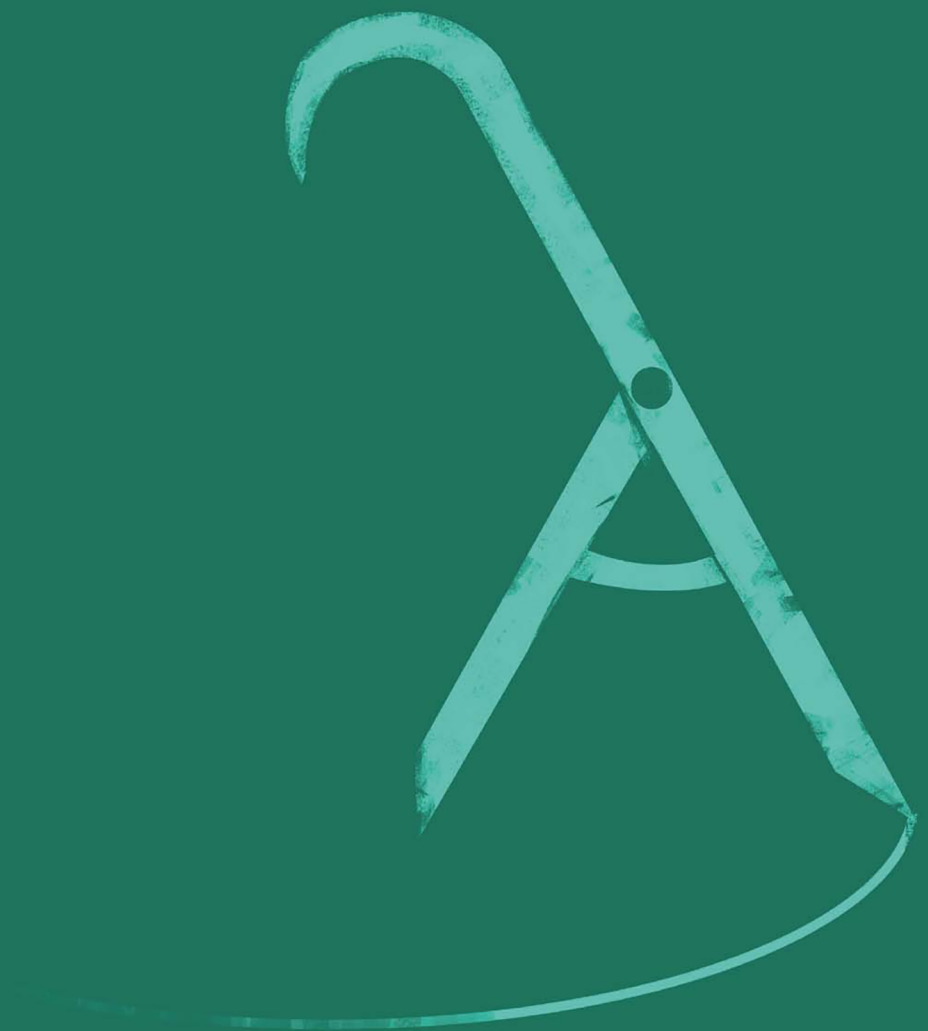
Andrew Bacon

A Philosophical Introduction to Higher-order Logics



ROUTLEDGE

A Philosophical Introduction to Higher-order Logics

This is the first comprehensive textbook on higher-order logic that is written specifically to introduce the subject matter to graduate students in philosophy. The book covers both the formal aspects of higher-order languages—their model theory and proof theory, the theory of λ-abstraction and its generalizations—and their philosophical applications, especially to the topics of modality and propositional granularity. The book has a strong focus on non-extensional higher-order logics, making it more appropriate for foundational metaphysics than other introductions to the subject from computer science, mathematics, and linguistics.

A Philosophical Introduction to Higher-Order Logics assumes only that readers have a basic knowledge of first-order logic. With an emphasis on exercises, it can be used as a textbook though is also ideal for self-study.

Author Andrew Bacon organizes the book's 18 chapters around four main parts:

I. Typed Language
II. Higher-Order Languages
III. General Higher-Order Languages
IV. Higher-Order Model Theory

In addition, two appendices cover the Curry-Howard isomorphism and its applications for modeling propositional structure. Each chapter includes exercises that move from easier to more difficult, strategically placed throughout the chapter, and concludes with an annotated suggested reading list providing graduate students with most valuable additional resources.

