. Otherwise there must exist  $X \in \mathcal{U}_{n+1}$  with  $X \succeq \tau_{n+1}$  such that  $\forall m > |\sigma| \ P_{|\sigma|}(X \upharpoonright_m, \varepsilon)$ . Let  $\tau_{n+1} \preceq \rho_{n+1} \prec X$  be such that  $[\rho_{n+1}] \subseteq \mathcal{U}_{n+1}$ . Suppose we are never in the case (2). Then, we construct a sequence  $Y = \tau_0 \prec \tau_1 \prec \dots$  such that  $Y \in \bigcap_n \mathcal{U}_n$  and such that  $r(\tau_n) > \varepsilon'$  for all  $n$ . In particular  $Y$  has a positive upper density and therefore  $Y \in \bigcup_n \mathcal{F}_n$ .

which is a contradiction. So we necessarily reach case (2), which concludes our intermediate proof.

Now let  $\bigcap_k \bigcup_n \mathcal{F}_{k,n}$  be a  $\Pi_3^0$  class containing all elements of positive upper density. Let  $\sigma_0$  be such that  $P_0(\sigma_0, 2^{-1})$  and let  $n_0$  be such that

$$\{X \succeq \sigma_0 : \forall m > 0 \ P_0(X \upharpoonright_m, 2^{-1})\} \subseteq \mathcal{F}_{0,n_0}.$$

Suppose  $\sigma_k$  and  $n_k$  have been defined. Then, we define  $\sigma_{k+1} \succeq \sigma_k$  and  $n_{k+1}$  such that  $P_{|\sigma_k|}(\sigma_{k+1}, 2^{-k-2})$  and such that

$$\{X \succeq \sigma_{k+1} : \forall m > |\sigma_k| \ P_{|\sigma_k|}(X \upharpoonright_m, 2^{-k-2})\} \subseteq \mathcal{F}_{k+1,n_{k+1}}.$$

Let  $Y = \sigma_0 \preceq \sigma_1 \preceq \dots$ . For all  $k$ , we have by construction  $\forall m > |\sigma_k| \ P_{\sigma_k}(Y \upharpoonright_m, 2^{-k-1})$ . Moreover  $Y \in \mathcal{F}_{n_k}$  for all  $k$ . So  $Y \in \bigcap_k \bigcup_n \mathcal{F}_{k,n}$ . It is further clear that the upper density of  $Y$  is 0. So  $\bigcap_k \bigcup_n \mathcal{F}_{k,n}$  also contains upper density 0 elements, and the class of positive upper density elements is therefore not  $\Pi_3^0$ .

**Solution 4.7.** For all  $n$ , the set  $\emptyset^{(n)}$  is  $\Sigma_n^0$ . Consider the class generated by the union of cylinders  $[0^m 1]$  for  $m \in \emptyset^{(n)}$ . It is an open class which admits an effective  $\Sigma_n^0$  description, and whose measure's binary representation equals  $\emptyset^{(n)}$ .

## Chapter 18

**Solution 1.3.** Using the hint, for  $q = 2^n$  and for  $a < 2^n$ , the set of  $X \in q^{\mathbb{N}}$  such that the lower limit of the frequency of occurrence of  $a$  in their prefixes does not converge to  $1/q$  is equal to  $\bigcup_\varepsilon \bigcap_n \bigcup_{m \geq n} C_{\varepsilon,m}^{q,a}$ . According to Hoeffding's inequality, for each  $\varepsilon$  the class  $\bigcap_n \bigcup_{m \geq n} C_{\varepsilon,m}^{q,a}$  is a  $\Pi_2^0$  class of measure 0, and such that the measure of each component  $\bigcup_{m \geq n} C_{\varepsilon,m}^{q,a}$  is bounded by  $2a_n/(1-a_1)$  where  $a_m = e^{-2m\varepsilon^2}$  (see the development done at the beginning of chapter 18).

## Chapter 19

**Solution 3.3.** In one direction it suffices to transform a  $\Pi_2^0(\emptyset^{(n)})$  class into a  $\Pi_{n+2}^0$  class. Each  $\Sigma_1^0(\emptyset^{(n)})$  open class is described by a  $\Sigma_1^0(\emptyset^{(n)})$  set of strings which is also  $\Sigma_{n+1}^0$  according to Theorem 5-5.5. We can therefore uniformly transform each open class in a  $\Sigma_{n+1}^0$  class, the intersection being then  $\Pi_{n+2}^0$ .

For the other direction it suffices to apply Theorem 17-4.4.

**Solution 3.6.** Given a  $\Pi_1^0$  class  $\mathcal{P}$  and a functional  $\Phi_e$ , we define using the fixed point theorem the code  $a_e$  of the  $\emptyset'$ -computable function which does the following: searches for a string  $\sigma$  such that  $\mathcal{P} \cap [\sigma] \neq \emptyset$  and such that  $\Phi_e(\sigma, a_e) \downarrow$ . If such a string  $\sigma$  is found then  $a_e$  codes for the function such that  $\Phi_{a_e}(\emptyset', a_e) =$

$\Phi_e(\sigma, a_e) \downarrow$ . Otherwise  $a_e$  remains partial. Note that the code  $a_e$  itself is uniformly computable without the help of  $\emptyset'$ .

At the same time, without the help of  $\emptyset'$ , we look for the first string  $\sigma$  such that  $\mathcal{P} \cap [\sigma]$  is currently non-empty and such that  $\Phi_e(\sigma, a_e) \downarrow$ , and we enumerate in a  $\Sigma_2^0$  class the code of the  $\Pi_1^0$  class  $\mathcal{P} \cap [\sigma]$ . If we ever notice that  $\mathcal{P} \cap [\sigma] = \emptyset$  we look for another string  $\sigma$  such that  $\mathcal{P} \cap [\sigma]$  is currently not empty and such that  $\Phi_e(\sigma, a_e) \downarrow$ , etc. Between each enumeration of codes of the form  $\mathcal{P} \cap [\sigma]$  we enumerate in our  $\Sigma_2^0$  class a code corresponding to the class  $\mathcal{P}$ , but which we empty completely once the next string  $\sigma$  has been found. There are several possibilities: either  $\Phi(X, a_e) \uparrow$  for all  $X \in \mathcal{P}$  in which case we will come to a point where no string  $\sigma$  will be found, and our  $\Sigma_2^0$  class will contain a code of  $\mathcal{P}$  which will never be emptied. Or  $\Phi(X, a_e) \downarrow$  for an element  $X \in \mathcal{P}$  in which case we will eventually find a string  $\sigma$  such that  $\mathcal{P} \cap [\sigma]$  is non-empty and such that  $\Phi(\sigma, a_e) \downarrow$ . It suffices then to synchronize our research with the computation carried out using the halting problem to ensure that  $\Phi_{a_e}(\emptyset', a_e) = \Phi_e(\sigma, a_e) \downarrow$  for the same string  $\sigma$ . At this point our  $\Sigma_2^0$  class only contains empty class codes, as well as the code of a  $\Pi_1^0$  subclass of  $\mathcal{P}$  such that  $\Phi_e(X, a_n) \downarrow = \Phi_{a_e}(\emptyset', a_e)$  for all  $X$  in this subclass.

We construct in this way for all  $e$ , a  $\Sigma_2^0$  class which intersects each non-empty  $\Pi_1^0$  class (and which therefore contains all computably dominated sets) and such that for all  $X$  in this class, either  $\Phi(X, a_e) \uparrow$ , or  $\Phi(X, a_e) \downarrow = \Phi_{a_e}(a_e)$ . The intersection of these  $\Sigma_2^0$  classes contains all the computably dominated sets, and none of them is of DNC( $\emptyset'$ ) degree.

**Solution 4.9.** Let  $\mathcal{F}$  be a  $\Pi_1^0$  class of positive measure and  $X \in \mathcal{F}$  a set for which there exist  $n$  and  $\varepsilon < 1$  such that for all  $m \geq n$  we have  $\lambda(\mathcal{F} \upharpoonright [X \upharpoonright_m]) < 1 - \varepsilon$ . We define the following test: let  $U_0 = \{X \upharpoonright_n\}$ . Suppose we have defined  $U_k \subseteq 2^{<\mathbb{N}}$  such that a prefix of  $Y$  is in  $U_k$ . For any  $\sigma \in U_k$ , we look for  $t$  such that  $\lambda(\mathcal{F}[t] \upharpoonright [\sigma]) < 1 - \varepsilon$ . If this happens we enumerate in  $U_{k+1}$  a finite set of strings generating the clopen class  $\mathcal{F}[t] \cap [\sigma]$ . By induction a prefix of  $X$  is necessarily in  $U_{k+1}$  and by definition  $\lambda([U_{k+1}]) < (1 - \varepsilon)\lambda([U_k])$ . So  $X$  is captured by  $\bigcap_k [U_k]$  which is a Martin-Löf test.

## Chapter 20

**Solution 1.2.** This is the same proof, except that we create a bounded requests set to “copy” the machine  $U^A$  by a machine with no oracle. Therefore, instead of looking for  $n > 2e$  such that  $n \in W_e[t + 1]$  and such that

$$\sum_{\substack{U(\tau)[t+1] \downarrow = m \\ \text{with } m \geq n}} 2^{-|\tau|} < 2^{-e-1},$$

we look for  $n > 2e$  such that  $n \in W_e[t+1]$  and such that

$$\sum_{\substack{U^{A \upharpoonright m}(\tau)[t+1] \downarrow \\ \text{with } m \geq n}} 2^{-|\tau|} < 2^{-e-1}.$$

If we find such an element  $n$  we enumerate it in  $A$  at step  $t+1$ , then for any  $\tau$  such that  $U^{A \upharpoonright m}(\tau)[t+1] \downarrow = x$  for  $m \geq n$  we enumerate  $(x, |\tau|)$  in  $L$  at step  $t+1$ .

**Solution 1.5.** Let  $A$  be a set. Let  $g : \mathbb{N} \rightarrow 2^{<\mathbb{N}}$  be a partial  $A$ -computable function such that if  $\Phi_e^A(e) \downarrow$ , then  $g(e) \prec A$  and  $\Phi_e^{g(e)}(e) \downarrow$ . In particular,  $K^A(g(e)) \leq^+ K(e) \leq^+ 2\log_2(e)$ . If also  $A$  is low-for-K, then  $K(g(e)) \leq^+ K^A(g(e))$ . Let  $c \in \mathbb{N}$  be such that for all  $e$ ,  $K(g(e)) \leq 2\log_2(e) + c$ . Let  $U$  be a universal machine such that  $K_U = K$ . To decide if  $\Phi_e^A(e) \downarrow$ , it suffices to consider each string  $\sigma$  of length at most  $2\log_2(e) + c$ , to test using  $\emptyset'$  if  $U(\sigma) \downarrow$  and  $U(\sigma) \prec A$  (which is possible because  $A$  is  $\Delta_2^0$ ) and if  $\Phi_e^{U(\sigma)}(e) \downarrow$ . If this is the case for a string  $\sigma$ , then  $\Phi_e^A(e) \downarrow$ . Conversely, if  $\Phi_e^A(e) \downarrow$ , then  $g(e) \downarrow$ , or  $K(g(e)) \leq 2\log_2(e) + c$ , then there exists a string  $\sigma$  of length at most  $2\log_2(e) + c$  such that  $U(\sigma) = g(e)$ . This string  $\sigma$  satisfies the desired property.

**Solution 3.3.** According to the low basis theorem, there exists an MLR set, low and therefore in particular incomplete and  $\Delta_2^0$ . This element being MLR it is obviously not K-trivial.

## Chapter 22

**Solution 2.2.** Let  $F(x)$  be a formula of second-order arithmetic such that  $F(0)$  is true, and such that for all  $x$ ,  $F(x) \rightarrow F(x+1)$ . By the comprehension scheme (9), the set  $X = \{n \in \mathbb{N} : F(x)\}$  exists. We have in particular  $0 \in X$ , and for all  $x$ , if  $x \in X$  then  $x+1 \in X$ . By the induction scheme (10), for all  $x$ ,  $x \in X$ , so for all  $x$ ,  $F(x)$  is true.

**Solution 5.3.** It suffices to see that the formula  $n \in X$  for a set variable  $X$  is  $\Delta_0^0$ .

**Solution 5.7.** Let  $X$  be a non-computable low set. We construct, using Theorem 10-3.31, a 1-generic set  $G \leq_T \emptyset'$  such that  $X \oplus G$  computes  $\emptyset'$ . Note that  $G' \leq_T G \oplus \emptyset'$ , so  $G$  is low.

**Solution 5.8.** We launch in parallel the construction with computable trees of two computably dominated sets  $X_0, X_1$ . We encode in  $X_0$  the information necessary to go from step  $n$  to  $n+1$  in the construction of  $X_1$ , then alternately in  $X_1$  the information necessary to go from step  $n$  to  $n+1$  in the construction of  $X_0$ . The encoding is done in relation to the choice of the path (go left or go right) of  $X_0$  or  $X_1$  in their respective computable trees. The set  $X_0 \oplus X_1$  can

then decode the construction, it will suffice to include an encoding of an arbitrary set  $Z$ .

**Solution 5.10.** Let  $X \in \mathcal{I}$  and  $Y \leq_T X$ . In particular, there is  $n$  such that  $X \in \mathcal{I}_n$ . By closure of  $\mathcal{I}_n$  under the Turing reduction,  $Y \in \mathcal{I}_n$ , hence  $Y \in \mathcal{I}$ . Let  $X, Y \in \mathcal{I}$ . In particular, there are  $n_0, n_1$  such that  $X \in \mathcal{I}_{n_0}$  and  $Y \in \mathcal{I}_{n_1}$ . Let  $n = \max(n_0, n_1)$ . We have  $X, Y \in \mathcal{I}_n$ , so by closure of  $\mathcal{I}_n$  under the effective join,  $X \oplus Y \in \mathcal{I}_n$ , so  $X \oplus Y \in \mathcal{I}$ .

**Solution 5.11.** Let  $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \dots$  be the sequence defined by  $\mathcal{I}_n = \{X \in 2^{\mathbb{N}} : X \leq_T Z_n\}$ . In particular,  $\mathcal{I}_n$  is a Turing ideal, therefore by Exercize 5.10,  $\mathcal{I} = \bigcup_n \mathcal{I}_n$  is a Turing ideal.

**Solution 6.2.** The formula  $F(e) \equiv \Phi_e^X(e) \downarrow$  being  $\Sigma_1^0$  with parameter  $X$ ,  $\text{ACA}_0$  proves the existence of the set  $Y = \{e : \Phi_e^X(e) \downarrow\}$  using the arithmetic comprehension scheme. Conversely, by Proposition 6.1, it suffices to show that  $\text{RCA}_0 + \forall X \exists Y Y = X'$  proves the  $\Sigma_1^0$  comprehension scheme. Let  $F(x)$  be a  $\Sigma_1^0$  formula with parameters  $X_0, \dots, X_{k-1}$ . In particular, with the  $\Delta_1^0$  comprehension scheme, the set  $X_0 \oplus \dots \oplus X_{k-1}$  exists, therefore the set  $Y = (X_0 \oplus \dots \oplus X_{k-1})'$  also exists. Let  $Z = \{x : F(x)\}$ . The set  $Z$  is  $X_0 \oplus \dots \oplus X_{k-1}$ -c.e, therefore  $Y$ -computable. By the  $\Delta_1^0$  comprehension scheme with parameter  $Y$ , the set  $Z$  exists.

**Solution 6.6.** For any  $\sigma \in 2^{\mathbb{N}}$ , let  $I_\sigma \subseteq [0, 1]$  be the interval of real numbers whose binary expansion begins with  $0.\sigma$ . Let

$$\mathcal{C} = \{X \in 2^{\mathbb{N}} : \forall n \exists m > n x_m \in I_{X \upharpoonright n}\}$$

The class  $\mathcal{C}$  is  $\Pi_1^0(\emptyset')$ , not empty, and contains exactly the  $X \in 2^{\mathbb{N}}$  such that  $0.X$  is a point of convergence of a subsequence of  $(x_n)_{n \in \mathbb{N}}$ . By Proposition 8-3.3 relativized to  $\emptyset'$ , there exists a  $\emptyset'$ -computable  $T \subseteq 2^{<\mathbb{N}}$  tree such that  $[T] = \mathcal{C}$ .

Let  $T \subseteq 2^{<\mathbb{N}}$  be an infinite  $\emptyset'$ -computable tree. By Exercize 8-7.13, there exists a computable set  $S = \{\sigma_0, \sigma_1, \dots\} \subseteq 2^{<\mathbb{N}}$  containing exactly one string of each length, and such that  $[T] = [\hat{S}]$ , where  $\hat{S}$  is the prefix closure of  $S$ . The sequence  $(x_n)_{n \in \mathbb{N}}$  defined by  $x_n = 0.\sigma_n$  is such that the points of convergence of its subsequences are exactly of the form  $0.X$  where  $X \in [T]$ .

**Solution 7.4.** The construction is exactly that of Proposition 7.3, but applying the computably dominated basis theorem instead of the cone avoidance basis theorem.

**Solution 7.5.** The construction is exactly that of Proposition 7.3, but applying the low basis theorem instead of the cone avoidance basis theorem.

**Solution 7.7.** For all  $n$ , let  $T_n = \{\sigma \in T : |\sigma| = n\}$  and  $\mathcal{U}_n = \bigcup_{\sigma \in T_n} [\sigma]$ . Note that  $[T] = \bigcap_n \mathcal{U}_n$ . As with all  $n$ ,  $\mathcal{U}_{n+1} \subseteq \mathcal{U}_n$ , then  $\lambda([T]) = \liminf_n \lambda(\mathcal{U}_n)$ .

Since  $T_n$  is prefix-free, we have

$$\lambda(\mathcal{U}_n) = \sum_{\sigma \in T_n} \lambda([\sigma]) = \sum_{\sigma \in T_n} \frac{1}{2^n} = \frac{|\{\sigma \in T : |\sigma| = n\}|}{2^n}$$

## Chapter 23

**Solution 2.2.** It suffices to repeat Example 2.1, this time having two elements  $\infty_1, \infty_2$  such that  $\infty_1 + \infty_2 = \infty_2$ ,  $\infty_2 + \infty_1 = \infty_1$ ,  $\infty_1 \times \infty_2 = \infty_1$ ,  $\infty_2 \times \infty_1 = \infty_2$ ,  $\infty_1 + \infty_1 = \infty_1$ ,  $\infty_2 + \infty_2 = \infty_2$ , as well as  $i + n = n + i = i$  for  $i \in \{\infty_1, \infty_2\}$  and  $n \in \omega$ , as well as  $i \times n = n \times i = i$  for  $i \in \{\infty_1, \infty_2\}$  and  $n \in \omega$  with  $n \neq 0$ , and finally  $i \times 0 = 0 \times i = 0$  for  $i \in \{\infty_1, \infty_2\}$ .

### Solution 2.4.

- (1) The strategy is the same as for showing  $x + y = y + x$ . We first show  $0 \times x = 0$ , then  $(x + 1) \times y = x \times y + y$ . This step uses associativity and commutativity of the addition. We use these two results to show  $x \times y = y \times x$ . The details are left to the readers.
- (2) We have  $x \times (y + 0) = x \times y = x \times y + x \times 0$  by (4) and (6) of  $\mathbf{Q}$ . Suppose  $x \times (y + z) = (x \times y) + (x \times z)$ . Then,  $x \times (y + (z + 1)) = x \times ((y + z) + 1) = (x \times (y + z)) + x = (x \times y) + (x \times z) + x = x \times y + x \times (z + 1)$  by (7) of  $\mathbf{Q}$ , by the induction hypothesis, and by associativity and commutativity of the addition.
- (3) We have  $(x \times y) \times 0 = 0 = x \times (y \times 0)$  by (6) of  $\mathbf{Q}$ . Suppose  $(x \times y) \times z = x \times (y \times z)$ . Then,  $(x \times y) \times (z + 1) = ((x \times y) \times z) + (x \times y) = x \times (y \times z) + (x \times y) = x \times (y \times z + y) = x \times (y \times (z + 1))$  by (7) of  $\mathbf{Q}$ , by the induction hypothesis and distributivity of the multiplication.

**Solution 2.6.** Suppose  $x < y$  and let  $z \neq 0$ . We have  $x + a = y$  for  $a \neq 0$ . So  $(x + a) \times z = y \times z$ . By distributivity of the multiplication, we have  $x \times z + a \times z = y \times z$ . As  $a \neq 0$  and  $z \neq 0$ , we have  $a \times z \neq 0$  by injectivity of the multiplication. So  $x \times z < y \times z$ .

**Solution 2.7.** Let us prove the existence by induction on  $a$ . Let  $b \neq 0$ . We have  $0 = 0 \times b + 0$ . Suppose  $\exists q, r \leq a$  ( $0 \leq r < b \wedge a = qb + r$ ). If  $r < b - 1$ , then  $a + 1 = qb + (r + 1)$ . If  $r = b - 1$ , then  $a + 1 = (q + 1)b$ . In all cases,  $\exists q, r \leq a + 1$  ( $0 \leq r < b \wedge a + 1 = qb + r$ ). By  $\text{I}\Delta_0^0$ , existence is assured for all  $a$ .

Let's show the uniqueness. Suppose  $a = qb + r = q'b + r'$  with  $0 \leq r, r' < b$ . Suppose by contradiction  $q \neq q'$ . We can assume without loss of generality that  $q + 1 \leq q'$ . By Exercize 2.6, we have  $qb + b \leq q'b$ . As  $r < b$ , we have  $qb + r < qb + b \leq q'b \leq q'b + r'$ . So  $qb + r < q'b + r'$ , which contradicts  $qb + r = q'b + r'$ . So  $q = q'$  which implies  $r = r'$ .

**Solution 3.6.** We will show it in the  $\Sigma_n^0$  case. The  $\Pi_n^0$  case is similar. Let  $P_n$  be the ordered induction scheme for the  $\Sigma_n^0$  formulas.