Key Features:

- Is the first comprehensive introduction to higher-order logic as a grounding for addressing problems in metaphysics
- Introduces the basic formal tools that are needed to theorize in, and model, higher-order languages
- Offers an abundance of
  - Simple exercises throughout the book, serving as comprehension checks on basic concepts and definitions
  - More difficult exercises designed to facilitate long-term learning
- Contains annotated sections on further reading, pointing the reader to related literature, learning resources, and historical context

Andrew Bacon is Associate Professor at the University of Southern California. He is the author of *Vagueness and Thought* (Oxford UP, 2018) and has written numerous articles applying logical methods to topics in metaphysics, epistemology, and language.

A Philosophical Introduction to Higher-order Logics

Andrew Bacon

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

Designed cover image: © Nathan J. Hicks

First published 2024
by Routledge
605 Third Avenue, New York, NY 10158

and by Routledge
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

Routledge is an imprint of the Taylor & Francis Group, an informa business

ISBN: 978-0-367-48302-9 (hbk)
ISBN: 978-0-367-48301-2 (pbk)
ISBN: 978-1-003-03918-1 (ebk)

DOI: 10.4324/9781003039181

Typeset in Times New Roman
by codeMantra

For Ceridwen

Contents

Nomenclature

Some common symbols:

H, C, etc.	Names for particular higher-order logics, the minimal system H, Classicism, etc.
$\mathbf{L}, \mathbf{L}', \ldots$	Name for an arbitrary logic
M, N, P, Q, a, b, c	Terms of arbitrary type, lower case reserved for terms in argument position
A, B, C	Terms of type t
X, Y, Z, x, y, z, w	Variables, lower case reserved for variables appearing exclusively in argument position
$\sigma, \tau, \rho, \ldots$	Types
Σ	A name for a signature
Λ	The logical signature of higher-order logic
$\mathcal{L}(\Sigma)$	The full λ-language in the signature Σ
$\mathcal{J}(\Sigma)$	An arbitrary λ-language in the signature Σ
$M \sim_{\beta\eta} N$	M and N are $\beta\eta$ equivalent
$CL[\Sigma, \{B, C, K\}]$, etc.	The combinatory language in signature Σ with combinators B, C and K
$\mathbf{ND}[PCW]$, etc.	The natural deduction Curry system containing the rules P, C, and W
$A, B, C \ldots, A^\sigma, B^\sigma, C^\sigma \ldots$	Names for applicative structures, names for domains of type σ
$\tilde{f}$	The counterpart of the function f in an applicative structure
$\mathbf{M}, \mathbf{N}, \ldots$	Models of higher-order languages
$g, h, \ldots$	Variables assignments
$[\![M]\!]^g$	The denotation of a term M relative to an assignment in a given model
$R, S, T, \ldots, R^\sigma, S^\sigma, T^\sigma \ldots$	Names for logical relations or Kripke logical relations, names for logical or Kripke logical relation at a given type

Some common abbreviations:

$M : \sigma$	'M is a term of type σ', 'M, of type σ,'
$MN_1N_2 \dots N_k$	$(\dots((MN_1)N_2)\dots N_k)$
$\lambda x_1 x_2 \dots x_n.M$	$\lambda x_1.(\lambda x_2.(\dots \lambda x_n.M))\dots)$
$\sigma_1 \to \sigma_2 \to \dots \to \sigma_n \to \tau$	$(\sigma_1 \to (\sigma_2 \to (\dots \to (\sigma_n \to \tau)\dots)))$
$\forall_\sigma x.A$	$\forall_\sigma(\lambda x.A)$
$A \wedge B$, $a =_\sigma b$, etc	$((\wedge A)B)$, $((=_\sigma a)b)$
$\bar{x}$, $\bar{a}$, $\bar{\sigma}$	$x_1, \dots, x_n$, $a_1, \dots, a_n$, $\sigma_1, \dots, \sigma_n$
$(\bar{\sigma} \to \tau)$	$\sigma_1 \to \sigma_2 \to \dots \to \sigma_n \to \tau$
$\lambda\bar{x}.M$, $M\bar{x}$	$\lambda x_1 \dots \lambda x_n.M$, $Mx_1 \dots x_n$
$\bar{x} : \bar{\sigma}$:	$x_1 : \sigma$, $x_2 : \sigma_2 \dots x_n : \sigma_n$
$\forall\bar{x}.A$	$\forall_{\sigma_1} x_1 \dots \forall_{\sigma_n} x_n.A$ where $\bar{x} : \bar{\sigma}$
$(MN)^n_m$	$\lambda\bar{x}\bar{y}.M\bar{x}(N\bar{y})$ where $\bar{x} = x_1 \dots x_m$, $\bar{y} = y_1 \dots y_n$, $\bar{x} : \bar{\sigma}, \bar{y} : \bar{\rho}, N : \bar{\rho} \to \tau, M : \bar{\sigma} \to \tau \to \theta$
$\wedge_{\bar{\sigma}\to t}$, $\neg_{\bar{\sigma}\to t}$, etc.	$\lambda XY\bar{x}\bar{y}(X\bar{x} \wedge Y\bar{y})$, $\lambda X\bar{y}.\neg(X\bar{y})$, etc. where $\bar{x}, \bar{y} : \bar{\sigma}$
I^σ	$\lambda x.x$ where $x : \sigma$
$S^{\sigma\tau\rho}$	$\lambda XYz.Xz(Yz)$ where $X : \sigma \to \tau \to \rho$, $Y : \sigma \to \tau$, $z : \sigma$
$K^{\sigma\tau}$	$\lambda xy.x$ where $x : \sigma, y : \tau$
$B^{\sigma\tau\rho}$, $B'^{\sigma\tau\rho}$	$\lambda XYz.X(Yz)$, $\lambda YXz.X(Yz)$ where $Y : \sigma \to \tau, X : \tau \to \rho, z : \sigma$
$C^{\sigma\tau\rho}$	$\lambda Xyz.Xzy$ where $X : \sigma \to \tau \to \rho, z : \sigma, y : \tau$
$W^{\sigma\tau}$	$\lambda Xy.Xyy$ where $X : \sigma \to \sigma \to \tau, y : \sigma$
$\Box_\top$	$\lambda p.p =_t \top$

Preface

There are some ways in which this book resembles a textbook, and other ways in which it resembles a monograph. Like a textbook it aims to impart to the reader certain logical tools that I believe to have a great number of applications in philosophy, specifically metaphysics. This represents a majority of the book. On the other hand, I have made no effort to be comprehensive in my coverage of those applications: I have simply taken two topics—modal metaphysics (chapters 7 and 8), and metaphysical structure (chapters 11–13)—that have personally captivated my interest, and to which the tools presented in this book have helped me get things straight in my own head. These cases, I hope, illustrate the power of higher-order logic both as a language for formulating important claims in metaphysics, and as a framework for investigating those questions. But I have followed my own inclinations in deciding what to explore, and this part of the book is in no way representative of the full variety of possible applications, or indeed, existing applications of these tools in the burgeoning literature on higher-order metaphysics. Some of these omissions are mentioned briefly at the end of the introductory chapter.[1] My hope is that the reader will be able to take the apparatus in this book and apply them to whatever questions captivate their interest.

I was motivated to write this book after struggling myself to find the right tools for my purposes in existing philosophical texts. I found, increasingly, that I was reinventing wheels that had previously been invented by computer scientists with completely different applications in mind. These include several logical tools described in this book; for instance, substructural type theory, the concept of a logical relation, and various ideas from category theory. However, because these wheels have been designed for different cars (as it were), and the expository texts written for a different audience, there is no resource which a philosopher can simply consult in order to learn about them in a way that transparently relates them to philosophical concerns. My hope is this book will fill that lacuna.

A couple of brief remarks on the title of the book are in order. Perhaps a more accurate (but less catchy) name would have been *A Philosophical Introduction to Higher-order Logics and λ-Calculi*. A substantial portion of the book is devoted to simply typed λ-languages: a very general class of languages that includes many logical languages, such as propositional, first-order and higher-order logic, and even non-logical languages such as programming languages. Pedagogically, the relationship between the λ-calculus and higher-order logic is a bit like the relationship between propositional logic and first-order logic: very few philosophically interesting theories can be formulated in propositional logic alone, but one needs to become reasonably fluent in it before learning first-order logic. I believe the λ-calculus stands in a similar relationship to higher-order logic, and should be studied first.