Let's show  $Q \vdash I\Sigma_n^0 \rightarrow P_n$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula such that  $\forall x[(\forall y < x F(y)) \rightarrow F(x)]$ , and let  $z$  be a set. Let us show that  $F(z)$  is true. Let  $G(x) \equiv (x \leq z \rightarrow \forall y < x F(y))$ . Clearly,  $G(0)$  is true because  $\forall y < 0 F(y)$  is true. Also, if  $G(x)$  is true, then  $G(x+1)$  is true. Indeed, if  $x \leq z$ , then  $\forall y < x F(y)$  is true, therefore  $F(x)$  is true, therefore  $\forall y < x+1 F(y)$  is true, therefore  $G(x+1)$  is true. If  $x > z$ , then trivially  $G(x+1)$  is true. It follows that  $G$  satisfies the premises of  $I\Sigma_n^0$ , so  $\forall x G(x)$  is true. In particular,  $G(z)$  is true, therefore  $\forall y < z F(y)$  is true, therefore  $F(z)$  is true.

Let us show  $Q \vdash P_n \rightarrow I\Sigma_n^0$ . Let  $F(x)$  be a  $\Sigma_n^0$  formula such that  $F(0)$  and  $\forall x(F(x) \rightarrow F(x+1))$ . In particular,  $\forall x[(\forall y < x F(y)) \rightarrow F(x)]$ , so by  $P_n$ ,  $\forall x F(x)$  is true.

**Solution 5.4.** (i) The relation  $x \in y$  being  $\Delta_0^0$ , for all  $y$ , the set  $\{x \in \mathbb{N} : x \in y\}$  exists by the  $\Delta_1^0$  comprehension scheme.

(ii) Let  $X$  be a set and  $b \in \mathbb{N}$ . The function  $f : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(0) = 0$  and  $f(n+1) = f(n) + 2^n$  if  $n \in X$  and  $f(n+1) = f(n)$  if  $n \notin X$  is primitive recursive in  $X$ , therefore provable in  $\text{RCA}_0$ . By construction,  $f(b)$  is the canonical code of the set  $X$ .

**Solution 5.7.** The case  $n = 0$  is trivial because  $\text{RCA}_0$  proves both statements.

Suppose  $I\Delta_n^0$ . Let  $F(x, y)$  and  $G(x, y)$  be  $\Pi_{n-1}^0$  and  $\Sigma_{n-1}^0$  formulas, respectively, such that  $\forall x(\exists y F(x, y) \leftrightarrow \forall y G(x, y))$ , and let  $a \in \mathbb{N}$ . Let us show that the set  $A = \{x \in \mathbb{N} : x < a \wedge \exists y F(x, y)\}$  exists. We have

$$\forall x \exists y (F(x, y) \vee \neg G(x, y))$$

By Theorem 3.9,  $I\Delta_n^0 \rightarrow \text{B}\Sigma_n^0$ , and by  $\text{B}\Sigma_n^0$ , there exists a  $b \in \mathbb{N}$  such that

$$\forall x \leq a \exists y \leq b (F(x, y) \vee \neg G(x, y))$$

Then,  $A = \{x \in \mathbb{N} : x < a \wedge \exists y \leq b F(x, y)\}$ , so it is  $\Pi_{n-1}^0$ . By Theorem 3.7,  $\text{B}\Sigma_n^0 \rightarrow I\Sigma_{n-1}^0$ , and by Proposition 5.6,  $I\Sigma_{n-1}^0 \rightarrow \text{BC}\Sigma_{n-1}^0$ , or  $\text{BC}\Sigma_{n-1}^0 \leftrightarrow \text{BC}\Pi_{n-1}^0$ , so the set  $A$  exists.

Suppose  $\text{BC}\Delta_n^0$ . Let  $F(x)$  be a  $\Delta_n^0$  predicate such that  $F(0)$  is true and for all  $n \in \mathbb{N}$ ,  $F(n) \rightarrow F(n+1)$ . Let  $a \in \mathbb{N}$ . Let us show that  $F(a)$  is true. By  $\text{BC}\Delta_n^0$ , the set  $X = \{x \leq a : F(x)\}$  exists. By the  $\Delta_1^0$  comprehension scheme, the set  $Y = X \cup \{y \in \mathbb{N} : y > x\}$  exists. We have  $0 \in Y$  and  $\forall x(x \in Y \rightarrow x+1 \in Y)$ . By the induction axiom (10) which is provable in  $\text{RCA}_0$ ,  $\forall x, x \in Y$ . In particular,  $a \in Y$ , so  $F(a)$  is true.

**Solution 6.9.** This purely syntactic transformation is done by induction on the structure of the formulas, replacing the basic formulas  $x \in X$  by the  $\Sigma_1$  formula  $F\exists(x)$  and  $x \notin X$  by the  $\Sigma_1$  formula  $\neg F\forall(x)$ , where  $F\exists$  and  $F\forall$  define the set  $X \in \mathcal{S}^N$  using parameters in  $\mathcal{M}$ . The existential quantifiers exit the formula with the help of  $\text{B}\Sigma_1^0$ , which is satisfied by  $\mathcal{M}$ .

**Solution 6.16.** It suffices to show the exercise in the case where  $F$  is  $\Delta_0^0$ . We will also impose that  $H(\sigma)$  is monotonic on  $\sigma$ , that is to say that if  $H(\sigma)$  is true, then  $H(\tau)$  is true for all  $\tau \succeq \sigma$ . We proceed by induction on  $F$ .

The following cases are immediate: If  $F(X) \equiv x \in X$ , then  $H(\sigma) \equiv x < |\sigma| \wedge \sigma(x) = 1$ . If  $F(X) \equiv x \notin X$ , then  $H(\sigma) \equiv x < |\sigma| \wedge \sigma(x) = 0$ . If  $F(X)$  is among  $t_1 = t_2$ ,  $t_1 \neq t_2$ , or  $t_1 < t_2$ , then the variable  $X$  does not appear in  $F$  and therefore  $H$  is the same as  $F$ . If  $F(X)$  is among  $F_1(X) \wedge F_2(X)$  or  $F_1(X) \equiv F_2(X)$ , then by induction hypothesis, there are  $\Delta_0^0$  monotonic formulas  $H_1(\sigma)$  and  $H_2(\sigma)$  such that for any set  $G \subseteq M$ ,  $\mathcal{M} \cup \{G\} \models F_i(G) \leftrightarrow \exists k H_i(G \upharpoonright_k)$ . Then,  $H(\sigma)$  is defined by  $H_1(\sigma) \wedge H_2(\sigma)$  or  $H_1(\sigma) \vee H_2(\sigma)$  depending on the case. If  $F(X) \equiv \exists x < t F_1(x, X)$ , then let  $H_1(x, \sigma)$  be the  $\Delta_0^0$  formula corresponding to  $F_1$  by induction hypothesis. The formula  $H(\sigma) \equiv \exists x < t H_1(x, \sigma)$  satisfies the induction.

The non-trivial case is the bounded universal quantification: if  $F(X) \equiv \forall x < t F_1(x, X)$ , then let  $H_1(x, \sigma)$  be the  $\Delta_0^0$  formula corresponding to  $F_1$  by induction hypothesis. Let  $H(\sigma) \equiv \forall x < t H_1(x, \sigma)$ . Let us show that  $H(\sigma)$  satisfies the induction. Let  $G \subseteq M$ . If  $\mathcal{M} \cup \{G\} \models \exists k H(G \upharpoonright_k)$ , then by definition, there exists  $k \in M$  such that  $\mathcal{M} \cup \{G\} \models \forall x < t H_1(x, G \upharpoonright_k)$ , therefore  $\mathcal{M} \cup \{G\} \models \forall x < t \exists k H_1(x, G \upharpoonright_k)$ . By induction hypothesis,  $\mathcal{M} \cup \{G\} \models \forall x < t F_1(x, G)$  and  $\mathcal{M} \cup \{G\} \models F(G)$ . Conversely, suppose that  $\mathcal{M} \cup \{G\} \models F(G)$ . Then, by definition,  $\mathcal{M} \cup \{G\} \models \forall x < t F_1(x, G)$ , therefore by induction hypothesis,  $\mathcal{M} \cup \{G\} \models \forall x < t \exists k F_1(x, G \upharpoonright_k)$ . By  $\text{B}\Sigma_1^0$  and monotonicity of  $F_1$ ,  $\mathcal{M} \cup \{G\} \models \exists k \forall x < t F_1(x, G \upharpoonright_k)$ .

## Chapter 24

**Solution 1.4.** Let's show  $\text{WKL} \leq_\omega \text{KL}$ . The Turing ideals satisfying  $\text{WKL}$  are the Scott ideals, while those satisfying  $\text{KL}$  are the jump ideals. Knowing that for all  $X$ ,  $X'$  is of PA degree relatively to  $X$ , any jump ideal is a Scott ideal.

Let's show  $\text{KL} \not\leq_\omega \text{WKL}$ . Let  $\mathcal{I}$  be a Scott ideal containing only low sets (see Exercize 22-7.5). Then,  $\mathcal{I}$  is not a jump ideal, because  $\emptyset \in \mathcal{I}$  but  $\emptyset' \notin \mathcal{I}$ . Thus,  $\mathcal{I} \models \text{WKL}$  but  $\mathcal{I} \not\models \text{KL}$ .

**Solution 2.2.** Let  $\mathcal{I}$  be a Turing ideal satisfying  $\text{Q}$ . Let us show that  $\mathcal{I} \models \text{P}$ . Let  $X$  be an instance of  $\text{P}$  in  $\mathcal{I}$ . As  $\text{P} \leq_c \text{Q}$ , there is an instance  $\tilde{X}$  of  $\text{Q}$  such that for any  $\text{Q}$ -solution  $\tilde{Y}$  to  $\tilde{X}$ ,  $X \oplus \tilde{Y}$  computes a  $\text{P}$ -solution to  $X$ . Since  $\mathcal{I}$  is closed under the Turing reduction,  $\tilde{X} \in \mathcal{I}$ . As  $\mathcal{I} \models \text{Q}$ , there is a  $\text{Q}$ -solution  $\tilde{Y}$  to  $\tilde{X}$  in  $\mathcal{I}$ . By closure of  $\mathcal{I}$  under the effective join,  $X \oplus \tilde{Y} \in \mathcal{I}$ , and finally by closure of  $\mathcal{I}$  under the Turing reduction,  $\mathcal{I}$  contains a  $\text{P}$ -solution to  $X$ . So  $\mathcal{I} \models \text{P}$ .

**Solution 2.3.** Let  $T$  be an infinite computable finitely-branching tree, all the paths of which are of PA degree relative to  $\emptyset'$  (see Proposition 8-7.4). Let  $\tilde{X} \leq_T T$  be an instance of  $\text{J}$ . In particular,  $\tilde{X}$  is computable, therefore  $\tilde{X}' \equiv_T \emptyset'$ , and  $\tilde{X}'$

is therefore not of PA degree relative to  $\emptyset'$ . As  $\tilde{X}'$  is the unique solution to  $\tilde{X}$  and does not  $T$ -compute an infinite path of  $T$ ,  $\text{KL} \not\leq_c J$ .

**Solution 2.7.** Suppose that  $P$  is computably true. Let  $X$  be an instance of  $P$ . In particular,  $X$  is an instance of  $\text{Id}$ , so the only solution is  $X$ , but  $X \oplus X \geq_T X$  and  $X$  computes a  $P$ -solution to  $X$ , so  $X \oplus X$  computes a solution to  $X$ .

Suppose that  $P \leq_c \text{Id}$ . Let  $X$  be an instance of  $P$ . Let  $\tilde{X}$  be an  $X$ -computable instance of  $\text{Id}$  such that for any  $\text{Id}$ -solution  $\tilde{Y}$  to  $\tilde{X}$ ,  $X \oplus \tilde{Y}$  computes a solution to  $P$ . Since  $\tilde{X}$  is the unique  $\text{Id}$ -solution  $\tilde{Y}$  and  $X \geq_T X \oplus \tilde{X}$ , then  $X$  computes a  $P$ -solution to  $X$ , so  $P$  is computably true.

**Solution 3.3.** Suppose  $P$  is uniformly true using the functional  $\Phi$ . Let  $\Psi$  be the functional defined by  $\Psi(A \oplus B) = \Psi(B)$ . Then,  $P \leq_W \text{Id}$ , as witnessed by the functionals  $\Phi$  and  $\Psi$ .

Conversely, suppose that  $P \leq_W \text{Id}$  using the functionals  $\Phi$  and  $\Psi$ . Let  $\Gamma$  be the functional defined by  $\Gamma(X) = \Psi(X \oplus \Phi(X))$ . For any instance  $X$  of  $P$ ,  $\Gamma(X)$  is a solution to  $X$ .

**Solution 3.7.** Let us show that  $\text{LLPO} \leq_W \text{LPO}$ . Let  $\Phi$  be the functional defined by  $\Phi^X(n) = 1 - X(n)$ . Let  $\Psi$  be the functional such that  $\Psi^{X \oplus i}(0)$  looks for an integer  $n$  such that  $X(n) = 1$  if  $i = 1$ , and returns  $(n + 1) \bmod 2$ . If  $i = 0$ , the functional returns 0 (by convention, we could have chosen 1). If  $X$  is an instance of  $\text{LLPO}$ , then  $\Phi^X$  is an instance of  $\text{LPO}$  whose solution is 1 iff there is  $n$  such that  $X(n) = 1$ . Then, if  $i = 1$ , the instance  $X$  of  $\text{LLPO}$  will have only one solution, and the search for  $\Psi^{X \oplus i}(0)$  halts and returns the correct solution. If  $i = 0$ , both values are solutions to  $X$ , so  $\Psi^{X \oplus i}(0) = 0$  is a valid solution. Thus,  $\Phi$  and  $\Psi$  witness that  $\text{LLPO} \leq_W \text{LPO}$ .

Let us show that  $\text{LPO} \not\leq_W \text{LLPO}$ . Let us reason by the absurd. Suppose that  $\text{LPO} \leq_W \text{LLPO}$  is witnessed by the functionals  $\Phi$  and  $\Psi$ . The sequence  $1^\infty$  is an instance of  $\text{LPO}$  whose solution is 0. Thus,  $\Phi(1^\infty)$  is an instance of  $\text{LLPO}$ . Case 1:  $\Phi(1^\infty) = 0^\infty$ , that is, the values 0 and 1 are both  $\text{LLPO}$ -solutions of  $\Phi(1^\infty)$ . Then,  $\Psi^{1^\infty \oplus 0}(0) \downarrow = \Psi^{1^\infty \oplus 1}(0) \downarrow = 0$ . Let  $n$  be the maximum size of the use of these two computations. Then,  $\Psi^{0^n 1^\infty \oplus 0}(0) \downarrow = \Psi^{0^n 1^\infty \oplus 1}(0) \downarrow = 0$ , which is not the  $\text{LPO}$ -solution to  $0^n 1^\infty$ . Case 2:  $\Phi(1^\infty)(s) = 1$  for an  $s \in \omega$ . Then,  $i = (s + 1) \bmod 2$  is the unique solution to  $\text{LLPO}$ -instance  $\Phi(1^\infty)$  and  $\Psi^{1^\infty \oplus i}(0) \downarrow = 0$ . Let  $n$  be greater than the use of  $\Phi(1^\infty)(s)$  and  $\Psi^{1^\infty \oplus i}(0)$ . In particular,  $\Phi(1^n 0^\infty)(s) = 1$  and  $\Psi^{1^n 0^\infty \oplus i}(0) \downarrow = 0$ , but  $i$  is the  $\text{LLPO}$ -solution to  $\Phi(1^n 0^\infty)$  but 0 is not the  $\text{LPO}$ -solution to  $1^n 0^\infty$ . Contradiction.

**Solution 3.8.** Suppose that  $\text{LLPO}$  is uniformly true, with  $\Phi$  as the Turing functional. In particular,  $0^\infty$  is an instance of  $\text{LLPO}$ , so  $\Phi(0^\infty)(0) \downarrow = i$  for an  $i < 2$ . Let  $n$  be the length of the use of this computation. In particular,  $X_0 = 0^n 10^\infty$  and  $X_1 = 0^n 010^\infty$  are two instances of  $\text{LLPO}$  such that  $\Phi(X_0) = \Phi(X_1) = i$  by the use property, but  $X_0$  and  $X_1$  have opposite solutions. Contradiction.

**Solution 3.10.** Let  $\Phi$  and  $\Psi$  be Turing functionals which witness that  $P \leq_W Q$ . Let  $\Phi_1$  be the functional defined by  $\Phi_1(\bigoplus_n X_n) = \bigoplus_n \Phi(X_n)$  and let  $\Psi_1$  be the functional defined by  $\Psi_1(\bigoplus_n X_n, \bigoplus_n Y_n) = \bigoplus_n \Psi(X_n, Y_n)$ . The functionals  $\Phi_1$  and  $\Psi_1$  witness that  $\widehat{P} \leq_W \widehat{Q}$ .

**Solution 3.12.** Let us show that  $\text{WKL} \leq_W \widehat{\text{LLPO}}$ . For any  $\sigma \in 2^{<\mathbb{N}}$ , let  $\Gamma_\sigma$  be the Turing functional defined by  $\Gamma_\sigma^T(2s+i) = 1$  iff  $s$  is the smallest integer such that  $\forall \tau \in 2^{|\sigma|+s+1} (\sigma i \preceq \tau \rightarrow \tau \notin T)$ . In other words, let  $T \subseteq 2^{<\mathbb{N}}$  be a binary tree, and  $\sigma \in T$  a node whose sub-branch in  $T$  is infinite. If  $i < 2$  is a solution to  $\Gamma_\sigma^T$ , seen as an instance of  $\text{LLPO}$ ,  $\sigma i$  is a node whose sub-branch in  $T$  is still infinite. Let  $\Phi$  be the Turing functional defined by  $\Phi^X = \bigcup_{\sigma \in 2^{<\mathbb{N}}} \Gamma_\sigma^X$ . Let  $\Psi$  be the Turing functional defined for all  $n$  by  $\Psi(T \oplus (\bigoplus_\sigma i_\sigma))(n) = i_{\Psi(T \oplus (\bigoplus_\sigma i_\sigma)) \upharpoonright n}$ . The functionals  $\Phi$  and  $\Psi$  witness that  $\text{WKL} \leq_W \widehat{\text{LLPO}}$ .

Let us show that  $\widehat{\text{LLPO}} \leq_W \text{WKL}$ . Let  $\Phi$  be the Turing functional defined by  $\Phi^{\bigoplus_n X_n} = \{\sigma \in 2^{<\mathbb{N}} : \forall n, s < |\sigma| X_n(2s + \sigma(n)) = 0\}$ . Note that  $\Phi^{\bigoplus_n X_n}$  is an infinite binary tree whose paths are all  $\widehat{\text{LLPO}}$ -solutions of  $\bigoplus_n X_n$ . Let  $\Psi$  be the functional defined by  $\Psi(X \oplus Y) = Y$ . The functional  $\Phi$  and  $\Psi$  witness that  $\widehat{\text{LLPO}} \leq_W \text{WKL}$ .

**Solution 4.4.** Let us show that  $\text{KL} \leq_\omega^2 \text{J}$ . Let  $T$  be an instance of  $\text{KL}$  played by Player 1. In other words,  $T$  is an infinite finitely-branching tree. Player 2 plays  $T$  as an instance of  $\text{J}$ . Player 1 has no choice but to play the unique  $\text{J}$ -solution to  $T$ , namely  $T'$ . Player 2 is playing  $T'$  as an instance  $\text{J}$ . Player 1 then plays  $T''$ , and Player 2  $T''$ -computes a solution to  $T$ , because any infinite finitely-branching tree admits a solution computable in its double jump. Player 2 therefore has a strategy that allows him to win in at most 3 turns.

Let us show that  $\text{J} \not\leq_\omega^1 \text{KL}$ . Player 1 plays an infinite computable finitely-branching tree  $T$  that has no  $\emptyset'$ -computable path. Player 2 must then play a  $\text{J}$ -instance  $X \leq T$ , and Player 1 then plays the  $\text{J}$ -solution  $X'$ . As  $X' \leq_T T'$  and  $T$  do not admit a  $T'$ -computable solution, Player 2 does not win in the second round either.

**Solution 5.6.** Note that for any problem  $Q$ , if  $P \leq_{sW} Q$ , then  $P \leq_W Q$ , because it suffices to ignore the instance  $X$  of  $P$ .

Suppose that  $Q$  is a cylinder for  $\leq_{sW}$  and that  $P \leq_W Q$ , as witnessed by functionals  $\Phi_0, \Psi_0$ . Let  $\Phi_1, \Psi_1$  be the functionals witnessing that  $\text{Id} \times Q \leq_{sW} Q$ . Let  $\Phi_2$  be the functional defined by  $\Phi_2(X) = \Phi_1(X \oplus \Phi_0(X))$  and  $\Psi_2$  the functional defined by  $\Psi_2(Y) = \Psi_0(\Psi_1(Y))$ . The functional  $\Phi_2$  and  $\Psi_2$  witness that  $P \leq_{sW} Q$ .

Now suppose that for any problem  $P$ , if  $P \leq_W Q$  then  $P \leq_{sW} Q$ . In particular, for  $P$  the problem  $\text{Id} \times Q$ , we have  $\text{Id} \times Q \leq_W Q$ , so  $\text{Id} \times Q \leq_{sW} Q$ . In other words,  $Q$  is a cylinder for  $\leq_{sW}$ .

## Chapter 25

**Solution 1.4.** See Theorem 2.5.

**Solution 2.15.** Let  $h : \mathbb{N} \rightarrow \mathbb{N}$  be a modulus of  $\emptyset'$ . We define the  $h$ -computable function  $f : [\mathbb{N}]^2 \rightarrow 2$  by mapping  $x < y$  to the color 1 iff  $h(x) < y$ . Any infinite homogeneous set  $H$  for  $f$  is homogeneous for color 1, and the function which to  $n$  associates the  $n$ -th element of  $H$  dominates  $h$ . Proposition 2.11 allows us to conclude.

**Solution 3.5.** For a  $\Delta_0^0$  formula  $\Phi(x, y, n)$  and a first-order parameter  $k$ , the statement is

$$\exists x < k \forall y \exists n \Phi(x, y, n) \vee \exists z \forall x < k \exists y < z \forall n \Phi(x, y, n).$$

It is clear that if  $\exists y < z \forall n \Phi(x, y, n)$ , then  $\forall a \exists y < z \forall n < a \Phi(x, y, n)$ . Conversely, if  $\forall y < z \exists n \neg \Phi(x, y, n)$ , then by  $\text{B}\Sigma_1^0$  (provable in  $\text{RCA}_0$ )  $\exists a \forall y < z \exists n < a \neg \Phi(x, y, n)$ . Thus, we have  $\exists y < z \forall n \Phi(x, y, n)$  iff  $\forall a \exists y < z \forall n < a \Phi(x, y, n)$ .

**Solution 3.7.** Trivial.

**Solution 3.9.** See the proof of Lemma 14-4.6.

**Solution 3.20.** Let  $A$  and  $C$  be fixed. By Corollary 11-4.21, there is a Scott ideal  $\mathcal{I}$  such that for all  $X \in \mathcal{I}$ ,  $C$  is not  $\Sigma_1^0(X)$ . Suppose (H1) otherwise we already have the desired solution. We are going to construct two infinite sets  $G^0 \subseteq A^0$ ,  $G^1 \subseteq A^1$  satisfying for all  $e_0, e_1 \in \mathbb{N}$  the requirement

$$\mathcal{R}_{e_0, e_1} : W_{e_0}^{G^0} \neq C \vee W_{e_1}^{G^1} \neq C$$

The sets  $G^0$  and  $G^1$  will be built by the forcing of Dzhafarov-Jockusch with the ideal  $\mathcal{I}$ . Let us show the following lemma.

“Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ . There is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  forcing  $\mathcal{R}_{e_0, e_1}$ .”

PROOF OF THE LEMMA. Let  $W = \{x \in \mathbb{N} : c \Vdash x \in W_{e_0}^{G^0} \vee x \in W_{e_1}^{G^1}\}$ . By Lemma 3.15,  $W$  is  $\Sigma_1^0(X)$  with  $X \in \mathcal{I}$ , and  $C$  is not  $\Sigma_1^0(X)$ , so  $W \neq C$ . Two cases arise:

- Case 1: there is some  $x \in W \setminus C$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and  $i < 2$  such that  $(\tau_i, Y) \Vdash x \in W_{e_i}^G$ .
- Case 2: there is some  $x \in C \setminus W$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and  $i < 2$  such that  $(\tau_i, Y) \Vdash x \notin W_{e_i}^G$ .

Let  $F$  be a sufficiently generic filter for the Dzhafarov-Jockusch forcing, and let  $(G^0, G^1) = \dot{F}$ . By Lemma 3.13,  $G^0$  and  $G^1$  are both infinite. By definition

of a forcing condition,  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . By the above lemma,  $(G^0, G^1)$  satisfy  $\mathcal{R}_{e_0, e_1}$  for all  $e_0, e_1 \in \mathbb{N}$ . So either  $C$  is not  $\Sigma_1^0(G^0)$ , or  $C$  is not  $\Sigma_1^0(G^1)$ . Thus, either  $G^0$  or  $G^1$  satisfies the theorem, which concludes the proof.

**Solution 3.21.** Let  $A$  and  $f$  be fixed. By Exercize 22-7.4, there exists a Scott ideal  $\mathcal{I}$  which contains only computably dominated sets. In particular, for all  $X \in \mathcal{I}$ ,  $f$  is  $X$ -hyperimmune. Suppose (H1) otherwise we already have the desired solution. We are going to construct two infinite sets  $G^0 \subseteq A^0$ ,  $G^1 \subseteq A^1$  satisfying for all  $e_0, e_1 \in \mathbb{N}$  the requirement

$$\mathcal{R}_{e_0, e_1} : \Phi_{e_0}^{G^0} \not\geq f \vee \Phi_{e_1}^{G^1} \not\geq f$$

The sets  $G^0$  and  $G^1$  will be built by the forcing of Dzhafarov-Jockusch with the ideal  $\mathcal{I}$ . Let us show the following lemma.

“Let  $c = (\sigma_0, \sigma_1, X)$  be a condition and  $e_0, e_1 \in \mathbb{N}$ . There is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  forcing  $\mathcal{R}_{e_0, e_1}$ .”

PROOF OF THE LEMMA. Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be the function which, for all  $n$ , searches for a finite set  $F \subseteq \mathbb{N}$  such that  $c \restriction F \vdash \Phi_{e_0}^{G^0}(n) \downarrow \in F \vee \Phi_{e_1}^{G^1}(n) \downarrow \in F$ . If such a set is found,  $g(n) = \max F$ , otherwise,  $g(n)$  is not defined. By Lemma 3.15, the function  $g$  is partial  $X$ -computable with  $X \in \mathcal{I}$ . Two cases arise:

- Case 1:  $g$  is total. Then, since  $f$  is  $X$ -hyperimmune, there is  $n \in \mathbb{N}$  such that  $g(n) < f(n)$ . By definition of  $g$ , there exists a set  $F$  satisfying  $\max F < f(n)$  and such that  $c \restriction F \vdash \Phi_{e_0}^{G^0}(n) \downarrow \in F \vee \Phi_{e_1}^{G^1}(n) \downarrow \in F$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and  $i < 2$  such that  $(\tau_i, Y) \Vdash \Phi_{e_i}^{G^i}(n) \downarrow \in F$ .
- Case 2: there is some  $n \in \mathbb{N}$  such that  $g(n)$  is not defined. By definition of  $g$ , for any finite set  $F$ ,  $c \restriction F \not\vdash \Phi_{e_0}^{G^0}(n) \downarrow \in F \vee \Phi_{e_1}^{G^1}(n) \downarrow \in F$ . By compactness,  $c \restriction F \not\vdash \Phi_{e_0}^{G^0}(n) \downarrow \vee \Phi_{e_1}^{G^1}(n) \downarrow$ . By Proposition 3.17, there is an extension  $(\tau_0, \tau_1, Y)$  of  $c$  and  $i < 2$  such that  $(\tau_i, Y) \Vdash \Phi_{e_i}^{G^i}(n) \uparrow$ .

Let  $F$  be a sufficiently generic filter for the Dzhafarov-Jockusch forcing, and let  $(G^0, G^1) = \dot{F}$ . By Lemma 3.13,  $G^0$  and  $G^1$  are both infinite. By definition of a forcing condition,  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . By the above lemma,  $(G^0, G^1)$  satisfy  $\mathcal{R}_{e_0, e_1}$  for all  $e_0, e_1 \in \mathbb{N}$ . So either  $f$  is  $G^0$ -hyperimmune, or  $f$  is  $G^1$ -hyperimmune. Thus, either  $G^0$  or  $G^1$  satisfies the theorem, which concludes the proof.

**Solution 3.22.** This is exactly the construction of Exercize 3.21, forcing the following requirements for all  $e_0, e_1 \in \mathbb{N}$ :

$$\mathcal{R}_{e_0, e_1} : \Phi_{e_0}^{G^0} \not\geq f_0 \vee \Phi_{e_1}^{G^1} \not\geq f_0$$

$$\mathcal{S}_{e_0, e_1} : \Phi_{e_0}^{G^0} \not\geq f_1 \vee \Phi_{e_1}^{G^1} \not\geq f_1$$

$$\mathcal{T}_{e_0, e_1} : \Phi_{e_0}^{G^0} \not\geq f_2 \vee \Phi_{e_1}^{G^1} \not\geq f_2$$

Let  $F$  be a sufficiently generic filter for the Dzhafarov-Jockusch forcing, and let  $(G^0, G^1) = \dot{F}$ . By Lemma 3.13,  $G^0$  and  $G^1$  are both infinite. By definition of a forcing condition,  $G^0 \subseteq A^0$  and  $G^1 \subseteq A^1$ . By the lemma of Exercize 3.21,  $(G^0, G^1)$  satisfies  $\mathcal{R}_{e_0, e_1}$ ,  $\mathcal{S}_{e_0, e_1}$  and  $\mathcal{T}_{e_0, e_1}$  for all  $e_0, e_1 \in \mathbb{N}$ . So for all  $i < 3$ , either  $f_i$  is  $G^0$ -hyperimmune, or  $f_i$  is  $G^1$ -hyperimmune. By the pigeonhole principle, at least two functions  $f_0, f_1$  and  $f_2$  are simultaneously  $G^0$ -hyperimmune or  $G^1$ -hyperimmune. Thus, either  $G^0$  or  $G^1$  satisfies the theorem, which concludes the proof.

**Solution 3.35.** The class  $\mathcal{C} = \{X : \forall e \forall s (W_{e,s} \subseteq X \vee W_{e,s} \subseteq \bar{X}) \rightarrow |W_{e,s}| \leq 2e + 2\}$  is a non-empty  $\Pi_1^0$  class which only contains effectively bi-immune sets. In particular,  $\mathcal{C}$  contains a low member, so  $\Delta_2^0$ .

**Solution 3.36.** This is an immediate generalization of Proposition 3.34. We define a  $\emptyset'$ -computable sequence of strings  $\sigma_0 \prec \sigma_1 \prec \dots$  in  $k^{<\mathbb{N}}$  by alternating the concatenation of long sequences of  $i$  for each  $i < k$  such that the principal function of  $\mathbb{N} \setminus A_i$  diagonalizes against all computable functions.

Suppose  $\sigma_{ke+j}$  defined for  $j < k$ ,  $\sigma_{ke+j+1} = \sigma_{ke+j} \hat{\ }^\ell 0$  if  $\Phi_e(|\sigma_{ke+j}| + 1) \downarrow = \ell$  and  $\sigma_{ke+k} = \sigma_{2e} 0$  otherwise. We easily verify that if  $\Phi_e$  is total, then for  $m = |\sigma_{ke+j}| + 1$ , we have  $p_A(m) \geq |\sigma_{ke+k+1}| - 1 > \Phi_e(m)$ .

**Solution 3.37.** Let  $A_0 \sqcup A_1 \sqcup A_2 = \mathbb{N}$  be a  $\Delta_2^0$  3-partition such that for all  $i < 3$ ,  $\mathbb{N} \setminus A_i$  is hyperimmune. Its existence is ensured by Exercize 3.36. We can see this 3-partition as an instance  $g$  of  $\text{RT}_3^1$  defined by  $g(x) = i$  such that  $x \in A_i$ .

Recall that the *principal function* of an infinite set  $H$  is the function  $p_H : \mathbb{N} \rightarrow \mathbb{N}$  which to  $n$  associates the  $(n+1)$ st element of  $H$ . Recall also that  $H$  is hyperimmune iff  $p_H$  is not bounded by any computable function. For any  $i < 3$ , let  $f_i$  be the principal function of  $\mathbb{N} \setminus A_i$ . Note that  $f_i$  is hyperimmune for all  $i < 3$ , and that for any infinite  $g$ -homogeneous set  $G$  color  $i$ ,  $G$  is a subset of  $A_i$ , so  $G$  is a subset of  $\mathbb{N} \setminus A_j$  for all  $j \neq i$ . It follows that its principal function  $p_G$  dominates  $f_j$  for all  $j \neq i$ , so that none of the  $f_j$  is  $G$ -hyperimmune for  $j \neq i$ .

Let  $h : \mathbb{N} \rightarrow 2$  be an instance of  $\text{RT}_2^1$ . By Exercize 3.22, there exists an infinite  $h$ -homogeneous set  $H$  such that at least 2 among the 3 functions  $f_0, f_1$  and  $f_2$  are  $H$ -hyperimmune. It follows from the previous observation that  $H$  does not compute any infinite  $g$ -homogeneous set.

**Solution 4.2.** Let us first show that any sufficiently generic set for  $\mathbb{P}$  is infinite. Let  $\mathcal{W}_x \subseteq \mathbb{P}$  be the set of Mathias conditions  $(\sigma, X)$  such that  $\sigma(y) = 1$  for a  $y > x$ . Let us show that  $\mathcal{W}_x$  is dense in  $\mathbb{P}$ . Let  $(\sigma, X) \in \mathbb{P}$ . In particular,  $X$  is infinite, so there exists  $y > x$  such that  $y \in X$ . The condition  $(\sigma \cup \{y\}, X \setminus \{0, \dots, y\})$  is an extension of  $(\sigma, X)$  in  $\mathcal{W}_x$ . Thus, for any sufficiently generic  $F$  filter,  $F \cap \mathcal{W}_x \neq \emptyset$ , and therefore  $\dot{F}$  will be infinite.

Let  $Z$  be a computable set and let  $\mathcal{W}_Z \subseteq \mathbb{P}$  be the set of Mathias conditions  $(\sigma, X)$  such that  $X \subseteq Z$  or  $X \subseteq \bar{Z}$ . Let us show that  $\mathcal{W}_Z$  is dense in  $\mathbb{P}$ . Let  $(\sigma, X) \in \mathbb{P}$ . Then,  $X \cap Z$  and  $X \cap \bar{Z}$  are both computable, and at least one of the two is infinite. Then, either  $(\sigma, X \cap Z)$  or  $(\sigma, X \cap \bar{Z})$  is a valid Mathias condition extending  $(\sigma, X)$ , and in any case, that extension will be in  $\mathcal{W}_Z$ . For any sufficiently generic filter  $F$ ,  $F \cap \mathcal{W}_Z \neq \emptyset$ , and therefore  $\dot{F} \subseteq^* Z$  or  $\dot{F} \subseteq^* \bar{Z}$ . Thus,  $\dot{F}$  is a cohesive set for all computable sets.

**Solution 4.11.** Let  $C$  be a non- $\Sigma_1^0$  set. Let  $\mathbb{P}$  be the set of Mathias conditions  $(\sigma, X)$  such that  $X$  is a computable set. Let us show that for any condition  $(\sigma, X)$  and any  $e \in \mathbb{N}$ , there exists an extension  $(\tau, Y)$  forcing  $W_e^G \neq C$ . Let  $(\sigma, X)$  be a condition, and let  $U = \{x \in \mathbb{N} : (\exists \rho \subseteq X) x \in W_e^{\sigma \cup \rho}\}$ . The set  $U$  is  $\Sigma_1^0$ , unlike  $C$ . If there is some  $x \in C \setminus U$ , then by definition of  $U$ ,  $(\sigma, X)$  already forces  $x \notin W_e^G$ , and therefore forces  $W_e^G \neq C$ . Otherwise, there is some  $x \in U \setminus C$ . Let  $\rho \subseteq X$  be such that  $x \in W_e^{\sigma \cup \rho}$ . Then, the condition  $(\sigma \cup \rho, X \setminus \{0, \dots, |\rho|\})$  is an extension of  $(\sigma, X)$  forcing  $x \in W_e^G$ , therefore forcing  $W_e^G \neq C$ . Thus, for any set  $G$  sufficiently generic for  $\mathbb{P}$ ,  $C$  will not be  $\Sigma_1^0(G)$ . Moreover, for any condition  $(\sigma, X)$  and any  $n$ , either  $(\sigma, X \cap R_n)$  or  $(\sigma, X \cap \bar{R}_n)$  is a valid extension, so if  $G$  is a sufficiently generic set for  $\mathbb{P}$ ,  $G$  will be cohesive for  $(R_n)_{n \in \mathbb{N}}$ .

**Solution 4.18.** Let  $(R_n)_{n \in \mathbb{N}}$  be the computable sequence defined by  $R_x = \{y : f(\{x, y\}) = 1\}$ . By Exercise 4.11, there exists a cohesive set  $D$  for  $(R_n)_{n \in \mathbb{N}}$  such that  $C$  is not  $\Sigma_1^0(D)$ . In particular, for all  $x \in D$ ,  $\lim_{y \in D} f(\{x, y\})$  exists. Let  $g : D \rightarrow 2$  be the coloring defined by  $g(x) = \lim_{y \in D} f(\{x, y\})$ . By Proposition 2.2 and Exercise 3.20, there exists an infinite set  $H \subseteq D$  homogeneous for  $g$  such that  $C$  is not  $\Sigma_1^0(H \oplus D)$ . By Fact 4.1,  $H \oplus D$  computes an infinite set  $Y$  homogeneous for  $f$ . In particular,  $C$  is not  $\Sigma_1^0(Y)$ .

## Chapter 27

### Solution 3.7.

- (1) After crushing  $\omega$  Blorks,  $\omega$  Blorks remain. At step  $\omega \times \omega = \omega + \omega + \omega + \dots$  all Blorks that spawned during the crushing of the  $n$ -th block of  $\omega$  Blorks will be wiped out during the crushing of the  $n + 1$ -th block of  $\omega$  Blorks. So there are no more Blorks at step  $\omega \times \omega$ .
- (2) Let  $f(\alpha)$  be the number of Blorks remaining after crushing  $\alpha$  Blorks. We have  $f(\omega) = \omega^2$  and generally  $f(\omega^n) = \omega^{(n+1)}$ . In particular  $f(\omega^\omega) = f(\sup_n \omega^n) = \sup_n f(\omega^n) = \sup_n \omega^{(n+1)} = \omega^\omega$ . So there are  $\omega^\omega$  Blorks after the  $\omega^\omega$  first steps. We then repeat the argument of (1) to see that there are no more Blorks at step  $\omega^\omega \times \omega$ .
- (3) Given the Blorks present at step  $\alpha$ , let  $A_\alpha$  be the subset of Blorks which will end up being struck down at a certain step ( $A_\alpha$  is defined via the comprehension axiom). Now let  $f(\alpha)$  be the smallest step at which all the Blorks

in  $A_\alpha$  are struck down (the replacement axiom implies that the supremum of the stages at which the Borks in  $A_\alpha$  are struck down is indeed an ordinal). We define  $\alpha_0 = \omega$  then for all  $n$  we define  $\alpha_{n+1} = f(\alpha_n)$ . If  $\alpha_n = \alpha_{n+1}$  for a certain  $n$  this means that no more Borks will be struck down from stage  $\alpha_n$  and therefore necessarily that there is no more Bork at this stage. In this case all Borks will be wiped out.

Otherwise  $\alpha_n < \alpha_{n+1}$  for all  $n$ . Let  $\beta = \sup_n \alpha_n$ . We then have  $f(\beta) = f(\sup_n \alpha_n) = \sup_n f(\alpha_n) = \sup_n \alpha_{n+1} = \beta$ . As  $f(\beta) = \beta$  this means that no more Borks will be struck down from step  $\beta$  and therefore necessarily that there is no more Bork at this step. Either way, all Borks will eventually disappear.

**Solution 5.3.** We must show by induction that if  $|a| = \alpha$  for  $a \in \mathcal{O}$  then  $\{|b| : b <_o a\} = \{\beta : \beta < \alpha\}$ .

**Solution 5.5.** The proof is done by induction on  $b$  via the well-founded order  $<_o$ . If  $b = 1$  then by definition  $|a +_o b| = |a| = |a| + |b|$ . If  $b = 2^c$  then  $|a +_o b| = |2^{a+_oc}| = \text{succ}(|a +_o c|) = \text{succ}(|a| + |c|) = |a| + |b|$ . If  $b = 3.5^e$  then  $|a +_o b| = \sup_n |a +_o \Phi_e(n)| = \sup_n (|a| + |\Phi_e(n)|) = |a| + |b|$ .

**Solution 5.12.** The function  $f$  on input  $n$  returns the code of an enumeration  $0 < 1 < 2 < \dots$ . If at some point we notice that  $n \in \emptyset'$  the function chooses a number  $k$  for which nothing has yet been decided and henceforth enumerates axioms such that  $m < k$  for any other integer  $m$ , while keeping the usual order on integers other than  $k$ .

**Solution 5.17.** We have  $\emptyset = |T| = |T|_{\text{KB}}$  for the empty tree  $T$ . Let  $\alpha$  be an ordinal and suppose  $|T| \leq |T|_{\text{KB}}$  for any tree  $T$  such that  $|T| < \alpha$ . Let  $T$  be a tree such that  $|T| = \alpha$ . Then, for each node  $\sigma$  of  $T$  of length 1 we have  $|T \upharpoonright_\sigma| < \alpha$ . By induction hypothesis we therefore have  $|T \upharpoonright_\sigma| \leq |T \upharpoonright_\sigma|_{\text{KB}}$  for each  $\sigma \in T$  of length 1. In the context of  $T$  we have  $|T \upharpoonright_\sigma| = |\sigma|$  and  $|T \upharpoonright_\sigma|_{\text{KB}} \leq |\sigma|_{\text{KB}}$  (because the order  $<_{\text{KB}}$  on  $T \upharpoonright_\sigma$  is embedded in the order  $<_{\text{KB}}$  on  $T$ ). By definition we therefore have  $|T| = \sup\{|\sigma| + 1 : \sigma \in T \text{ of length } 1\} \leq \sup\{|\sigma|_{\text{KB}} + 1 : \sigma \in T \text{ of length } 1\} = |T|_{\text{KB}}$ .

## Chapter 28

**Solution 1.11.** Let us show that each set  $\mathcal{T}_{<\omega(\alpha+k)}$  is  $\Sigma^0_{\alpha+2k}$  and each set  $\mathcal{T}_{\leq\omega(\alpha+k)+p}$  is  $\Pi^0_{\alpha+2k+1}$ . For any  $p \in \mathbb{N}$  the set  $\mathcal{T}_{\leq p}$  is  $\Pi^0_1$  uniformly in  $p$ : it is the set of codes which enumerate trees with heights less than or equal to  $p$ .