The other remark regarding the title concerns the use of a plural noun. There is an important difference between propositional and first-order logic on the one hand, and higher-order logics on the other. One cannot consistently extend classical propositional logic with further logical axiom schemas, and the only logical principles one can consistently add to first-order logic with identity make fairly uninteresting statements about how many different things there are. By contrast there are many different ways to consistently extend classical higher-order logic with further purely logical axioms.[2] Although higher-order logics are neutral on questions of mathematical *ontology*—they are formulated without reference to primitive mathematical notions, like number and set membership—some of these logical axioms seem mathematical in nature. There are purely logical statements you can make that settle the continuum hypothesis in the sense that they would imply the continuum hypothesis if there *were* any sets.[3] Similarly, while higher-order logics are neutral about the ontology of propositions, conceived of as certain kinds of abstract individuals, one can formulate purely logical statements that make structural claims about reality: claims that imply that propositions would be structured, if there were any abstract entities that bore an appropriately close relationship to reality (statements like these are explored in chapters 11–13). One can similarly make purely logical statements that correspond in this way to the existence or non-existence of maximally specific propositions, 'possible worlds', or statements that imply that modal reality admits a lot of different possibilities, or not very many at all, or statements that contradict the **S5** axiom for a suitably broad kind of necessity (statements like these are explored in chapter 8). The line between logic, mathematics and metaphysics becomes somewhat blurry: logic can constrain mathematical and metaphysical theorizing in new and non-obvious ways. The wider point here is that there are a great many interesting higher-order logics, and so the subject should be studied with this in mind: it is not the study of a single system, like first-order logic, but rather of a class of interestingly different systems. In this sense the study of higher-order logic is much more like the study of modal logic.

Part I of the book concerns typed languages. Most languages we are familiar with, such as the language of propositional logic, first-order logic, propositional modal logic, and so on, are implicitly typed languages. There are rules about which expressions can be combined with which — one can apply a predicate to an individual constant, but not to a sentence or another predicate, whereas one can apply operators to sentences, but not to predicates, individual constants, other operators, and so on. We also know how to introduce new devices that behave in grammatically novel ways: for instance we could easily add to first-order logic predicate modifiers, that combine with predicates to make other predicates; then we could introduce predicate modifier modifiers that combine with predicate modifiers to make new predicate modifiers, and so on. Type theory systematizes these rules and a typed language is simply a language which fits this schema (chapter 1). There is a particularly important device, λ, that guarantees that there is a predicate corresponding to any open formula parametrized by a variable x, and which makes similar guarantees for expressions of other types. Chapters 2 and 3 introduce the simply typed λ-calculus: a theory concerning typed languages containing the λ device, governed by two central principles β and η. The latter chapter also outlines relations between the λ-calculus and a certain variable-free alternative to it called combinatory logic.

Part II of the book concerns a certain kind of typed language that contains, for each grammatical type, devices that stand to that type as the first-order order quantifiers stand to individual terms. They can bind variables of that type in the same way that the first-order quantifiers bind individual variables, and they are subject to logical laws that are

completely parallel to the first-order universal and existential quantifiers. We thus call them 'higher-order quantifiers'. Higher-order quantifiers let us express generality in sentence position, predicate position, operator position, and so on, in the same way that the first-order quantifiers express generality into name position. Chapter 4 outlines these quantificational devices in some detail, and in chapter 5 the key concept of a higher-order logic is introduced. Chapter 6 applies higher-order logic to current theorizing about the granularity of propositions, properties and relations. Chapter 7 and 8 are an extended investigation of the higher-order logic Classicism, introduced in chapter 6, in application to questions in modal metaphysics: In higher-order logic it is possible to define the analogue in reality of an operator expression with a normal modal logic, and many other notions from modal metaphysics—'necessity in the highest degree', entailment, possible world—have a claim to being reducible to pure logic.

In part III of the book we consider weakening the λ formalism. The full λ-calculus is not ontologically neutral: it contains binary predicate expressions that are converses of other predicates, committing us in the background logic to the existence of converse relations. This is not a first-order ontological commitment but a higher-order one. λ-languages contain many other types of expressions that have ontologically contentious implications. Chapters 9 and 10 explore general λ-calculi that do not have these expressions. Chapters 11–13 consist of extended applications of the machinery to the question of the structure of propositions, properties and relations. Various ideas about the structure of reality discussed in the philosophical literature are formalized in higher-order languages, and some limitative results are presented.

Part IV concerns the model theory of higher-order logics. Chapters 14 and 15 introduce the notion of a model for an arbitrary λ-language and the logical language of higher-order logic respectively. Chapter 16 introduces the key notion of a *logical relation*, an important model theoretic tool which can be used to construct partial quotients of models and partial homomorphisms between models, as well as to establish definability and undefinability results. Chapter 17 introduces concepts from category theory that are useful in the study of type theory, and explores three concrete categories – the category of sets, the category of modalized sets and the category of M-sets – which have particular applications to the study of higher-order logic. Chapter 18 investigates the model theory of Classicism in particular and outlines some model theoretic constructions that can be wielded to settle the consistency of several higher-order logics discussed in chapter 8. The book ends with two appendices on topics relating to part III that are slightly off the main track, but which connect the syntax and semantics of λ-languages with (seemingly disparate) research on non-classical propositional logics.