Suppose that for  $\alpha = 0$  or limit, for  $k \in \mathbb{N}$  and for all  $p \in \mathbb{N}$  the set  $\mathcal{T}_{\leq\omega(\alpha+k)+p}$  is  $\Pi^0_{\alpha+2k+1}$  uniformly in  $p, k$  and a code of  $\alpha$ . Then,  $\mathcal{T}_{<\omega(\alpha+k+1)} = \bigcup_p \mathcal{T}_{\leq\omega(\alpha+k)+p}$  is  $\Sigma^0_{\alpha+2(k+1)}$  uniformly in  $k+1$  and in a code of  $\alpha$ . Likewise, if  $\mathcal{T}_{\leq\omega(\alpha+k)}$

is  $\Pi_{\alpha+2k+1}^0$  uniformly in  $k$  and a code of  $\alpha$ , for  $\alpha = 0$  or limit and for all  $k \in \mathbb{N}$ , then  $T_{<\omega(\alpha+\omega)} = \bigcup_k T_{\leq\omega(\alpha+k)}$  is  $\Sigma_{\alpha+\omega}^0$ .

Suppose now that for  $\alpha = 0$  or limit, for  $k \in \mathbb{N}$  the set  $T_{<\omega(\alpha+k)}$  is  $\Sigma_{\alpha+2k}^0$  uniformly in  $k$  and in a code of  $\alpha$ . Then, for  $p \in \mathbb{N}$  the set  $T_{\leq\omega(\alpha+k)+p}$  is  $\Pi_{\alpha+2k+1}^0$ , because it is the set of c.e. codes of tree such that for any node  $\sigma$  of length  $p+1$  enumerated, the code of the tree  $T \upharpoonright_\sigma$  belongs to  $T_{<\omega(\alpha+k)}$ , which by induction hypothesis is a  $\Pi_{\alpha+2k+1}^0$  condition uniformly in  $p, k$  and in a code of  $\alpha$ .

We finally show the existence of a function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that:

- For any  $\Sigma_{\alpha+2k}^0$ -code  $e$  of a set  $S_e$  the function  $n \mapsto f(e, n)$  is total and satisfies  $n \in S_e$  implies  $f(e, n) \in \mathcal{T}_{<\omega(\alpha+k)}$  and  $n \notin S_e$  implies  $f(e, n)$  ill-founded.
- For any  $\Pi_{\alpha+2k+1}^0$ -code  $e$  of a set  $S_e$  the function  $n \mapsto f(e, n)$  is total and satisfies  $n \in S_e$  implies  $f(e, n) \in \mathcal{T}_{\leq\omega(\alpha+k)}$  and  $n \notin S_e$  implies  $f(e, n)$  ill-founded.

If  $e$  is the  $\Pi_1^0$ -code of a set  $S_e$  then  $f(e, n)$  returns the code of a c.e. tree remaining empty while  $n \in S_e$  and ill-founded otherwise ( $n \notin S_e$  is a  $\Sigma_1^0$  event). If  $e$  is the  $\Sigma_\alpha^0$ -code of a set  $\bigcup_m F_{a_m}$  where each  $F_{a_m}$  is of code  $a_n$  then  $f(e, n)$  returns the result of the OR function of Exercise 29-5.11 on  $\{f(a_m, n) : m \in \mathbb{N}\}$ . If  $e$  is the  $\Pi_\alpha^0$ -code of a set  $\bigcap_m F_{a_m}$  where each  $F_{a_m}$  has code  $a_n$  then  $f(e, n)$  returns the result of the AND function of lemma 29-5.10 on  $\{f(a_m, n) : m \in \mathbb{N}\}$ . We leave it to the reader to verify that  $f$  has the requested property.

**Solution 4.6.** We must use the relations of Lemma 1.3 to show that the unions and finite intersections of  $\Sigma_\alpha^0$  classes are uniformly  $\Sigma_\alpha^0$ . We can then use this to obtain from a  $\Sigma_\alpha^0$ -code of the class  $\bigcup_n \mathcal{B}_n$ , a  $\Sigma_\alpha^0$ -code of the class  $\bigcup_n (\bigcup_{m \leq n} \mathcal{B}_m)$ .

**Solution 4.7.** From the previous exercise, we can assume without loss of generality that our classes are increasing. We must show by induction that we have a computable function  $f$  such that for all  $\Sigma_\alpha^0$ -code  $e$  of a class  $\mathcal{A}$ , then  $f(e)$  is the  $\Sigma_\alpha^0$ -code of the set  $\{q \in \mathbb{Q} : \lambda(\mathcal{A}) > q\}$ . To show this we will actually need to create function codes  $f_{i,\varepsilon}$  for  $i \in \{-1, 1\}$  and  $\varepsilon \in \mathbb{Q}$  such that for any  $\Sigma_\alpha^0$ -code  $e$  of a class  $\mathcal{A}$ , then  $f_{i,\varepsilon}(e)$  is the  $\Sigma_\alpha^0$ -code of the set  $\{q \in \mathbb{Q} : \lambda(\mathcal{A}) > i \times q + \varepsilon\}$ .

The idea is this. If  $\mathcal{A} = \bigcup_m \mathcal{B}_m$  where each  $\mathcal{B}_m$  is  $\Pi_\beta^0$  for  $\beta < \alpha$ , then  $\{q \in \mathbb{Q} : \lambda(\mathcal{A}) > q\} = \bigcup_m \{q \in \mathbb{Q} : \lambda(\mathcal{B}_m) > q\} = \bigcup_{m,n} \{q \in \mathbb{Q} : \lambda(\mathcal{B}_m) > q + 2^{-n}\} = \bigcup_{m,n} \{q \in \mathbb{Q} : \lambda(2^\mathbb{N} \setminus \mathcal{B}_m) \leq 1 - (q + 2^{-n})\}$ . Note that each set  $\{q \in \mathbb{Q} : \lambda(2^\mathbb{N} \setminus \mathcal{B}_m) \leq 1 - (q + 2^{-n})\}$  is the complement of the set  $\{q \in \mathbb{Q} : \lambda(2^\mathbb{N} \setminus \mathcal{B}_m) > 1 - (q + 2^{-n})\} = \{q \in \mathbb{Q} : \lambda(2^\mathbb{N} \setminus \mathcal{B}_m) > -q + (1 - 2^{-n})\}$  which has by induction a  $\Sigma_\beta^0$ -code via  $f_{-1, 1-2^{-n}}$ . The details are left to the reader.

## Chapter 29

**Solution 2.9.** Let  $X = \{n \in \mathbb{N} : \exists f \mathcal{A}(Y, f, n)\}$  and

$$\begin{aligned} Y &= \{n \in \mathbb{N} : \exists f_1 \mathcal{A}_1(Z, f_1, n)\} \\ \mathbb{N} \setminus Y &= \{n \in \mathbb{N} : \exists f_2 \mathcal{A}_2(Z, f_2, n)\} \end{aligned}$$

where  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are arithmetic. Then,

$$X = \left\{ n \in \mathbb{N} : \exists Y \forall m \left[ \begin{array}{l} (m \in Y \wedge \exists f_1 \mathcal{A}_1(Z, f_1, m)) \vee \\ (m \notin Y \wedge \exists f_2 \mathcal{A}_2(Z, f_2, m)) \end{array} \right] \wedge \exists f \mathcal{A}(Y, f, n) \right\}$$

**Solution 3.9.** For each  $e$ , according to Theorem 28-4.5, the class  $\mathcal{B}_{e,\alpha} = \{X : e \in \mathcal{O}_{=\alpha}^X\}$  is Borel. For  $e$  fixed, if  $\alpha_1 \neq \alpha_2$  then the classes  $\mathcal{B}_{e,\alpha_1}$  and  $\mathcal{B}_{e,\alpha_2}$  are disjoint. So by countable additivity of the measure, for each rational  $q > 0$  and each  $e$  there is only finitely many ordinals  $\alpha$  such that  $\lambda(\mathcal{B}_{e,\alpha}) > q$ . Let  $\alpha_{e,q}$  be the supremum of these ordinals. Let  $\beta = \sup_{e,q} \alpha_{e,q}$  and  $\mathcal{B} = \bigcup_e \mathcal{B}_{e,\beta}$ . We necessarily have  $\beta < \omega_1$  and  $\lambda(\mathcal{B}) = 0$ . Note that  $\mathcal{B}$  is a Borel class.

Let  $\mathcal{A}$  be a  $\Pi_1^1$  class. Then,  $\mathcal{A} = \bigcup_{\alpha < \omega_1} \{Y : e \in \mathcal{O}_{<\alpha}^Y\}$  for some  $e$ . The class  $\bigcup_{\alpha \leq \beta} \{Y : e \in \mathcal{O}_{<\alpha}^Y\}$  is Borel, and the class  $\bigcup_{\beta < \alpha < \omega_1} \{Y : e \in \mathcal{O}_{<\alpha}^Y\}$  is included in  $\mathcal{B}$ , since if  $e \in \mathcal{O}_{<\alpha}^Y$  for  $\alpha > \beta$  then there exists  $e'$  such that  $e' \in \mathcal{O}_{=\beta}^Y$ .

**Solution 5.11.** We define a sequence of trees  $(U_i)_{i \in \mathbb{N}}$ .  $U_0 = T_0$ . If  $U_i$  is defined then  $U_{i+1}$  is obtained by including all the nodes of  $U_i$  length less than or equal to  $i+1$ , and for each of these nodes  $\sigma$ , we add the nodes  $\sigma\tau$  for all  $\tau \in T_{i+1} \vee U_i \upharpoonright \sigma$ . We finally return a code for the tree  $T = \lim_i U_i$ .

**Solution 6.2.** Let  $U$  be a well-founded  $\Sigma_1^1$  tree. Let  $U_\sigma$  be the tree well-founded iff  $\sigma \notin U$ . We compute uniformly in  $X$  the well-founded tree  $T^X = \bigvee_{\sigma \prec X} U_\sigma$ . We finally compute the following c.e. tree  $T$ : for each string  $\sigma \in 2^{<\mathbb{N}}$  such that a node  $m$  of length 1 is enumerated in  $T^\sigma$  we enumerates the node  $\langle m, b(\sigma) \rangle$  of length 1 in  $T$ , using a bijection  $b : 2^{<\mathbb{N}} \rightarrow \mathbb{N}$ . For any node  $\langle \tau, b(\sigma_1) \dots b(\sigma_n) \rangle$  enumerated in  $T$  with  $|\tau| = n$ , if a string  $\tau m$  is enumerated in  $T^{\sigma_{n+1}}$  for a string  $\sigma_{n+1} \succeq \sigma_n$  we enumerate  $\langle \tau m, b(\sigma_1) \dots b(\sigma_n) b(\sigma_{n+1}) \rangle$  in  $T$ .

**Solution 6.5.** It suffices to show that Lemma 6.3 can be obtained uniformly. Let  $\mathcal{A}$  be a  $\Sigma_1^1$  class such that  $\forall X \in \mathcal{A} \exists a \in \mathcal{O} \mathcal{B}(a, X)$  where  $\mathcal{B}$  is a  $\Pi_1^1$  predicate. Let  $e$  be the code of the  $X$ -c.e. tree  $T_e^X$  which is well-founded iff  $X \notin \mathcal{A}$ . Let  $w_1$  be the code of the  $X$ -c.e. tree  $T_{w_1}^X$  which is well-founded iff  $a \in \mathcal{O}^X$ . We can further assume  $|T_{w_1}^X| \geq |a|$  for  $a \in \mathcal{O}^X$ . Let  $w_2$  be the code of the  $X$ -c.e. tree  $T_{w_2}^X$  which is well-founded iff  $\mathcal{B}(a, X)$ . Let  $T^X$  be the c.e. tree  $T_e^X \vee \bigvee_{a \in \mathcal{O}} T_{w_1}^X \wedge T_{w_2}^X$ . Note that  $T^X$  is well-founded for all  $X$ . We finally define using a bijection  $b : 2^{<\mathbb{N}} \rightarrow \mathbb{N}$ :

$$T = \{ \langle \sigma, b(\tau_1) \dots b(\tau_k) \rangle : |\sigma| = k, \tau_1 \preceq \dots \preceq \tau_k \text{ and } \sigma \in T^{\tau_k} \}.$$

We leave it to the reader to verify that the ordinal  $|T|$  satisfies the requested property.

**Solution 7.7.** It suffices to show  $\omega_1^X > \omega_1^{ck}$ . We use Theorem 4.3 to see  $X$  as a set that we enumerate along the computable ordinals. We define the function  $f : \mathbb{N} \rightarrow \omega_1^{ck}$  which to  $n$  associates the smallest ordinal  $\alpha < \omega_1^{ck}$  such that  $X[\alpha] \upharpoonright_n = X \upharpoonright_n$ . Here  $X[\alpha]$  is “the enumeration” of  $X$  up to step  $\alpha$ . We necessarily have  $\sup_n f(n) = \omega_1^{ck}$  because otherwise the enumeration of  $X$  would be completed at a step  $\alpha < \omega_1^{ck}$  and  $X$  would then be  $\Delta_1^1$ . Since  $f$  is total, its image is  $\Delta_1^1(X)$ . According to Spector’s boundedness Lemma 5.3 relativized to  $X$  we therefore have  $\omega_1^{ck} < \omega_1^X$ .

## Chapter 30

**Solution 4.4.** Let  $\mathcal{A} = \{X : \forall n \exists \alpha < \omega_1^{ck} \Phi_e(X, n) \in \mathcal{O}_{<\alpha}^X\}$  for  $e \in \mathbb{N}$ . Suppose  $\lambda(\mathcal{A}) \geq r$  for  $r \in \mathbb{Q}$ . Let  $f$  be the function which to  $n$  associates the smallest  $\alpha < \omega_1^{ck}$  such that  $\lambda(\bigcap_{m \leq n} \{X : \Phi_e(X, m) \in \mathcal{O}_{<\alpha}^X\}) \geq r - 2^{-n}$ . Using Exercise 28-4.7, the function  $f$  is  $\Pi_1^1$ . By the hypothesis  $\lambda(\mathcal{A}) \geq r$ , the function  $f$  is total and therefore has a  $\Delta_1^1$  graph. So its image is  $\Delta_1^1$  and by Lemma 29-5.3 of Spector’s boundedness,  $\sup_n f(n) = \alpha < \omega_1^{ck}$ . We then have  $\lambda(\bigcap_n \{X : \Phi_e(X, n) \in \mathcal{O}_{<\alpha}^X\}) \geq r$ .

As is the case for any rational  $r < \lambda(\mathcal{A})$ , we deduce that the elements  $X \in \mathcal{A}$  for which  $(\Phi_e(X, n))_{n \in \mathbb{N}}$  is co-final in  $\omega_1^{ck}$ , are of zero measure. As it is the case for all  $e$  then the set of elements  $X$  able to compute a sequence  $(a_n)_{n \in \mathbb{N}}$  of elements of  $\mathcal{O}_{<\omega_1^{ck}}^X$  which is co-final in  $\omega_1^{ck}$  is of zero measure. If  $\omega_1^X > \omega_1^{ck}$  then  $X$  must be able to compute such a sequence, so  $\lambda(\{X : \omega_1^X > \omega_1^{ck}\}) = 0$ .

**Solution 4.5.** Let  $\mathcal{S} = \{X : \omega_1^X > \omega_1^{ck}\}$ . According to Corollary 29-3.7, a  $\Pi_1^1$  class  $\mathcal{A}$  is of the form  $\bigcup_{\alpha < \omega_1} \mathcal{A}_\alpha$ . We can moreover suppose that the classes  $\mathcal{A}_\alpha$  are pairwise disjoint. We thus make sure that for all  $\alpha \geq \omega_1^{ck}$  and  $X \in \mathcal{A}_\alpha$ ,  $X \notin \bigcup_{\beta < \omega_1^{ck}} \mathcal{A}_\beta$ , therefore  $\omega_1^X > \omega_1^{ck}$ . In other words,  $\bigcup_{\omega_1^{ck} \leq \alpha < \omega_1} \mathcal{A}_\alpha \subseteq \mathcal{S}$ . The class  $\mathcal{A}$  is of measure zero iff each  $\mathcal{A}_\alpha$  for  $\alpha < \omega_1^{ck}$  is of measure zero. Note that each class  $\mathcal{A}_\alpha$  for  $\alpha < \omega_1^{ck}$  is  $\Delta_1^1$ .

Let  $A \subseteq \mathbb{N}$  be the  $\Pi_1^1$  set of pairs  $\langle a, n \rangle \in \mathbb{N}$  such that  $a \in \mathcal{O}$  and such that  $\lambda(\{X : n \in H_{2^a}^X\}) = 0$ . Note that according to Theorem 28-4.4 any  $\Delta_1^1$  class of measure zero has a code in  $A$ .

The largest  $\Pi_1^1$  class of measure zero is then given by  $\mathcal{S} \cup \bigcup_{(a,n) \in A} \{X : n \in H_{2^a}^X\}$ .



# Bibliography

- [1] *Cabal Seminar 76-77*, volume 689, 1976.
- [2] Uri ABRAHAM et Richard A. SHORE : Initial Segments of the Degrees of Size  $\aleph_1$ . *Israel Journal of Mathematics*, 53(1), 1986.
- [3] Wilhelm ACKERMANN : Zum Hilbertschen Aufbau Der Reellen Zahlen. *Math. Ann.*, 99(1):118–133, 1928.
- [4] J. W. ADDISON : Separation Principles in the Hierarchies of Classical and Effective Descriptive Set Theory. *Fundamenta Mathematicae*, 46: 123–135, 1958. Publisher: Instytut Matematyczny Polskiej Akademii Nauk.
- [5] Klaus AMBOS-SPIES, Carl G. JOCKUSCH, Richard A. SHORE et Robert I. SOARE : An Algebraic Decomposition of the Recursively Enumerable Degrees and the Coincidence of Several Degree Classes with the Promptly Simple Degrees. *Transactions of the American Mathematical Society*, 281(1):109–109, janvier 1984.
- [6] Klaus AMBOS-SPIES, Bjørn KJOS-HANSEN, Steffen LEMPP et Theodore A. SLAMAN : Comparing DNR and WWKL. *Journal of Symbolic Logic*, pages 1089–1104, 2004.
- [7] Uri ANDREWS, Peter GERDES et Joseph S. MILLER : The Degrees of Bi-Hyperhyperimmune Sets. *Ann. Pure Appl. Logic*, 165(3):803–811, 2014.
- [8] Marat Mirzaevich ARSLANOV : Some Generalizations of a Fixed-Point Theorem. *Izvestiya Vysshikh Uchebnykh Zavedenii. Matematika*, (5):9–16, 1981. Publisher: Kazan (Volga region) Federal University.
- [9] Theodore BAKER, John GILL et Robert SOLOVAY : Relativizations of the  $P=NP$  Question. *SIAM Journal on computing*, 4(4):431–442, 1975.
- [10] Jack BARONE et Albert NOVIKOFF : A History of the Axiomatic Formulation of Probability from Borel to Kolmogorov: Part I. *Archive for History of Exact Sciences*, 18(2):123–190, 1978. Publisher: Springer.
- [11] James E. BAUMGARTNER : A Short Proof of Hindman’s Theorem. *Journal of Combinatorial Theory, Series A*, 17(3):384–386, 1974.
- [12] Jean-Pierre BELNA : *Cantor*, volume 20. Belles Lettres, 2000.
- [13] Laurent BIENVENU, Adam R. DAY, Noam GREENBERG, Antonín KUČERA, Joseph S. MILLER, André NIES et Dan TURETSKY : Computing  $K$ -Trivial Sets by Incomplete Random Sets. *The Bulletin of Symbolic Logic*, pages 80–90, 2014. Publisher: JSTOR.

- [14] Laurent BIENVENU et Rod DOWNEY : Kolmogorov Complexity and Solovay Functions. *In 26th International Symposium on Theoretical Aspects of Computer Science*, 2009.
- [15] Laurent BIENVENU, Noam GREENBERG, Antonin KUČERA, André NIES et Dan TURETSKY : K-Triviality, Oberwolfach Randomness, and Differentiability. 2012.
- [16] Laurent BIENVENU, Noam GREENBERG, Antonín KUČERA, Joseph S. MILLER, André NIES et Dan TURETSKY : Joining Non-Low c.e. Sets with Diagonally Non-Computable Functions. *J. Logic Comput.*, 23(6):1183–1194, 2013.
- [17] Laurent BIENVENU, Noam GREENBERG, Antonín KUČERA, André NIES et Dan TURETSKY : Coherent Randomness Tests and Computing the K-Trivial Sets. *J. Eur. Math. Soc. (JEMS)*, 18(4):773–812, 2016.
- [18] Laurent BIENVENU, Rupert HÖLZL, Joseph S. MILLER et André NIES : The Denjoy Alternative for Computable Functions. *In 29th International Symposium on Theoretical Aspects of Computer Science*, page 543, 2012.
- [19] Laurent BIENVENU, Rupert HÖLZL, Joseph S. MILLER et André NIES : Denjoy, Demuth and Density. *Journal of Mathematical Logic*, 14(01):1450004, 2014. Publisher: World Scientific.
- [20] Laurent BIENVENU, Wolfgang MERKLE et Andre NIES : Solovay Functions and K-Triviality. *In 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, 2011.
- [21] Andreas R. BLASS, Jeffrey L. HIRST et Stephen G. SIMPSON : Logical Analysis of some Theorems of Combinatorics and Topological Dynamics. *Logic and Combinatorics*, S. Simpson, ed., *Contemporary Math*, 69:125–156, 1987.
- [22] Émile BOREL : Les «paradoxes» de la théorie des ensembles. *In Annales Scientifiques De L'École Normale Supérieure*, volume 25, pages 443–448, 1908.
- [23] M Émile BOREL : Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 27(1):247–271, 1909. Publisher: Springer.
- [24] V. BRATTKA, G. GHERARDI et A. PAULY : Weihrauch Complexity in Computable Analysis. *ArXiv*, abs/1707.03202, 2017.
- [25] Vasco BRATTKA et Tahina RAKOTONIAINA : On the Uniform Computational Content of Ramsey's Theorem. *J. Symb. Log.*, 82(4):1278–1316, 2017.
- [26] Cristian S. CALUDE, Peter H. HERTLING, Bakhadyr KHOUSSAINOV et Yongge WANG : Recursively Enumerable Reals and Chaitin's Numbers. *In Annual Symposium on Theoretical Aspects of Computer Science*, pages 596–606. Springer, 1998.

- [27] Georg CANTOR : Fondements d'une théorie générale des ensembles. *Acta Mathematica*, 2(1):381–408, 1883. Publisher: Springer.
- [28] Gregory J. CHAITIN : A Theory of Program Size Formally Identical to Information Theory. *Journal of the ACM (JACM)*, 22(3):329–340, 1975. Publisher: ACM New York, NY, USA.
- [29] Gregory J. CHAITIN : Information-Theoretic Characterizations of Recursive Infinite Strings. *Theoretical Computer Science*, 2(1):45–48, 1976. Publisher: Elsevier.
- [30] Gregory J. CHAITIN : Incompleteness Theorems for Random Reals. *Advances in Applied Mathematics*, 8(2):119–146, 1987. Publisher: Elsevier.
- [31] David G. CHAMPERNOWNE : The Construction of Decimals Normal in the Scale of Ten. *Journal of the London Mathematical Society*, 1(4):254–260, 1933.
- [32] Peter A. CHOLAK, Carl G. JOCKUSCH et Theodore A. SLAMAN : On the Strength of Ramsey's Theorem for Pairs. *The Journal of Symbolic Logic*, 66(1):1–55, 2001. Publisher: Cambridge University Press.
- [33] Peter A. CHOLAK et Ludovic PATEY : Thin Set Theorems and Cone Avoidance. *Transactions of the AMS*. To appear., 2019.
- [34] Chi Tat CHONG et Liang YU : *Recursion Theory: Computational Aspects of Definability*, volume 8. Walter de Gruyter GmbH & Co KG, 2015.
- [35] Alonzo CHURCH et Stephen C. KLEENE : Formal Definitions in the Theory of Ordinal Numbers. *Fundamenta Mathematicae*, 28(1):11–21, 1937.
- [36] Paul J. COHEN : The Independence of the Continuum Hypothesis, II. *Proceedings of the National Academy of Sciences of the United States of America*, 51(1):105, 1964. Publisher: National Academy of Sciences.
- [37] Joshua A COLE et Stephen G SIMPSON : Mass Problems and Hyperarithmeticity. *Journal of Mathematical Logic*, 7(02):125–143, 2007.
- [38] W Wistar COMFORT : Ultrafilters: Some Old and some New Results. *Bulletin of the American Mathematical Society*, 83(4):417–455, 1977.
- [39] Barry S. COOPER : Degrees of Unsolvability Complementary between Recursively Enumerable Degrees. *Annals of Mathematical Logic*, 4(1): 31–73, 1972.
- [40] Barry S. COOPER : *Computability Theory*. CRC Press, 2003.
- [41] René CORI et Daniel LASCAR : *Logique mathématique, Volume II*. Masson, 1993.
- [42] Adam R. DAY et Joseph S. MILLER : Density, Forcing, and the Covering Problem. *Mathematical Research Letters*, 22(3):719–727, 2015. Publisher: International Press of Boston.

- [43] Patrick DEHORNOY : La théorie des ensembles. *Calvage et Mounet, Paris*, 2017.
- [44] Osvald DEMUTH et Antonín KUČERA : Remarks on 1-Genericity, Semigenericity and Related Concepts. *Commentationes Mathematicae Universitatis Carolinae*, 28(1):85–94, 1987. Publisher: Charles University in Prague, Faculty of Mathematics and Physics.
- [45] R. G. DOWNEY : Maximal theories. *Ann. Pure Appl. Logic*, 33(3): 245–282, 1987.
- [46] Rod DOWNEY, Noam GREENBERG, Matthew HARRISON-TRAINOR, Ludovic PATEY et Dan TURETSKY : Relationships between Computability-Theoretic Properties of Problems, 2019.
- [47] Rod DOWNEY, Denis HIRSCHFELDT, Steffen LEMPP et Reed SOLOMON : A  $\Delta_2^0$  Set with no Infinite Low Subset in either it or its Complement. *Journal of Symbolic Logic*, 66(3):1371–1381, 2001.
- [48] Rod DOWNEY, André NIES, Rebecca WEBER et Liang YU : Lowness and  $\Pi_2^0$  Nullsets. *Journal of Symbolic Logic*, 71(3):1044–1052, 2006.
- [49] Rodney G. DOWNEY et Denis R. HIRSCHFELDT : *Algorithmic Randomness and Complexity*. Springer Science & Business Media, 2010.
- [50] Arnaud DURAND et Paul ROZIÈRE : Calculabilité et incomplétude - Notes de cours.
- [51] Damir D. DZHAFAROV : Cohesive Avoidance and Strong Reductions. *Proceedings of the American Mathematical Society*, 143(2):869–876, 2014.
- [52] Damir D. DZHAFAROV, Denis R. HIRSCHFELDT et Sarah C. REITZES : Reduction Games, Provability, and Compactness, 2020.
- [53] Damir D DZHAFAROV et Carl G. JOCKUSCH : Ramsey’s Theorem and Cone Avoidance. *The Journal of Symbolic Logic*, 74(2):557–578, 2009. Publisher: Cambridge University Press.
- [54] Calvin C. ELGOT et Abraham ROBINSON : Random-Access Stored-Program Machines, an Approach to Programming Languages. In *Selected Papers*, pages 17–51. Springer, 1982.
- [55] Herbert ENDERTON et Hilary PUTNAM : A Note on the Hyperarithmetical Hierarchy. *The Journal of Symbolic Logic*, 35(3):429–430, 1970. Publisher: Cambridge University Press.
- [56] Andrei Petrovich ERSHOV : On Operator Algorithms. In *Doklady Akademii Nauk*, volume 122, pages 967–970. Russian Academy of Sciences, 1958.
- [57] Solomon FEFERMAN et Clifford SPECTOR : Incompleteness along Paths in Progressions of Theories 1. *The Journal of Symbolic Logic*, 27(4):383–390, 1962. Publisher: Cambridge University Press.
- [58] Michael R. FELLOWS : *Computer Science and Mathematics in the Elementary Schools*. Citeseer, 1991.

- [59] Lance FORTNOW et Michael SIPSER : Are there Interactive Protocols for Co-Np Languages? *Information Processing Letters*, 28(5):249–251, 1988.
- [60] Johanna FRANKLIN et Keng Meng NG : Difference Randomness. *Proceedings of the American Mathematical Society*, 139(1):345–360, 2011.
- [61] Richard FRIEDBERG : A Criterion for Completeness of Degrees of Unsolvability. *The Journal of Symbolic Logic*, 22(2):159–160, 1957.
- [62] Richard M. FRIEDBERG : Two Recursively Enumerable Sets of Incomparable Degrees of Unsolvability (Solution of Post's Problem, 1944). *Proceedings of the National Academy of Sciences of the United States of America*, 43(2):236–238, 1957. Publisher: National Academy of Sciences.
- [63] Richard M. FRIEDBERG : Three Theorems on Recursive Enumeration. I. Decomposition. II. Maximal Set. III. Enumeration without Duplication. *J. Symbolic Logic*, 23:309–316, 1958.
- [64] Harvey FRIEDMAN : Some Systems of Second Order Arithmetic and their Use. In *Proceedings of the International Congress of Mathematicians (Vancouver, BC, 1974)*, volume 1, pages 235–242. Citeseer, 1975.
- [65] Harvey FRIEDMAN : Uniformly Defined Descending Sequences of Degrees. *The Journal of Symbolic Logic*, 41(2):363–367, 1976. Publisher: JSTOR.
- [66] Harvey M. FRIEDMAN : Higher Set Theory and Mathematical Practice. *Mathematical Logic in the 20th Century*, 2(3):49, 2003.
- [67] Harvey Martin FRIEDMAN : *Subsystems of Set Theory and Analysis*. PhD Thesis, Massachusetts Institute of Technology, 1967.
- [68] Peter GÁCS : On the Symmetry of Algorithmic Information. In *Doklady Akademii Nauk*, volume 218, pages 1265–1267. Russian Academy of Sciences, 1974. Issue: 6.
- [69] Péter GÁCS : Every Sequence is Reducible to a Random One. *Information and Control*, 70(2/3):186–192, 1986.
- [70] GALILÉE : *Discours et démonstrations mathématiques concernant deux sciences nouvelles*. Masson Éditeur, 1972. Introduction, traduction, notes et index de Maurice Clavelin.
- [71] Robin O. GANDY : On a Problem of Kleene's. *Bulletin of the American Mathematical Society*, 66(6):501–502, 1960.
- [72] Guido GHERARDI et Alberto MARCONE : How Incomputable is the separable Hahn-Banach Theorem? *Notre Dame J. Form. Log.*, 50(4):393–425 (2010), 2009.
- [73] Kurt GÖDEL : The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis. *Proceedings of the National*

- Academy of Sciences of the United States of America*, 24(12):556, 1938. Publisher: National Academy of Sciences.
- [74] Herman Heine GOLDSTINE et John VON NEUMANN, John andXXX??? Von Neumann : Planning and Coding of Problems for an Electronic Computing Instrument. 1947.
  - [75] Noam GREENBERG et Joseph S. MILLER : Lowness for Kurtz Randomness. *The Journal of Symbolic Logic*, 74(2):665–678, 2009. Publisher: Cambridge University Press.
  - [76] Marcia J. GROSZEK et Theodore A. SLAMAN : Independence Results on the Global Structure of the Turing Degrees. *Transactions of the American Mathematical Society*, 277(2):579–588, 1983.
  - [77] Marcia J. GROSZEK et Theodore A. SLAMAN :  $\Pi_1^0$  Classes and Minimal Degrees. *Annals of Pure and Applied Logic*, 87(2):117–144, 1997.
  - [78] Marcia J GROSZEK et Theodore A SLAMAN : Moduli of Computation (Talk). *Buenos Aires, Argentina*, 2007.
  - [79] David GUASPARI : A Note on the Kondo-Addison Theorem. *The Journal of Symbolic Logic*, 39(3):567–570, 1974. Publisher: JSTOR.
  - [80] Petr HÁJEK et Pavel PUDLÁK : *Metamathematics of First-Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1998.
  - [81] Joel David HAMKINS et Andy LEWIS-PYE : Infinite Time Turing Machines. *Journal of Symbolic Logic*, pages 567–604, 2000. Publisher: JSTOR.
  - [82] Leo HARRINGTON et Saharon SHELAH : The Undecidability of the Recursively Enumerable Degrees. *Bulletin of the American Mathematical Society*, 6, 1982.
  - [83] Joseph HARRISON : Recursive Pseudo-Well-Orderings. *Transactions of the American Mathematical Society*, 131(2):526–543, 1968. Publisher: JSTOR.
  - [84] Leon HENKIN : The Completeness of the First-Order Functional Calculus. *The Journal of Symbolic Logic*, 14(3):159–166, 1949.
  - [85] Neil HINDMAN : Finite Sums From Sequences within Cells of a Partition of  $N$ . *J. Combinatorial Theory Ser. A*, 17:1–11, 1974.
  - [86] Denis R HIRSCHFELDT : Slicing the Truth. *Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore*, 28, 2015. Publisher: World Scientific Publishing.
  - [87] Denis R. HIRSCHFELDT et Carl G. JOCKUSCH : On Notions of Computability-Theoretic Reduction between  $\Pi_2^1$  Principles. *J. Math. Log.*, 16(1):1650002, 59, 2016.
  - [88] Denis R. HIRSCHFELDT, Carl G. JOCKUSCH, Bjørn KJOS-HANSEN, Steffen LEMPP et Theodore A. SLAMAN : The Strength of some Combinatorial Principles Related to Ramsey’s Theorem for Pairs. *Computational Prospects of Infinity, Part II: Presented Talks, World Scientific Press, Singapore*, pages 143–161, 2008.

- [89] Denis R HIRSCHFELDT, André NIES et Frank STEPHAN : Using Random Sets as Oracles. *Journal of the London Mathematical Society*, 75(3):610–622, 2007. Publisher: Oxford University Press.
- [90] Jeffry L. HIRST : *Combinatorics in Subsystems of Second Order Arithmetic*. Thèse de doctorat, Pennsylvania State University, août 1987.
- [91] Greg HJORTH et André NIES : Randomness via Effective Descriptive Set Theory. *Journal of the London Mathematical Society*, 75(2):495–508, 2007.
- [92] Wassily Hoeffding : Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. Publisher: Taylor & Francis Group.
- [93] Mathieu HOYRUP et Cristóbal ROJAS : Applications of Effective Probability Theory to Martin-Löf Randomness. In *International Colloquium on Automata, Languages, and Programming*, pages 549–561. Springer, 2009.
- [94] Shamil ISHMUKHAMETOV : Weak Recursive Degrees and a Problem of Spector. In *Recursion Theory and Complexity (Kazan, 1997)*, volume 2 de *De Gruyter Ser. Log. Appl.*, pages 81–87. de Gruyter, Berlin, 1999.
- [95] Carl JOCKUSCH et Robert SOARE : Degrees of Members of  $\Pi_1^0$  Classes. *Pacific Journal of Mathematics*, 40:605–616, 1972.
- [96] Carl G. JOCKUSCH : Relationships between Reducibilities. *Transactions of the American Mathematical Society*, 142:229–237, 1969.
- [97] Carl G JOCKUSCH : Degrees in which the Recursive Sets are Uniformly Recursive. *Canadian Journal of Mathematics*, 24(6):1092–1099, 1972. Publisher: Cambridge University Press.
- [98] Carl G. JOCKUSCH : Ramsey’s Theorem and Recursion Theory. *The Journal of Symbolic Logic*, 37(2):268–280, 1972. Publisher: Cambridge University Press.
- [99] Carl G. JOCKUSCH : Degrees of Generic Sets. *Recursion Theory: Its Generalizations and Applications*, pages 110–139, 1980. Publisher: Cambridge Univ. Press Cambridge.
- [100] Carl G. JOCKUSCH, Manuel LERMAN, Robert I. SOARE et Robert M. SOLOVAY : Recursively Enumerable Sets modulo Iterated Jumps and Extensions of Arslanov’s Completeness Criterion. *The Journal of Symbolic Logic*, 54(4):1288–1323, 1989. Publisher: Cambridge University Press.
- [101] Carl G. JOCKUSCH et Richard A. SHORE : Pseudo-Jump Operators. II: Transfinite Iterations, Hierarchies and Minimal Covers. *The Journal of Symbolic Logic*, 49(4):1205–1236, 1984. Publisher: JSTOR.

- [102] Carl G. JOCKUSCH et Frank STEPHAN : A Cohesive Set which is not High. *Mathematical Logic Quarterly*, 39(1):515–530, 1993. Publisher: Wiley Online Library.
- [103] Carl G. JOCKUSCH, Jr. et Robert I. SOARE :  $\Pi_1^0$  Classes and Degrees of Theories. *Trans. Amer. Math. Soc.*, 173:33–56, 1972.
- [104] Carl G JOCKUSCH JR : Degrees of Functions with no Fixed Points. *In Studies in Logic and the Foundations of Mathematics*, volume 126, pages 191–201. Elsevier, 1989.
- [105] Akihiro KANAMORI : Mathias and Set Theory. *Mathematical Logic Quarterly*, 62(3):278–294, 2016.
- [106] Loren KANTOR, Jean-Michel et Graham : *Au nom de l'infini. Une histoire vraie de mysticisme religieux et de création mathématique*. Belin, 2010.
- [107] Steven M. KAUTZ : *Degrees of Random Sets*. PhD Thesis, Citeseer, 1991.
- [108] Richard KAYE : *Models of Peano Arithmetic*. 1991.
- [109] Alexander S. KECHRIS : The Theory of Countable Analytical Sets. *Transactions of the American Mathematical Society*, 202:259–297, 1975.
- [110] Mushfeq KHAN et Joseph S. MILLER : Forcing with Bushy Trees. *Bulletin of Symbolic Logic*, 23(2):160–180, 2017. Publisher: Cambridge University Press.
- [111] Bjørn KJOS-HANSSEN, Wolfgang MERKLE et Frank STEPHAN : Kolmogorov Complexity and the Recursion Theorem. *Transactions of the American Mathematical Society*, 363(10):5465–5480, 2011.
- [112] Bjørn KJOS-HANSSEN, Wolfgang MERKLE et Frank STEPHAN : Kolmogorov Complexity and the Recursion Theorem. *Transactions of the American Mathematical Society*, 363(10):5465–5480, 2011.
- [113] Stephen C. KLEENE : On Notation for Ordinal Numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938. Publisher: JSTOR.
- [114] Stephen C. KLEENE : Arithmetical Predicates and Function Quantifiers. *Transactions of the American Mathematical Society*, 79(2):312–340, 1955. Publisher: JSTOR.
- [115] Stephen C. KLEENE : Hierarchies of Number-Theoretic Predicates. *Bulletin of the American Mathematical Society*, 61(3):193–213, 1955.
- [116] Stephen C. KLEENE : On the Forms of the Predicates in the Theory of Constructive Ordinals (Second Paper). *American Journal of Mathematics*, 77(3):405–428, 1955. Publisher: JSTOR.
- [117] Stephen C. KLEENE et Emil L. POST : The Upper Semi-Lattice of Degrees of Recursive Unsolvability. *Annals of Mathematics*, pages 379–407, 1954. Publisher: JSTOR.
- [118] Andrej KOLMOGOROFF : Zur Deutung der Intuitionistischen Logik. *Mathematische Zeitschrift*, 35(1):58–65, 1932.

- [119] Andrei KOLMOGOROV : Logical Basis for Information Theory and Probability Theory. *IEEE Transactions on Information Theory*, 14(5):662–664, 1968.
- [120] Andrei N. KOLMOGOROV : On Tables of Random Numbers. *Sankhya The Indian Journal of Statistics, Series A*, pages 369–376, 1963. Publisher: JSTOR.
- [121] Andrei N. KOLMOGOROV : Three Approaches to the Quantitative Definition Ofinformation'. *Problems of Information Transmission*, 1(1):1–7, 1965.
- [122] Motokiti KONDÔ : Sur l'uniformisation des complémentaires analytiques et les ensembles projectifs de la seconde classe. In *Japanese Journal of Mathematics: Transactions and Abstracts*, volume 15, pages 197–230. The Mathematical Society of Japan, 1939. Issue: 0.
- [123] G. KREISEL : Analysis of the Cantor-Bendixson Theorem by Means of the Analytic Hierarchy. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astr. Phys.*, 7:621–626. (unbound insert), 1959.
- [124] Masahiro KUMABE et Andrew LEWIS-PYE : A Fixed-Point-Free Minimal Degree. *Journal of the London Mathematical Society*, 80(3), 2009.
- [125] Stuart A. KURTZ : *Randomness and Genericity in the Degrees of Unsolvability*. PhD Thesis, 1982.
- [126] Antonín KUČERA et Theodore SLAMAN : Randomness and Recursive Enumerability. *SIAM Journal on Computing*, 31(1):199–211, 2001. Publisher: SIAM.
- [127] Antonín KUČERA : Measure,  $\Pi_1^0$ -Classes and Complete Extensions of PA. In *Recursion Theory Week*, pages 245–259. Springer, 1985.
- [128] A. H. LACHLAN : Lower Bounds for Pairs of Recursively Enumerable Degrees. *Proceedings of the London Mathematical Society*, s3-16(1): 537–569, 1966.
- [129] Alistair H. LACHLAN : Complete Recursively Enumerable Sets. *Proceedings of the American Mathematical Society*, 19(1):99–102, 1968.
- [130] Alistair H. LACHLAN : Embedding Nondistributive Lattices in the Recursively Enumerable Degrees. In *Conference in Mathematical Logic—London 70*, pages 149–177. Springer, 1972.
- [131] Alistair H. LACHLAN : Uniform Enumeration Operations. *Journal of Symbolic Logic*, pages 401–409, 1975.
- [132] Alistair H. LACHLAN et Robert LEBEUF : Countable Initial Segments of the Degrees of Unsolvability. *The Journal of Symbolic Logic*, 41(2): 289–300, 1976.
- [133] Alistair H. LACHLAN et Robert I. SOARE : Not Every Finite Lattice is Embeddable in the Recursively Enumerable Degrees. *Advances in Mathematics*, 37(1):74–82, 1980.

- [134] Joachim LAMBEK : How to Program an Infinite Abacus. *Canadian Mathematical Bulletin*, 4(3):295–302, 1961.
- [135] Henri LEBESGUE : Sur une généralisation de l'intégrale définie. *CR Acad. Sci. Paris*, 132:1025–1028, 1901.
- [136] Steffen LEMPP, André NIES et Theodore A. SLAMAN : The  $\Pi^0_3$ -Theory of the Computably Enumerable Turing Degrees is Undecidable. *Transactions of the American Mathematical Society*, pages 2719–2736, 1998.
- [137] Manuel LERMAN : *Degrees of Unsolvability*, volume 11. Cambridge University Press, 2017.
- [138] Leonid A. LEVIN : *Some Theorems on the Algorithmic Approach to Probability Theory and Information Theory*. Dissertation in Mathematics, Université de Moscou, 1971.
- [139] Leonid A. LEVIN : On the Notion of a Random Sequence. *In Soviet. Math. Dokl.*, volume 14, pages 1413–1416, 1973. Issue: 5.
- [140] Leonid A. LEVIN : The Concept of a Random Sequence. *In Dokl. Akad. Nauk SSSR*, volume 212, pages 2–2, 1973. Issue: 548-550.
- [141] Leonid A. LEVIN : Laws of Information Conservation (Non-growth) and Aspects of the Foundation of Probability Theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974. Publisher: Russian Academy of Sciences.
- [142] Andrew LEWIS-PYE : Strong Minimal Covers and a Question of Yates: The Story so Far. *In Logic Colloquium 2006*, volume 32 de *Lect. Notes Log.*, pages 213–228. Assoc. Symbol. Logic, Chicago, IL, 2009.
- [143] Ming LI et Paul VITÁNYI : *An Introduction to Kolmogorov Complexity and its Applications*, volume 3. Springer, 2008.
- [144] J. E. LITTLEWOOD : *Lectures on the Theory of Functions*. Oxford University Press, 1944.
- [145] Jiayi LIU :  $RT^2_2$  does not Imply  $WKL_0$ . *The Journal of Symbolic Logic*, pages 609–620, 2012. Publisher: JSTOR.
- [146] Richard MANSFIELD : Perfect Subsets of Definable Sets of Real Numbers. *Pacific Journal of Mathematics*, 35(2):451–457, 1970. Publisher: Mathematical Sciences Publishers.
- [147] Donald A. MARTIN : Classes of Recursively Enumerable Sets and Degrees of Unsolvability. *Mathematical Logic Quarterly*, 12(1):295–310, 1966. Publisher: Wiley Online Library.
- [148] Donald A. MARTIN : Borel Determinacy. *Annals of Mathematics*, pages 363–371, 1975.
- [149] Per MARTIN-LÖF : The Definition of Random Sequences. *Information and Control*, 9(6):602–619, 1966. Publisher: Elsevier.
- [150] A. R. David MATHIAS : *On a Generalization of Ramsey's Theorem*. Thèse de doctorat, University of Cambridge, 1969.

- [151] A. R. David MATHIAS : Happy Families. *Ann. Math. Logic*, 12(1):59–111, 1977.
- [152] Kenneth McALOON : Paris-Harrington Incompleteness and Progressions of Theories. In *Recursion Theory (Ithaca, N.Y., 1982)*, volume 42 de *Proc. Sympos. Pure Math.*, pages 447–460. Amer. Math. Soc., Providence, RI, 1985.
- [153] Yuri T. MEDVEDEV : Degrees of Difficulty of the Mass Problem. In *Doklady Akademii Nauk SSSR, Ns*, volume 104, pages 501–504, 1955.
- [154] Zdzisław Alexander MELZAK : An Informal Arithmetical Approach to Computability and Computation. *Canadian Mathematical Bulletin*, 4(3):279–293, 1961.
- [155] Joseph R. MILETI : Partition Theorems and Computability Theory. *The Bulletin of Symbolic Logic*, 11(3):411–427, 2005. Publisher: JSTOR.
- [156] Webb MILLER et Donald A. MARTIN : The Degrees of Hyperimmune Sets. *Mathematical Logic Quarterly*, 14(7-12):159–166, 1968. Publisher: Wiley Online Library.
- [157] Benoît MONIN : Higher Randomness and Forcing with Closed Sets. *Theory of Computing Systems*, 60(3):421–437, 2017.
- [158] Benoît MONIN et Ludovic PATEY : Pigeons do not jump high. *Advances in Mathematics*, 352:1066–1095, 2019.
- [159] Benoît MONIN et Ludovic PATEY : SRT<sub>22</sub> does not imply COH in Omega-Models, 2019.
- [160] Benoît MONIN et Ludovic PATEY : The Weakness of the Pigeonhole Principle under Hyperarithmetical Reductions, 2019.
- [161] Yiannis John MOSCHOVAKIS : Many-One Degrees of the Predicates  $\text{Ha}(X)$ . *Pacific Journal of Mathematics*, 18(2):329–342, 1966. Publisher: Mathematical Sciences Publishers.
- [162] Albert A. MUCHNIK : On the Unsolvability of the Problem of Reducibility in the Theory of Algorithms. In *Dokl. Akad. Nauk SSSR*, volume 108, pages 194–197, 1956. Issue: 1.
- [163] Albert A. MUCHNIK : Solution of Post’s Reduction Problem and of Certain Other Problems in the Theory of Algorithms. *Trudy Moskovskogo Matematicheskogo Obshchestva*, 7:391–405, 1958.
- [164] Albert A. MUCHNIK : Strong and Weak Reducibility of Algorithmic Problems 1. *Computability: The Journal of the Association CiE*, 5(1):49–59, 2016. Traduction de l’article original de 1963.
- [165] André NIES : Lowness Properties and Randomness. *Advances in Mathematics*, 197(1):274–305, 2005.
- [166] André NIES : *Computability and Randomness*, volume 51. Oxford University Press, 2009.

- [167] André NIES, Richard A. SHORE et Theodore A. SLAMAN : Interpretability and Definability in the Recursively Enumerable Degrees. *Proceedings of the London Mathematical Society*, 77(2):241–291, 1998.
- [168] Piergiorgio ODIFREDDI : *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. Elsevier, 1992.
- [169] J. B. PARIS et L. A. S. KIRBY :  $\Sigma_n$ -Collection Schemas in Arithmetic, 1978.
- [170] Charles PARSONS : On a Number Theoretic Choice Schema and its Relation to Induction. In *Intuitionism and Proof Theory (Proc. Conf., Buffalo, N.Y., 1968)*, pages 459–473. North-Holland, Amsterdam, 1970.
- [171] Ludovic PATEY : The Weakness of Being Cohesive, Thin or Free in Reverse Mathematics. *Israel J. Math.*, 216(2):905–955, 2016.
- [172] Ludovic PATEY et Keita YOKOYAMA : The Proof-Theoretic Strength of Ramsey’s Theorem for Pairs and Two Colors. *Adv. Math.*, 330:1034–1070, 2018.
- [173] Rózsa PÉTER : Programmierung und Partiell-Rekursive Funktionen. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3-4):373–401, 1963.
- [174] Von Rózsa PÉTER : Graphschemata und Rekursive Funktionen. *Dialectica*, 12(3-4):373–393, 1958.
- [175] H. POINCARÉ : *Dernières pensées*. Flammarion, 1917.
- [176] Henri POINCARÉ : L’intuition et la logique en mathématiques. *La valeur de la Science*.
- [177] Henri POINCARÉ : *Science et méthode*. Flammarion, 1908.
- [178] Christopher P. PORTER : *Mathematical and Philosophical Perspectives on Algorithmic Randomness*. University of Notre Dame, 2012.
- [179] David B. POSNER et Robert W. ROBINSON : Degrees Joining to  $0'$ . *Journal of Symbolic Logic*, pages 714–722, 1981. Publisher: JSTOR.
- [180] Emil L. POST : Recursively Enumerable Sets of Positive Integers and their Decision Problems. *Bulletin of the American Mathematical Society*, 50(5):284–316, 1944.
- [181] Tibor RADO : On Non-Computable Functions. *Bell System Technical Journal*, 41(3):877–884, 1962.
- [182] Hartley ROGERS, Jr. : Gödel Numberings of Partial Recursive Functions. *J. Symbolic Logic*, 23:331–341, 1958.
- [183] Barkley ROSSER : An Informal Exposition of Proofs of Godel’s Theorems and Church’s Theorem. *Journal of Symbolic Logic*, 4(2):53–60, 1939. Publisher: Association for Symbolic Logic.
- [184] Gerald E. SACKS : A Minimal Degree less than  $0'$ . *Bull. Amer. Math. Soc.*, 67:416–419, 1961.

- [185] Gerald E. SACKS : On Suborderings of Degrees of Recursive Unsolvability. *Mathematical Logic Quarterly*, 7(1-5):46–56, 1961.
- [186] Gerald E. SACKS : A Maximal Set which is not Complete. *Michigan Mathematical Journal*, 11(3):193–205, 1964.
- [187] Gerald E. SACKS : *Degrees of Unsolvability*. Numéro 55. Princeton University Press, 1966.
- [188] Gerald E. SACKS : Measure-Theoretic Uniformity in Recursion Theory and Set Theory. *Transactions of the American Mathematical Society*, 142:381–420, 1969.
- [189] Gerald E. SACKS : Forcing with Perfect Closed Sets. In *Axiomatic Set Theory*, volume 1, pages 331–355. Amer. Math. Soc Providence, RI, 1971.
- [190] Gerald E. SACKS : Countable Admissible Ordinals and Hyperdegrees. *Advances in Mathematics*, 20(2):213–262, 1976. Publisher: Academic Press.
- [191] Gerald E. SACKS : *Higher Recursion Theory*, volume 2. Cambridge University Press, 2017.
- [192] Claus-Peter SCHNORR : The Process Complexity and Effective Random Tests. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, pages 168–176, 1972.
- [193] Dana SCOTT : Algebras of Sets Binumerable in Complete Extensions of Arithmetic. In *Proc. Sympos. Pure Math*, volume 5, pages 117–121, 1962.
- [194] David SEETAPUN et Theodore SLAMAN : On the Strength of Ramsey’s Theorem. *Notre Dame Journal of Formal Logic*, 36(4):570–582, 1995. Publisher: University of Notre Dame.
- [195] Adi SHAMIR :  $IP = PSPACE$ . *Journal of the ACM (JACM)*, 39(4): 869–877, 1992.
- [196] John C. SHEPHERDSON et Howard E. STURGIS : Computability of Recursive Functions. *Journal of the ACM (JACM)*, 10(2):217–255, 1963.
- [197] J. R. SHOENFIELD : A Theorem on Minimal Degrees. *J. Symbolic Logic*, 31:539–544, 1966.
- [198] Richard A. SHORE : On the  $\forall\exists$ -Sentences of  $\alpha$ -Recursion Theory. In *Studies in Logic and the Foundations of Mathematics*, volume 94, pages 331–353. Elsevier, 1978.
- [199] Richard A. SHORE et Theodore A. SLAMAN : Defining the Turing Jump. *Mathematical Research Letters*, 6(6):711–722, 1999.
- [200] Stephen G. SIMPSON : First-Order Theory of the Degrees of Recursive Unsolvability. *Annals of Mathematics*, pages 121–139, 1977.
- [201] Stephen G. SIMPSON : Partial Realizations of Hilbert’s Program. *J. Symbolic Logic*, 53(2):349–363, 1988.

- [202] Stephen G. SIMPSON : *Subsystems of Second Order Arithmetic*, volume 1. Cambridge University Press, 2009.
- [203] Theodore A. SLAMAN :  $\Sigma_n$ -Bounding and  $\Delta_n$ -Induction. 132:2449–2449, 2004.
- [204] Theodore A SLAMAN et John R STEEL : Definable Functions on Degrees. In *Cabal Seminar 81–85*, pages 37–55. Springer, 1988.
- [205] Theodore A. SLAMAN et William H. WOODIN : Definability in the Turing Degrees. *Illinois Journal of Mathematics*, 30(2):320–334, 1986.
- [206] Theodore A. SLAMAN et William H. WOODIN : Definability in Degree Structures. *Preprint*, 2005.
- [207] Robert I. SOARE : Turing Computability. *Theory and Applications of Computability*. Springer, 2016.
- [208] Ray J. SOLOMONOFF : A Preliminary Report on a General Theory of Inductive Inference. United States Air Force, Office of Scientific Research, 1960.
- [209] Ray J. SOLOMONOFF : A Formal Theory of Inductive Inference. Part I. *Information and Control*, 7(1):1–22, 1964. Publisher: Elsevier.
- [210] Robert M SOLOVAY : On the Cardinality of  $\Sigma_2^1$  Sets of Reals. In *Foundations of Mathematics*, pages 58–73. Springer, 1969.
- [211] Robert M. SOLOVAY : Draft of Paper (Or Series of Papers) on Chaitin’s Work. mai 1975.
- [212] Ernst SPECKER : Ramsey’s Theorem does not hold in Recursive Set Theory. In *Studies in Logic and the Foundations of Mathematics*, volume 61, pages 439–442. Elsevier, 1971.
- [213] Clifford SPECTOR : Recursive Well-Orderings. *The Journal of Symbolic Logic*, 20(2):151–163, 1955. Publisher: JSTOR.
- [214] Clifford SPECTOR : On Degrees of Recursive Unsolvability. *Ann. of Math. (2)*, 64:581–592, 1956.
- [215] John R. STEEL : Forcing with Tagged Trees. *Annals of Mathematical Logic*, 15(1):55–74, 1978. Publisher: Elsevier.
- [216] John R STEEL : A Classification of Jump Operator. *The Journal of Symbolic Logic*, 47(2):347–358, 1982.
- [217] Frank STEPHAN : Martin-Löf Random and PA-complete Sets. In *Logic Colloquium*, volume 2, pages 342–348. Association for Symbolic Logic and AK Peters, Ltd, 2006.
- [218] Mikhail SUSLIN : Sur une définition des ensembles mesurables B sans nombres transfinis. *CR Acad. Sci. Paris*, 164(2):88–91, 1917.
- [219] William W TAIT : Finitism. *The Journal of Philosophy*, pages 524–546, 1981.
- [220] Hisao TANAKA *et al.* : A Basis Result for  $\Pi_1^1$ -Sets of Positive Measure. *Rikkyo Daigaku sugaku zasshi*, 16(2):115–127, 1967.

- [221] S. A. TERWIJN et D. ZAMBELLA : Algorithmic Randomness and Lowness. *ILLC Technical Report*, ML-1997-07, 1997.
- [222] Steven K. THOMASON : Sublattices of the Recursively Enumerable Degrees. *Mathematical Logic Quarterly*, 17(1):273–280, 1971.
- [223] Henry TOWNSNER : A Simple Proof and some Difficult Examples for Hindman’s Theorem. *Notre Dame J. Form. Log.*, 53(1):53–65, 2012.
- [224] John TROMP et Gunnar FARNEBÄCK : Combinatorics of Go. *In International Conference on Computers and Games*, pages 84–99. Springer, 2006.
- [225] Alan Mathison TURING : Systems of Logic Based on Ordinals. *Proceedings of the London Mathematical Society*, 2(1):161–228, 1939. Publisher: Wiley Online Library.
- [226] Michiel van LAMBALGEN : *Random Sequences*. Ph.D. Dissertation, University of Amsterdam, 1987.
- [227] Jean VILLE : Étude critique de la notion de collectif. *Bull. Amer. Math. Soc.*, 45(11):824, 1939.
- [228] Hao WANG : A Variant to Turing’s Theory of Computing Machines. *Journal of the ACM (JACM)*, 4(1):63–92, 1957.
- [229] Hao WANG : *Popular Lectures on Mathematical Logic*. Dover Publications, Inc., New York, second édition, 1993XXX???
- [230] Wei WANG : Some Logically Weak Ramseyan Theorems. *Advances in Mathematics*, 261:1–25, 2014.
- [231] Wei WANG : The Definability Strength of Combinatorial Principles. *J. Symb. Log.*, 81(4):1531–1554, 2016.
- [232] K. WEIHRAUCH : *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2000.
- [233] Mike C. E. YATES : A Minimal Pair of Recursively Enumerable Degrees. *Journal of Symbolic Logic*, 31(2):159–168, juin 1966.
- [234] Mike C. E. YATES : Initial Segments of the Degrees of Unsolvability Part II: Minimal Degrees. *The Journal of Symbolic Logic*, 35(2):243–266, 1970.
- [235] Liang YU : Descriptive Set Theoretical Complexity of Randomness Notions. *Fundamenta Mathematicae*, 215(3):219–231, 2011.



# Notations

$ E  =  F $	The sets $E$ and $F$ are equipotent	13
$ E  \leq  F $	L'ensemble $E$ est subpotent à l'ensemble $F$	14
$\langle n, m \rangle$	The image of $n, m$ by Cantor's bijection	17
$\langle x_1, \dots, x_k \rangle$	The image of $x_1, \dots, x_k$ by Cantor's bijection	18
$\mathcal{P}(A)$	L'ensemble des sous-ensembles de $A$	20
$2^{\mathbb{N}}$	Cantor space	22
$X(n)$	Le $n$ -ième élément de $X$ (en commençant à 0)	23
$f(x) = g(x)$	The computable functions $f$ and $g$ halt on $x$ and return the same value, or both diverge on $x$ .	32
$\Phi_e$	The computable function of code $e$	33
$\Phi_e(x) \downarrow$	L'exécution de $\Phi_e$ s'arrête sur l'entrée $x$	33
$\Phi_e(x) \uparrow$	L'exécution de $\Phi_e$ ne s'arrête pas sur l'entrée $x$	33
$\Phi_e(x) \downarrow = y$	L'exécution de $\Phi_e$ s'arrête sur l'entrée $x$ et retourne la valeur $y$	33
$\Phi_e(x) \uparrow \neq y$	L'exécution de $\Phi_e$ ne s'arrête pas sur l'entrée $x$ ou bien s'arrête et retourne une valeur différente de $y$	33
$\Phi_e(x)[t] \downarrow$	The execution of $\Phi_e$ halts on input $x$ in less than $t$ computation steps	34
$\Phi_e(x)[t] \uparrow$	The execution of $\Phi_e$ does not halt on input $x$ in less than $t$ computation steps	34
$\pi_0, \pi_1$	Les fonctions inverses de $(n, m) \mapsto \langle n, m \rangle$	35
$A[s]$	Approximation of $A$ at step $s$	45
$\emptyset'$	Halting problem	45
$2^{<\mathbb{N}}$	The set of binary strings	49
$\epsilon$	The empty string	49
$\sigma\tau$	The concatenation of $\sigma$ and $\tau$	49
$\sigma \preceq \tau$	The string $\sigma$ is a prefix of the string $\tau$	49
$ \sigma $	The size of $\sigma$	49
$\sigma(n)$	The bit number $n$ of $\sigma$ (starting at 0)	49
$\sigma X$	The concatenation of $\sigma$ and $X$	50
$\sigma \prec X$	The string $\sigma$ is a prefix of $X$	50
$X \upharpoonright_n$	The prefix of $X$ of size $n$	50
$\Phi_e(X, n)$	The result of computing the functional $\Phi_e$ with the oracle $A$ and on the input $n$	51
$\Phi_e^X(n)$	Other notation for $\Phi_e(X, n)$	51
$\Phi_e(X, n)[t] \downarrow$	The functional $\Phi_e$ with the oracle $X$ , halts in less than $t$ steps on the input $n$	51
$\Phi_e(X, n)[t] \uparrow$	The functional $\Phi_e$ with oracle $X$ , does not halt in less than $t$ steps on the input $n$	51

$\Phi_e(\sigma, n)\downarrow$	The functional $\Phi_e$ with the oracle piece $\sigma$ , halts in less than $t$ steps on the input $n$	54
$\Phi_e^\sigma(n)\downarrow$	The functional $\Phi_e$ with the oracle piece $\sigma$ , does not halt in less than $t$ steps on the input $n$	54
$\text{use}_\Phi^X$	The function of the usage of $X$ on the functional $\Phi$	54
$\leq_T$	Turing reduction	55
$\deg_T(X)$	The Turing degree of $X$	55
$\equiv_T$	Turing equivalence	55
$(\mathcal{D}, \leq)$	The structure of Turing degrees	55
$A \oplus B$	The join of $A$ and $B$	56
$\Phi_e(X) = A$	$\forall n \Phi_e(X, n) \downarrow = A(n)$	57
$X'$	The Turing jump of $X$	58
$[\sigma]$	The set $\{X : \sigma \prec X\}$	67
$\Sigma_n^0$ (set)	$\Sigma_n^0$ set	79
$\Pi_n^0$ (set)	$\Pi_n^0$ set	79
$\Delta_n^0$ (set)	Set $\Delta_n^0$	81
$\Sigma_1^0$ (set.)	c.e. set	84
$\Delta_1^0$ (set)	Computable set	85
$\Sigma_n^0(X)$ (set)	Set $\Sigma_n^0$ relative to $X$	86
$\Pi_n^0(X)$ (set)	Set $\Pi_n^0$ relative to $X$	86
$\Delta_n^0(X)$ (ens.)	Set $\Delta_n^0$ relative to $X$	86
$\leq_m$	Many-one reduction	87
$\equiv_m$	Many-one equivalence	87
$X^{(n)}$	The $n$ -th Turing jump of $X$	88
$W_e$	The c.e. set of code $e$	93
$W_e^X$	The set $X$ -c.e. of code $e$	93
$g(\bar{x})$	A shortcut for $g(x_1, \dots, x_n)$ (with $g : \mathbb{N}^n \rightarrow \mathbb{N}$ )	98
$\text{dom } f$	The domain of definition of the function $f$	109
$\text{Im } f$	The image of $f$	109
$p_i^n$	Recursive projection primitive function	115
$c_i^n$	Constant primitive recursive function	115
$\text{succ}$	Recursive successor primitive function	115
$\leq_{tt}$	Truth-table reduction	141
$\equiv_{tt}$	Truth-table equivalence	141
$\forall^\infty x$	For all $x$ except a finite number	144
$\exists^\infty x$	For an infinity of $x$	144
$[T]$	The set of paths of $T$	152
$i^\infty$	The infinite sequence which repeats the bit $i \in \{0, 1\}$	155
$[W]$	The set $\bigcup_{\sigma \in W} [\sigma]$	156
$\Sigma_1^0$ (class)	An effectively open	163
$\Pi_1^0$ (class)	An effectively closed	163
$f^{<\mathbb{N}}$	The set of strings $\sigma \in \mathbb{N}^{<\mathbb{N}}$ such that for all $n <  \sigma $ , $\sigma(n) < f(n)$	179
$\mathcal{Q}$	Robinson arithmetic	199

$T \vdash F$	$F$ is demonstrable in $T$	200
$T \not\vdash F$	$F$ is not demonstrable in $T$	200
$T \vdash \perp$	$T$ is inconsistent	200
$\mathcal{M} \models F$	$\mathcal{M}$ is a model of $F$	202
$\Delta_0$ (form.)	$\Delta_0$ formula	208
$\Sigma_n$ (form.)	$\Sigma_n^0$ formula	208
$\Pi_n$ (form.)	$\Pi_n^0$ formula	208
$\text{Coh}(T)$	Formula which expresses the consistency of $T$	215
$\Sigma_n^0$ (form.)	$\Sigma_n^0$ Formula	227
$\Pi_n^0$ (form.)	$\Pi_n^0$ Formula	227
$\sigma \Vdash^* \mathcal{R}$	$\sigma$ forces the requirement $\mathcal{R}$ ( $\Sigma_1^0/\Pi_1^0$ ) case	229
$\sigma \perp W$	No extension of $\sigma$ is in $W$	242
$W^\perp$	The set $\{\sigma \in 2^{<\mathbb{N}} : \sigma \perp W\}$	242
$\bigoplus_{n \in \mathbb{N}} A_n$	The effective join of $(A_n)_{n \in \mathbb{N}}$	246
$\sigma \Vdash \mathcal{R}$	$\sigma$ semantically force the requirement $\mathcal{R}$	252
$\sigma \Vdash^* \mathcal{R}$	$\sigma$ forces the requirement $\mathcal{R}$ (general case)	253
$\mathcal{B}\Delta\mathcal{U}$	The class $(\mathcal{B} \setminus \mathcal{U}) \cup (\mathcal{U} \setminus \mathcal{B})$	255
$[c]_\in$	The set of maximum filters containing $c$	266
$\dot{F}$	The element corresponding to the maximum filter $F$ in a Cantor forcing	267
$[c]$	The set $\{\dot{F} : F \in [c]_\in\}$	268
$? \vdash$	Forcing question (J.-S. or S., $\Sigma_1^0/\Pi_1^0$ case)	279
$? \vdash$	Question of forcing (J.-S. or S., general case)	282
$? \vdash$	forcing question	283
$\leq_w$	Muchnik reduction	297
$\leq_s$	Medvedev reduction	297
$\mathbf{0}_w$	The Muchnik degree of $\{\emptyset\}$	298
$\mathbf{a} \cup \mathbf{b}$	The least upper bound of $\mathbf{a}$ and $\mathbf{b}$	331
$C_M(\sigma)$	The Kolmogorov complexity of $\sigma$ for the machine $M$	356
$C(\sigma)$	The Kolmogorov complexity of $\sigma$	357
$C(n)$	The Kolmogorov complexity of $n$	360
$\leq^+$	Less than or equal to near additive constant	360
$=^+$	Equal to near additive constant	360
$K_M(\sigma)$	The prefix-free complexity of $\sigma$ for the machine $M$	366
$K(\sigma)$	The prefix-free complexity of $\sigma$	366
$\Omega$	The number $\Omega$ of Chaitin	368
$\text{weight}(A)$	The weight of a bounded requests set $A$	371
$\Pi_2^0$ (class)	An effective intersection of effectively open classes	384
$\Sigma_n^0$ (class)	Borel class $\Sigma_n^0$	386
$\Pi_n^0$ (class)	Borel class $\Pi_n^0$	386
$\Delta_n^0$ (class)	Borel class $\Delta_n^0$	386
$\Sigma_n^0$ (class)	Effective Borel $\Sigma_n^0$ class	387

$\Pi_n^0$ (class)	Effective Borel $\Pi_n^0$ class	387
$\Delta_n^0$ (class)	Effective Borel $\Delta_n^0$ class	387
$\lambda$	Lebesgue measure	391
$\lambda(\mathcal{B} \mid [\sigma])$	The measure of $\mathcal{B} \cap [\sigma]$ inside $[\sigma]$	403
$K_M^X(\sigma)$	The prefix-free complexity of $\sigma$ for the machine $M$ , relativized to $X$	416
$\rho(\mathcal{B} \mid Z)$	The lower density of $Z$ in $\mathcal{B}$	425
$\bar{\rho}(\mathcal{B} \mid Z)$	The top density of $Z$ in $\mathcal{B}$	425
$\mathcal{L}_{Z_2}$	The language of second-order arithmetic	466
$Z_2$	The theory of second-order arithmetic	469
$\omega$	Standard integers	472
$\text{RCA}_0$	The system $\text{RCA}_0$	481
$\text{ACA}_0$	The system $\text{ACA}_0$	485
$\text{WKL}_0$	The system $\text{WKL}_0$	488
$\text{WWKL}_0$	The system $\text{WWKL}_0$	492
$\text{ACA}'_0$	The system $\text{ACA}'_0$	496
$\text{ACA}_0^+$	The theory $\text{ACA}_0^+$	496
$\text{ATR}_0$	The system $\text{ATR}_0$	498
$\Pi_1^1\text{-CA}_0$	The system $\Pi_1^1\text{-CA}_0$	499
$\mathbf{l}_{\text{open}}$	The induction scheme for quantifier-free formulas	505
$\mathbf{l}\Delta_0^0$	The system $\mathbf{l}\Delta_0^0$	508
$x \mid y$	$x$ divide $y$	508
$\mathbf{l}\Sigma_n^0$	The induction scheme for the $\Sigma_n^0$ formulas	510
$\mathbf{l}\Pi_n^0$	The induction scheme for the $\Pi_n^0$ formulas	510
$\text{B}\Sigma_n^0$	The bounded collection scheme for $\Sigma_n^0$ formulas	510
$\text{B}\Pi_n^0$	The bounded collection scheme for $\Pi_n^0$ formulas	510
$\mathbf{L}\Sigma_n^0$	The minimum scheme for the $\Sigma_n^0$ formulas	512
$\mathbf{L}\Pi_n^0$	The minimum scheme for the $\Pi_n^0$ formulas	512
$\mathbf{l}\Delta_n^0$	The induction scheme for the $\Delta_n^0$ formulas	515
$x \in y$	$x$ belongs to the set coded by $y$	520
$\text{B}\Sigma_n^0$	The bounded comprehension scheme for $\Sigma_n^0$ formulas	522
$\text{B}\Pi_n^0$	The bounded comprehension scheme for $\Pi_n^0$ formulas	522
$\text{B}\Delta_n^0$	The bounded comprehension scheme for $\Delta_n^0$ formulas	522
$\Sigma_1\text{-PA}$	The system $\Sigma_1\text{-PA}$	527
$\mathcal{M}[G]$	The $\omega$ -extension of $\mathcal{M}$ consisting of sets definable by $\Delta_1^0$ formulas with parameters in $\mathcal{M} \cup \{G\}$	529
$\mathcal{PR}$	The class of primitive recursive functions	535
$\text{PRA}$	The system $\text{PRA}$	535
$\Pi_2^1$ (prob.)	A problem of the form $\forall X (F(X) \rightarrow \exists Y G(X, Y))$	539
$\text{ADS}$	Statement “Ascending-Descending Sequence”	540
$\leq_\omega$	$\omega$ -reduction	540
$\text{KL}$	The problem associated with König’s lemma	541
$\text{J}$	The Turing jump problem	541
$\leq_c$	Computable reduction	545

$\leq_W$	Weihrach reduction	547
LPO	The Limited Principle of Omniscience	549
LLPO	The Lesser Limited Principle of Omniscience	549
$\hat{P}$	The parallelization of the problem $P$	549
$G(Q \rightarrow P)$	Reduction game between the problems $P$ and $Q$	551
$P \leq_\omega^n Q$	Player 2 has a winning strategy for the game $G(Q \rightarrow P)$ in at most $n + 1$ turns	552
$\leq_{sc}$	strong computational reduction	553
$\leq_{sW}$	Strong Weihrach reduction	553
$[X]^n$	The set of subsets of $X$ of size $n$	558
$RT_k^n$	Ramsey's theorem for $n$ -tuples and $k$ colors	558
$cRT_k^n$	Variant of Ramsey's theorem for Weihrach reduction	560
$\sigma \cup \tau$	The union of the strings $\sigma$ and $\tau$ seen as sets	576
$\tau - \sigma$	The difference of the strings $\sigma$ and $\tau$ seen as sets	576
$? \vdash$	Forcing question for the Dzhafarov-Jockusch forcing	579
$? \nvdash$	Negation of the forcing question	579
COH	The principle of existence of cohesive set	591
$\mathcal{O}$ (Kleene)	The set $\mathcal{O}$ of Kleene	612
$<_o$	The partial order on the elements of $\mathcal{O}$	612
$H_a$	The iteration of the jump on the ordinal code $a$	614
$\mathfrak{n}$	The finite ordinal $\mathfrak{n}$	614
$\alpha < \beta$	The ordinal $\alpha$ is smaller than the ordinal $\beta$	616
$\text{succ}(\alpha)$	The successor of the ordinal $\alpha$	618
$\sup_{i \in I} \alpha_i$	The supremum of the ordinals $(\alpha_i)_{i \in I}$	618
Ord	The class of ordinals	619
$ < $	The isomorphic ordinal with the well-founded order $<$	621
$\Sigma_\alpha^0$ (class)	Borel class $\Sigma_\alpha^0$	623
$\Pi_\alpha^0$ (class)	Borel class $\Pi_\alpha^0$	623
$\omega_1$	The smallest uncountable ordinal	629
$\omega_n$	The smallest non-subpotent ordinal at $\omega_{n-1}$	632
$\mathcal{O}_{<\alpha}$	The set of ordinals codes less than $\alpha$	633
$\mathcal{O}_{=\alpha}$	The set of codes of the ordinal $\alpha$	633
$\omega_1^{ck}$	Smallest non-computable ordinal	633
$T \upharpoonright_\sigma$	The tree of nodes of $T$ comparable with $\sigma$	637
$T \upharpoonright_\sigma$	The tree $T \upharpoonright_\sigma$ but in "taking $\sigma$ as root"	637
$ T $	The ordinal encoded by the well-founded tree $T$	637
$\mathcal{T}$	The set of c.e. codes of well-founded trees	637
$\mathcal{T}_{<\alpha}$	The set of c.e. codes of well-founded trees such that $ T  < \alpha$	637
$\mathcal{T}_{=\alpha}$	The set of c.e. codes of well-founded trees such that $ T  = \alpha$	637
$<_{KB}$	The order of Kleene-Brouwer	638
$\mathcal{O}^X$	The set $\mathcal{O}$ of Kleene relativized to $X$	639
$\mathcal{T}^X$	The set $\mathcal{T}$ put into perspective $X$	640
$\omega_1^X$	The smallest non $X$ -computable ordinal	640

$\Sigma_\alpha^0$ (set)	Set $\Sigma_\alpha^0$	641
$\Pi_\alpha^0$ (set)	Set $\Pi_\alpha^0$	641
$\Delta_\alpha^0$ (set)	Set $\Delta_\alpha^0$	641
$\emptyset^{(\alpha)}$	The $\alpha$ iteration of the Turing jump	649
$H_a^X$	The iteration of the jump from $X$ on the ordinal code $a$	651
$\Sigma_\alpha^0$ (class)	Effective $\Sigma_\alpha^0$ class	652
$\Pi_\alpha^0$ (class)	Effective $\Pi_\alpha^0$ class	652
$\Delta_\alpha^0$ (class)	Class effective $\Delta_\alpha^0$ borelian	652
$\Sigma_1^1$ (set)	Set $\Sigma_1^1$	662
$\Pi_1^1$ (set)	Set $\Pi_1^1$	662
$\Delta_1^1$ (set)	Set $\Delta_1^1$	662
$\Sigma_1^1$ (class)	Effective analytical class	662
$\Pi_1^1$ (class)	Effective co-analytical class	662
$\Delta_1^1$ (class)	Effective $\Delta_1^1$ class	662
$\Sigma_1^1$ (class)	Analytical class	666
$\Pi_1^1$ (class)	Co-analytic class	666
$\vee$	Function “or” on trees	678
$\wedge$	Function “and” on trees	678
$\leq_h$	Hyperarithmetical reduction	682
HYP	The class of hyperarithmetical sets	694
$\mathcal{S}$	The class of sets which compute $\omega_1^{ck}$	694
$X \leq_T \emptyset^{(\alpha)}$	$\forall Y (\alpha < \omega_1^Y \rightarrow X \leq_T Y^{(\alpha)})$	695
$\mathcal{C}$	The largest $\Pi_1^1$ class which does not contain a perfect closed	695
$J_a$	The hyperjump iterated on the code $a$	702
$\omega_\alpha^{ck}$	The $\alpha$ -th smallest non ordinal $X$ -computable for some $X$	703
$\text{WO}(<, A)$	The order $<$ is indeed based on $A$	707
$L_O(a, X)$	$a$ is a linear order code with oracle $X$	710
$W_O(a, X)$	$a$ is a well-founded order code with oracle $X$	710
$\text{PBO}^X$	The set of $X$ -nicknames well-orders	712
$\text{SUPP}_X$	The set of jump supports	713

# Index

- 1-generic (set), 242
- 1-random (set), 420
- 2-random (set), 419
- $\Pi_1^1$ -CA<sub>0</sub>, 499
- $\beta$ -model, 721
- $\lambda$ -calculus, 100
- $\lambda$ -function, 100
- $\omega$ -extension, 526, 529
- $\omega$ -reduction, 540
- $\omega$ -structure, 473
- $\omega$ -substructure, 526, 529
- $n$ -generic (set), 246
- $n$ -random (set), 420
  
- ACA'<sub>0</sub>, 496
- ACA<sup>+</sup><sub>0</sub>, 496
- ACA<sub>0</sub>, 485
- Ackermann
  - function, 118
- adaptive
  - Cost function, 444
- analytic
  - Class, 662, 666
  - Set, 662
  - hierarchy, 498
- anti-chain
  - Turing degree, 336
- approachable
  - From above, 358
  - From the left, 368
- approximation
  - $\Delta_2^0$ , 61
  - From above, 358
  - From the left, 368
  - c.e., 44
- arithmetic
  - formula, 192, 467
  - hierarchy, 79
  - language, 191
  - Peano, 198
  - primitive recursive, 535
  - Robinson, 198, 504
  - second-order, 218, 226, 465
  - term, 192
- Arslanov
  - completeness criterion, 134
- atomic
  - formula, 192
- ATR<sub>0</sub>, 498
- avoidance
  - of  $k$  cones, 544
- avoidance (strings), 242
- axiom
  - Choice, 629, 686, 699
  - Countable choice, 630
  - bounded comprehension, 522
  - choice, 222, 223
  - collection, 510
  - comprehension, 219, 468, 479
  - extensionality, 219
  - foundation, 221
  - induction, 469, 480, 505, 508, 510, 515
  - infinity, 220
  - minimum, 512
  - ordered induction, 514
  - replacement, 220
- axiomatic
  - system, 198
- Baire
  - category, 231, 234
  - closed class, 177
  - open class, 177
  - property, 255
  - space, 176
- base
  - Class  $\Sigma_1^1$ , 687
  - For random, 430
- basis
  - $\Pi_1^0$  class, 166
  - Hyperarithmetical cone avoidance
    - basis, 690

- computably dominated, 167
  - cone avoidance, 169
  - hyperlow, 687
  - low, 166
  - perfect  $\Pi_1^0$  class, 169
- bijection, 13
  - Cantor, 116
- Borel
  - Effective hierarchy, 387
  - Hierarchy, 387, 623
  - effective hierarchy, 652
- borelian
  - Class, 386
  - Hierarchy, 386
- Borélien
  - Class, 622
- bounded
  - comprehension, 522
  - variable, 193
- branching
  - node, 176
- busy beaver, 359
- c.e.
  - left, 368
  - tree, 637
  - approximation, 44
  - degree, 134, 300
  - operator, 294
  - set, 43
  - tree, 327
- c.e. array, 129
- canonical
  - code of a set, 94, 129, 520
- Cantor
  - bijection, 17, 116
  - closed class, 156
  - compact, 158
  - continuous function, 160
  - cylinder, 156
  - forcing, 268
  - open class, 156
  - space, 22, 155
  - triadic set, 24
- Cantor-Bernstein, 15
- cappable
  - degree, 321
- capture
  - Difference test, 423
  - Martin-Löf test, 400
  - Solovay test, 401
- categorical
  - theory, 471
- category
  - Baire, 231, 234
- Cauchy
  - sequence, 476
- chain
  - Turing degree, 336
- Chaitin
  - $\Omega$  (number), 368
  - Chaitin's theorem, 362
  - Theorem KC, 371
- Chaitin/Levin random set, 367
- Chinese Remainder Theorem, 210
- choice
  - Axiom, 629, 686, 699
  - axiom, 222, 223
- Church-Turing (thesis), 102
- class, 151
  - $\Pi_1^0$ , 163
  - $\Sigma_1^0$ , 163
  - $\Delta_n^0$ , 386
  - $\tilde{\Pi}_n^0$ , 386
  - $\tilde{\Pi}_1^1$ , 666
  - $\Sigma_n^0$ , 386
  - $\Sigma_1^1$ , 666
  - $\mathcal{C}$ , 695
  - $\Delta_n^0$ , 387
  - $\Delta_1^1$ , 662
  - HYP, 694
  - $\Pi_2^0$ , 384
  - $\Pi_n^0$ , 387
  - $\Pi_1^1$ , 662
  - $\mathcal{S}$ , 694
  - $\Sigma_n^0$ , 387
  - $\Sigma_1^1$ , 662
  - analytic, 666
  - Borélienne, 622
  - co-analytic, 666
  - Measurable, 392, 670
  - oracle, 416
  - Set theory, 619

- closed, 156
- co-meager, 233
- compact, 158
- dense, 232
- interior, 233
- large, 583
- meager, 233
- open, 156
- perfect, 161, 234
- universal, 174
- clique, 559
- closed
  - Baire, 177
  - class, 156
  - effective, 163
  - formula, 193
  - term, 192
- closure
  - formula, 467
- co-analytic
  - Class, 666
  - Set, 662
- co-meager (class), 233
- code
  - $\Pi_1^0$  class, 163
  - $\Sigma_1^0$  class, 163
  - $\Delta_1^0$  (set), 92
  - $\Delta_\alpha^0$  (class), 652
  - $\Delta_\alpha^0$  (set), 641
  - $\Delta_n^0$  (set), 93
  - $\Pi_\alpha^0$  (class), 652
  - $\Pi_\alpha^0$  (set), 641
  - $\Sigma_1^0$  (set), 93
  - $\Sigma_\alpha^0$  (class), 652
  - $\Sigma_\alpha^0$  (set), 641
  - $\Sigma_n^0$  (set), 93
  - of lowness, 94
  - program, 33
- coding
  - acceptable, 37
  - lists, 120, 518
  - program, 33
- coding theorem, 374
- cohesive
  - set, 591
- cohesiveness
  - principle, 591
- collection
  - scheme, 510
- coloring (function), 558
- compact
  - class, 158
  - forcing question, 285
- compactness, 158
- compatibility
  - condition, 266
  - string, 49
- complete
  - theory, 171, 205
  - Turing, 59
- complete set
  - $\Pi_1^1$ , 669
  - $\Sigma_\alpha^0$ , 645, 648
  - $\Sigma_1^1$ , 719
- completeness
  - Gödel theorem, 205
- completion
  - Peano arithmetic, 172
- complexity
  - Full, 357
  - Kolmogorov, 356, 357
  - Plain, 366
  - Without prefix, 366
- comprehension
  - $\Delta_1^0$ , 479
  - scheme, 468, 708
- computable
  - Ordinal, 634
  - by goto program, 109
  - by structured program, 109
  - function, 30, 34
  - reduction, 545
  - relative to  $X$ , 51
  - set, 34
- computably
  - bounded (tree), 179
  - dominated, 137
  - enumerable, 43, 134
  - independent, 334
  - reducible, 545
  - reducible (strongly), 553
  - true, 547

- computation
  - function, 62
- computation step, 33
- computation time, 33
- condition
  - compatible, 266
  - Dzhafarov-Jockusch, 578
  - forcing, 265
  - incompatible, 266
  - Jockusch-Soare, 266
  - Mathias, 577
  - Sacks, 266
- cone
  - avoidance, 544
- cone (Turing degree), 168, 294
- conjecture
  - Martin, 295
- conservation
  - extension, 524
  - theorem, 524
- consistency
  - $\text{Coh}(T)$ , 215
  - theory, 200
- consistent
  - theory, 171
- continuous
  - Hypothesis, 631, 692, 701
  - function, 160, 477
- continuum
  - Hypothesis, 631, 692, 701
  - hypothesis, 224
- cost
  - Adaptive function, 444
  - Function, 444
- countable
  - Choice, 630
  - Ordinal, 629
  - set, 16
- cut, 512
- cylinder, 156
  - $[W]$ , 156
  - $[\sigma]$ , 67, 156
  - problem, 553
- decreasing intersection, 156
- deduction
  - lemma, 196
  - system, 194
- definable (in a structure), 526
- degree
  - Many-one, 87
  - c.e., 134, 300
  - cappable, 321
  - complete/incomplete, 59
  - computably dominated, 137
  - computably enumerable, 134
  - DNC, 130
  - $\text{DNC}_2$ , 133, 172
  - $\text{DNC}_f$ , 133
  - high, 72
  - hyperimmune, 136
  - low, 71
  - Medvedev, 298
  - minimal, 324
  - minimal cover, 328
  - Muchnik, 298
  - non-cappable, 321
  - PA, 172
  - spectrum, 173
  - strong minimal cover, 329
  - truth-table, 141
  - Turing, 55
- dense
  - below  $\sigma$ , 251
  - below  $c$ , 268
  - class, 232
  - set, 229, 267
- diagonally non-computable, 130
- difference
  - Random, 423
  - Test, 423
- distributive
  - lattice, 348
- DNC
  - degree, 130
  - function, 130
- $\text{DNC}_2$ , 133, 172
- $\text{DNC}_f$ , 133
- dominating function, 63
- domination, 135

- effective
  - Analytic class, 662
  - Analytic set, 662
  - Co-analytic class, 662
  - Co-analytic set, 662
- effectively immune, 129
- embedding
  - partial order, 334
- encoding
  - lists, 210
- ensemble
  - Martin-Löf random, 400
- Entscheidungsproblem, 97
- equipotence, 13
- equivalence
  - truth-table, 141
  - Turing, 55
- equivalence
  - Many-one, 87
- escaping
  - function, 239
- exponential
  - Ordinal, 624
- extendible
  - node, 154
- extension
  - $\omega$ -extension, 526, 529
  - compatible, 266
  - conservative, 524
  - consistent, 336
  - finite (method), 64
  - forcing, 265
  - string, 49
- f-tree, 138, 273
  - $\Gamma$ -splitting, 325
  - avoiding  $\Gamma$ -splitting, 325
  - path, 138
  - sub-f-tree, 138
  - uniform, 324
- filter, 266
  - generic, 267
  - sufficiently generic, 268
- finitely-branching tree, 176
- first-order
  - part, 525
  - term, 466
- first-order part, 525
- fixed-point free
  - function, 130
- force
  - Semantically, 252
  - semantically, 256, 268
  - syntactically, 253, 270
- forcing
  - Relation  $\Vdash$ , 252
  - Semantics, 252
  - Cohen, 225
  - condition, 265
  - Dzhafarov-Jockusch, 578
  - extension, 265
  - f-tree, 138
  - Jockusch-Soare, 271
  - Mathias, 577
  - question, 279, 282, 283
  - relation, 253, 269, 270
  - relation  $\Vdash$ , 256
  - relation  $\Vdash^*$ , 229, 253
  - sacks, 273
  - semantic, 268
  - semantics, 256
  - syntactic, 270
- form
  - prenex, 197
- formula
  - $\Delta_0$ , 208
  - $\mathcal{L}_{Z_2}$ , 467
  - $\Pi_n^0$ , 227
  - $\Pi_n$ , 208
  - $\Sigma_n^0$ , 227
  - $\Sigma_n$ , 208
  - arithmetic, 192, 467
  - atomic, 192
  - closed, 193, 467
  - functional, 502
  - literal, 192
  - parametric, 201
  - prenex form, 197
  - provably total, 502
  - second-order, 226
  - second-order arithmetic, 467
  - true, 202
  - with parameters, 474

- free
  - variable, 193
- full set
  - $\Pi_\alpha^0$ , 647
  - $\Pi_n^0$ , 88
  - $\Pi_n^0(X)$ , 88
  - $\Sigma_\alpha^0$ , 647
  - $\Sigma_n^0$ , 88
  - $\Sigma_n^0(X)$ , 88
- function
  - $\beta$  of Gödel, 210
  - succ (integers), 99, 115
  - $c_i^n$ , 99, 115
  - $p_i^n$ , 99, 115
  - Modulus, 650
  - Solovay, 453
  - Ackermann, 118
  - bijective, 13
  - computable, 30, 34
  - continuous, 160, 477
  - diagonally non-computable, 130
  - DNC, 130
  - DNC<sub>2</sub>, 133, 172
  - DNC<sub>f</sub>, 133
  - dominating, 63
  - domination, 135
  - escaping, 239
  - fixed-point free, 130
  - hyperimmune, 135
  - injective, 13
  - primitive recursive, 535
  - recursive, 104
    - primitive, 98
  - relativized primitive recursive, 504
  - represented, 210
  - stable, 59
  - surjective, 13
- functional, 50
  - formula, 502
  - Turing, 50
- Gödel
  - incompleteness theorem, 213, 215
- game
  - Borel, 220
  - determination of Martin, 221
  - reduction, 551
- general
  - recursion, 98
- general recursive
  - function, 98
  - predicate, 115
- generalized
  - low set, 244
- generalized low, 244
- generalized low<sub>n</sub>
  - set, 263
- generic
  - filter, 267
  - set, 230
  - sufficiently, 230
- goto (program), 107
- graph
  - clique, 559
- Gödel
  - $\beta$  function, 210
  - completeness theorem, 205
- halting
  - problem, 45
- height
  - Tree, 637
- Henkin
  - structure, 470
  - witness, 206, 207
- hierarchy
  - $\omega_\alpha^{ck}$  (ordinals), 703
  - Borel, 386, 623
  - Borel effective, 387
  - Hyperarithmetical, 641
  - Hyperjump, 702
  - Kleene, 641
  - Relativized arithmetic, 86
    - analytic, 498
    - arithmetic, 79
    - effectively Borel, 652
- high, 72
- Hilbert
  - program, 534
- homogeneous
  - set, 558

- hyperarithmetic
  - Hierarchy, 641
  - Reduction, 682
  - Set, 641
- hyperimmune
  - degree, 136
  - function, 135
  - set, 129, 136
- hyperjump, 702
  - Hierarchy, 702
- hyperlow, 687
- hypersaut, 683
- ideal
  - jump, 486
  - Scott, 488
  - Turing, 481
- immune, 128
- impredicativity, 188, 499
- incompatibility
  - string, 49
- increasing union, 156
- index set, 91
- induction
  - $\Delta_n^0$ , 515
  - $\Pi_n^0$ , 510
  - $\Sigma_1^0$ , 480
  - $\Sigma_n^0$ , 510
  - Transfinite, 625
  - scheme, 469
- infinite
  - set, 521
- injection, 13
- injury
  - finite, 301
  - infinite, 313
- instance
  - problem, 539
  - universal, 595
- integer
  - non-standard, 217, 472
  - standard, 472
- interior (class), 233
- invariant
  - operator, 294
  - uniformly, 294
- isolated point, 161
- isomorphism
  - Order, 621
- join
  - effective, 56, 246
  - lattice, 331
- jump
  - Hyperjump, 702
  - Iterated relativized, 651
  - Support, 712
  - Transfinite iterations, 614
  - ideal, 486
  - Turing, 58
- K-trivial, 374
- König
  - lemma, 177
  - weak lemma, 153
- KC (theorem), 371
- Kleene
  - $\mathcal{O}$ , 612
  - $\mathcal{O}$  relativized, 639
  - Hierarchy, 641
  - Ordinal, 633
  - fixed point, 39
- Kleene-Brouwer
  - Order, 638
- Kolmogorov
  - Complexity, 356, 357
- language
  - $\mathcal{L}_{Z_2}$ , 466
  - arithmetic, 191
  - first-order, 191
  - second-order arithmetic, 466
- large (class), 583
- lattice, 330
  - distributive, 348
  - join, 331
  - upper, 330
- leaf, 152
- Lebesgue
  - Density lemma, 404
  - Density theorem, 425
  - Measure, 383, 391
- left-c.e., 368

- lemma
  - Lebesgue density, 404
  - deduction, 196, 205
  - Gauss, 517
  - König, 177
  - padding, 38
  - Shoenfield limit, 59
  - weak König, 153
- limit-computable, 59
- literal
  - formula, 192
- low
  - For the Martin-Löf random, 419
  - For-K, 429
  - Hyperlow, 687
  - set, 71
- low-for-K, 429
- lower
  - Density, 425
  - bound, 324
- lower bound, 324
- lowness, 429
- machine, 356
  - oracle, 416
  - oracle, 52
  - prefix-free, 365
  - register, 104
  - Turing, 100
  - universal, 125
- many-one
  - Degree, 87
  - Reduction, 87
- Martin
  - conjecture, 295
  - determination theorem, 221
- Martin-Löf
  - Relativized random, 416
  - Relativized test, 416
  - Test, 400
  - random, 400
- mass (problem), 297
- Mathias
  - condition, 577
  - forcing, 577
- maximal
  - filter, 266
- meager (class), 233
- measurable
  - Class, 392, 670
- measure
  - Lebesgue, 383, 391
  - Relative, 403
  - positive, 492
  - tree, 492
- Medvedev
  - degree, 298
  - reduction, 297
- meets (string), 230
- merging ( $\Pi$ -merging), 286
- method
  - of finite extensions, 64
  - permitting, 300
  - priority, 301
- minimal
  - degree, 324
  - pair, 313
- minimal cover, 328
- minimum
  - scheme, 512
- MLR, 403
  - left-c.e., 424
  - Strongly, 411
- model
  - $\beta$ -model, 721
  - non-standard, 217, 472
  - standard, 472
  - structure, 202
- modulus, 62, 650
- morphism
  - Tree, 677
- Muchnik
  - degree, 298
  - reduction, 297
- nickname
  - well-order, 712
- node, 152
  - branching, 152
  - extensible, 154
  - leaf, 152
  - successor, 152
- non-cappable
  - degree, 321

- non-standard
  - Ordinal, 715
  - integer, 217, 472
  - model, 217, 472
- normal
  - Shape, 665
- Oberwolfach
  - Random, 447
  - Test, 447
- on a cone, 294
- open
  - Baire, 177
  - class, 156
  - effective, 163
- operator
  - c.e., 294
  - invariant, 294
  - uniformly invariant, 294
- oracle, 50
  - machine, 52
- order
  - Kleene-Brouwer, 638
- ordered
  - induction, 514
- ordinal
  - Addition, 624
  - Church-Kleene, 633
  - computable, 634
  - computable relativized, 639
  - Constructive, 633
  - Constructive relativized, 639
  - Countable, 629
  - Finished, 616
  - Kleene, 633
  - Limit, 615
  - Multiplication, 624
  - Non-standard, 715
  - Recursion transfinite, 627
  - Recursively inaccessible, 704
  - Successor, 615
  - Transfinite induction, 625
  - Uncountable, 629
- PA
  - degree, 172
  - theory, 199
- pair
  - exact, 332
- parallelization
  - problem, 549
- parameter, 467
  - formula, 474
- parametric
  - formula, 201
- partial order
  - embedding, 334
- pass
  - Difference test, 423
  - Martin-Löf test, 400
  - Solovay test, 401
- path
  - binary tree, 152
  - f-tree, 138
  - tree (Baire space), 176
- Peano
  - arithmetic, 198
- perfect
  - class, 161
  - tree, 161
- permitting
  - method, 300
- pigeonhole principle, 559
- positive
  - lower density, 426
  - upper density, 426
- Posner
  - trick, 313
- Post
  - correspondence problem, 290
  - problem, 289
- power of the continuum, 22
- pre-homogeneous
  - set, 566
- pre-order, 55
- predicate, 35
  - $\Sigma_n^0$ , 80
  - general recursive, 115
  - primitive recursive, 115
- prefix-free
  - Minimal, 394
  - machine, 365
- prefix-free)
  - set, 365

- prenex (form), 197
- preservation
  - arithmetic hierarchy, 284
  - property, 543
  - strong, 597
- primitive recursive
  - function, 98
  - predicate, 115
  - relativized function, 504
- principle
  - Lesser Limited of Omniscience, 549
  - Limited of Omniscience, 549
  - of cohesiveness, 591
  - pigeonhole, 559
- problem, 539
  - $\Pi_2^1$ , 539
  - computably true, 547
  - cylinder, 553
  - domino, 290
  - halting, 45
  - Hilbert, 291
  - mass, 297
  - parallelization, 549
  - Post, 289
  - Post correspondence, 290
  - product, 553
  - satisfaction, 540
  - uniformly true, 548
- product
  - problem, 553
- program
  - clean, 112
  - goto, 107
  - Hilbert, 534
  - structured, 105
  - universal, 36, 125
- programming diagram, 107
- promptly simple, 322
- property
  - Semantics, 91
  - Syntax, 91
  - Borel-Lebesgue, 158
  - Heine-Borel, 158
  - of Baire, 255
  - of preservation, 543
  - of strength, 64
  - of use, 54
  - of weakness, 64, 542
- provably
  - computable, 502
  - total, 502
- provably total function, 502
- Q, 468
- question
  - $\Pi$ -merging, 286
  - compact, 285
  - forcing, 279, 282, 283
  - Sacks, 294
- r.e.
  - set, 43
- RAM (machine), 104
- random
  - Chaitin/Levin, 367
  - Difference, 423
  - Martin-Löf relativized, 416
  - Oberwolfach, 447
  - Martin-Löf, 400
- RCA<sub>0</sub>, 481
- recursion, 35
  - Transfinite, 627, 708
  - primitive, 98
  - transfinite, 498
- recursive
  - function, 98
  - general function, 98
- recursively
  - Inaccessible, 704
  - enumerable, 43
- reduction
  - $\omega$ -reduction, 540
  - Hyperarithmetical, 682
  - Many-one, 87
  - Solovay, 370
  - computable, 545
  - game, 551
  - Medvedev, 297
  - Muchnik, 297
  - truth-table, 141
  - Turing, 55
  - Weihrauch, 547

- register machine, 104
- relation
  - forcing, 253, 269, 270
- relativization, 52
- relativized
  - Normal form, 666
- requirement, 64, 228
- Robinson
  - arithmetic, 198
- Sacks
  - density theorem, 347
  - forcing, 273
  - preservation strategy, 312
- satisfaction
  - $\Delta_2^0$  Approximation, 445
  - problem, 540
- scheme
  - $\text{BC}\Sigma_n^0/\text{BC}\Pi_n^0$ , 522
  - $\text{B}\Sigma_n^0/\text{B}\Pi_n^0$ , 510
  - $\Delta_1^0$  comprehension, 479
  - $\text{I}\Delta_0^0$ , 508
  - $\text{I}\Delta_n^0$ , 515
  - $\text{I}\Sigma_n^0/\text{I}\Pi_n^0$ , 510
  - $\text{I}_{\text{open}}$ , 505
  - $\text{L}\Sigma_n^0/\text{L}\Pi_n^0$ , 512
  - $\Sigma_1^0$  induction, 480
  - Comprehension, 708
  - Transfinite recursion, 708
  - bounded comprehension, 522
  - collection, 510
  - comprehension, 468
  - induction, 469
  - minimum, 512
  - ordered induction, 514
  - transfinite recursion, 498
- Scott
  - ideal, 488
- second-order
  - arithmetic, 218, 226, 465
  - formula, 226, 467
  - term, 466
- segment
  - final, 337
  - initial, 337
- semantics
  - full models, 471
  - standard, 471
- semilattice, 330
  - upper, 330
- sequence
  - Catalan, 601
  - Cauchy, 476
  - finite, 49
  - of bits, 22
  - uniformly computable, 44
- set
  - 1-generic, 242
  - $X$ -c.e., 51
  - $X$ -computable, 51
  - $X$ -computably enumerable, 51
  - $\Delta_1^0$ , 85
  - $\Delta_n^0$ , 81
  - $\Delta_n^0(X)$ , 86
  - $\Pi_n^0(X)$ , 86
  - $\Sigma_1^0$ , 84
  - $\Sigma_n^0$ , 79
  - $\Sigma_n^0(X)$ , 86
  - $n$ -generic, 246
  - 1-random, 420
  - 2-random, 419
  - $\Delta_1^1$ , 662
  - $\Pi_n^0$ -complet, 88
  - $\Pi_n^0(X)$ -complet, 88
  - $\Pi_1^1$ , 662
  - $\Sigma_n^0$ -complet, 88
  - $\Sigma_n^0(X)$ -complet, 88
  - $\Sigma_1^1$ , 662
  - $K$ -trivial, 374
  - $n$ -random, 420
  - Hyperarithmetic, 641
  - Minimal, 394
  - minimal prefix-free set, 394
  - omputably independent, 334
  - Simple, 374
  - Transitive, 616
  - Well-ordered, 620
  - c.e., 43
  - c.e. maximal, 47
  - c.e. maximum, 148
  - coded, 520

- cohesive, 591
- complete/incomplete, 59
- computable, 34
- computably dominated, 137
- computably enumerable, 43
- computably inseparable, 46
- countable, 16
- definable, 526
- dense, 229, 267
- dense below  $\sigma$ , 251
- DNC<sub>2</sub>, 172
- effectively immune, 129
- equipotent, 13
- generalized low <sub>$n$</sub> , 263
- generic, 230
- high, 72
- homogeneous, 558
- hyperimmune, 129
- immune, 128
- index, 91
- infinite, 521
- limit-computable, 59
- low, 71
- MLR, 400, 403
- of integers, 22, 24
- positive density, 426
- pre-homogeneous, 566
- prefix-free, 365
- promptly simple, 322
- r.e., 43
- recursive, 35
- recursively enumerable, 43
- subpotent, 14
- triadic of Cantor, 24
- uncountable, 16
- weakly 2-random, 411
- weakly 1-generic, 236, 238
- weakly  $n$ -generic, 238
- well ordered, 498
- set  $\Pi_n^0$ , 79
- simple
  - Set, 374
- singleton
  - $\Pi_1^1$ , 701
  - $\Pi_1^0$ , 606
  - $\Pi_2^0$ , 606
- SMN (theorem), 37
- Solovay
  - Function, 453
  - Reduction, 370
  - Test, 401
- solution
  - problem, 539
- space
  - Baire, 176
  - Cantor, 22, 155
- spectrum (degree), 173
- split ( $\Gamma$ -split), 325
- split (f-tree), 325
- stable (function), 59
- standard
  - $\omega$ , 472, 501
  - integer, 472
  - model, 472
  - semantics, 471
- statement, 193
  - ADS, 540
  - COH, 591
  - CRT <sub>$k$</sub>  <sup>$n$</sup> , 560
  - J, 541
  - KL, 541
  - LLPO, 549
  - LPO, 549
  - RT <sub>$k$</sub>  <sup>$n$</sup> , 558
  - WKL, 488
  - WWKL, 492
- strategy
  - Sacks preservation, 312
  - tree, 318
- strength property, 64
- string
  - binary, 49
  - compatible, 49
  - extension, 49
  - incompatible, 49
- strong
  - avoidance of a cone, 600
- strong avoidance
  - of a cone, 600
- strong computable reduction, 553
- strong minimal cover, 329
- strong Weihrauch reduction, 553

- structure
  - $\omega$ -structure, 473
  - arithmetic, 200
  - full, 470
  - Henkin, 470
  - model, 202
  - underlying set, 201
- structured program, 105
- sub-f-tree, 138
- subpotence, 14
- successor
  - function, 99, 115
  - tree, 152
- sufficiently generic, 230, 268
- surjection, 13
- system
  - axiomatic (see theory), 198
  - of deduction, 194
- term
  - arithmetic, 192
  - closed, 192
  - first-order, 466
  - second-order, 466
- test
  - Difference, 423
  - Martin-Löf, 400
  - Martin-Löf relativized, 416
  - Oberwolfach, 447
  - Solovay, 401
- Theorem
  - Posner-Robinson, 248
- theorem
  - Arslanov, 134
  - Basis of Gandy, 687
  - Chaitin, 362
  - KC, 371
  - Kreisel basis, 690
  - Kučera/Gács, 406
  - Lebesgue density, 425
  - Post, 89
  - Rice, 92
  - Solovay, 375
  - basis, 166
  - Bolzano-Weierstrass, 487
  - Borel-Lebesgue, 490
  - Cantor-Bernstein, 15
  - Chinese Remainder, 210
  - computably dominated basis, 167
  - cone avoidance basis, 169
  - consistency, 204
  - determination of Martin, 221
  - finite Ramsey, 560
  - fixed point, 39
  - Friedberg's jump inversion, 249
  - Friedberg-Muchnik, 302
  - Gödel completeness, 205
  - Gödel-Rosser incompleteness, 214
  - Hindman, 497
  - incompleteness of Gödel, 213, 215
  - intermediate value, 483
  - Kleene, 39
  - Löwenheim-Skolem, 525
  - Liu, 583
  - low basis, 166
  - Martin's domination, 145
  - MRDP, 293
  - Ramsey, 558
  - Rice, 42
  - SMN, 37
- theory, 198
  - $ACA'_0$ , 496
  - $ACA^+_0$ , 496
  - $ACA_0$ , 485
  - $ATR_0$ , 498
  - $B\Sigma^0_n/B\Pi^0_n$ , 522
  - $B\Sigma^0_n/B\Pi^0_n$ , 510
  - $I\Delta^0_0$ , 508
  - $I\Delta^0_n$ , 515
  - $I\Sigma^0_n/III^0_n$ , 510
  - $I_{open}$ , 505
  - $L\Sigma^0_n/L\Pi^0_n$ , 512
  - $\Pi^1_1$ - $CA_0$ , 499
  - PRA, 535
  - $\Pi^1_1$  Comprehension Axiom, 499
  - $RCA_0$ , 481
  - Q, 199, 468, 504
  - $\Sigma_1$ -PA, 527
  - WKL<sub>0</sub>, 488
  - WWKL<sub>0</sub>, 492
  - $Z_2$ , 469

- $\text{ATR}_0$ , 708
  - $\Pi_1^1\text{-CA}_0$ , 709
  - Arithmetical Comprehension Axiom, 485
  - Arithmetical Transfinite Recursion, 498
  - categorical, 471
  - complete, 171, 205
  - consistent, 171, 200
  - PA, 199
  - Recursive Comprehension Axiom, 481
  - second-order arithmetic, 469
  - Weak König's Lemma, 488
  - Weak Weak König's Lemma, 492
  - Zermelo, 219
  - ZF, 223
  - ZFC, 218, 223
- thesis
  - Church-Turing, 102
- transfinite iteration
  - $\text{ATR}_0$ , 498
  - $\text{ATR}_0$ , 708
  - Hyperjump, 702
  - Relativized Turing jump, 651
  - Turing jump, 614
- transitive
  - Set, 616
- tree, 152, 176
  - Height, 637
  - Morphism, 677
  - binary, 152
  - c.e., 327
  - computable, 153
  - computably bounded, 179
  - f-tree, 138
  - finitely-branching, 176
  - leaf, 152
  - node, 152
  - path, 152, 176
  - perfect, 161
  - positive measure, 492
  - strategies, 318
- trick
  - Posner, 313
- truth
  - formula, 202
  - truth-table
    - degree, 141
    - reduction, 141
  - Turing
    - complete, 59
    - degree, 55
    - equivalence, 55
    - functional, 50
    - ideal, 481
    - jump, 58
    - machine, 100
    - reduction, 55
    - thesis, 102
  - Turing machine
    - universal, 36, 125
- uncountable
  - Ordinal, 629
  - set, 16
- underlying
  - set, 201
- uniformity, 44
- uniformly
  - $\Sigma_n^0$ , 83
  - computable, 60
  - true, 548
- universal
  - Machine, 356
  - class, 174
  - closure, 468
  - instance, 595
- upper
  - Density, 425
  - bound, 324
  - lattice, 330
  - semilattice, 330
- upper bound, 324
- use, 54
  - property, 54
- variable
  - bounded, 193
  - free, 193
- von Neumann's ordinal, 616

- weakly
  - 1-generic, 236
  - 1-generic relativized, 238
  - $n$ -generic, 238
- weakly 2-random, 411
- weakness property, 64, 542
- Weihrauch
  - reduction, 547
  - strong reduction, 553
- well order, 498
- well-founded
  - Order, 620
  - Tree, 637
- well-order, 620
  - Nickname, 712
- witness
  - Henkin, 206, 207
  - Weihrauch reduction, 547
- $WKL_0$ , 488
- $WWKL_0$ , 492
- $Z_2$ , 469
- ZF, 223
- ZFC, 218, 223