Many people have made this book possible. My own interest in higher-order logic extends as long as my education in philosophy: it began in my first year as an undergraduate, where I learned about Frege's philosophy of mathematics from Ian Rumfitt, a topic that I continued to pursue under the supervision of Gabriel Uzquiano as a BPhil student. These early teachers instigated my initial fascination with the topic, and helped me appreciate the power of higher-order languages for philosophical theorizing. The catalyst for my present interest in higher-order logic, however, has been from conversations with Mike Caie, Cian Dorr, Peter Fritz, Jeremy Goodman and Harvey Lederman, taking place between 2015 to 2023, concerning applications of higher-order logic to metaphysics. I have learned a great deal from these conversations, and without this stimulus I would not have found my way into the subject to the extent that I presently have.

During the writing of this book, I benefited greatly from a number of people who spotted various mistakes in early drafts of the book, including Daniel Hoek, David Ripley, Chris Scambler, Lingzhi Shi, Shawn Standefer, Rohan Sud, Juhani Yli-Vakkuri, Arthur Wu and especially Cian Dorr for checking over the final chapter, and Helen Bacon for reading through a number of chapters for typos. I am also grateful to an anonymous referee for Routledge who, among other things, made some very helpful structural suggestions about the organization of the book. I owe a special debt of gratitude to Jin Zeng, who read an early draft of the book and helped me fix many important errors and inconsistencies.

Endnotes

1. The reader interested in the broader field of higher-order metaphysics might wish to consult the recent anthology Fritz and Jones (forthcoming).
2. 'Purely logical' here means stateable using only logical words (or logical words with the addition of schematic constants) – in this case the truth-functional connectives and the first and higher-order quantifiers.
3. In higher-order logic one can formulate a general claim that says that any relation satisfying the principles of Zermelo-Fraenkle set theory when taking the place of the membership relation is one in which the continuum hypothesis holds. Given the non-logical assumption that $\in$ — i.e. *set membership* — satisfies the axioms of Zermelo-Fraenkle set theory, we can infer that the continuum hypothesis is true. See Shapiro (1991) p105 for another formulation of a CH-like principle of higher-order logic.

Introduction

If you are reading this book, you have likely already heard something about its subject matter. Perhaps you have read that higher-order logics are generalizations of first-order logic, or perhaps you have come across some of their applications in philosophy or other disciplines. The aim of this book is to introduce the reader to higher-order languages, their proof theory, model theory and some of their applications in philosophy. However, there is also a distinctively higher-order way of *thinking* about philosophical questions, which is harder to impart, and which goes beyond simple proficiency with these logical tools: it is possible to become well-versed in the higher-order symbolism without having a proper grasp on what the symbolism *means*, or why we are studying it. This informal chapter is intended to introduce readers to typed languages and higher-order generalizations in a casual and imprecise way, give a sense of why they are useful as well as to introduce some of the main themes of the book along the way.

What is a typed language? (Section 0.1)

What are higher-order generalizations? (Section 0.2 and Section 0.3)

What is abstraction? (Section 0.4)

What is the difference between higher-order generalizations and first-order generalizations over properties? (Section 0.5)

Why are higher-order generalizations useful for doing philosophy? (Section 0.6)

What is the point of providing a (set-theoretic) model theory for a higher-order language? (Section 0.7)

Can we translate higher-order sentences into ordinary English? (Section 0.8)

How should one read this book? (Section 0.9)

0.1 Typed languages

When a student is first introduced to logic, they will encounter various different *types* of logical expressions. Sentences are fundamental to logic because they are used to state things and are the premises and conclusions of arguments. But sentences have logical structure—they are composed of subsentential expressions—in virtue of which arguments are valid or invalid. In propositional languages, we encounter devices that can combine with sentences to make other complex sentences, such as the truth-functional connectives, $\wedge, \vee, \neg$, and any other connectives we might add to a propositional language, such as modal or counterfactual connectives. In first-order languages we encounter, in addition to these

DOI: 10.4324/9781003039181-1

devices, names—expressions used to *name* things—and predicates that may be combined with names to make sentences. It is possible to think of the universal and existential quantifiers of first-order logic as something that combines with an ordinary predicate to make a sentence: the simplest sorts of quantified statements involve a quantifier combining with a predicate to make a universal or existential statement (e.g. 'somethings smells' = 'something' + 'smells'). On this analysis, quantifiers are *higher-order* predicates.

But why stop here? Just as connectives and first-order predicates can have different arities—$\wedge$ is binary, because it needs two sentences to make a sentence, whereas $\neg$ is unary (it is an 'operator') because it needs only one—we could consider languages that have higher-order predicates that combine with multiple first-order predicates to make sentences. Logicians have studied extensions of first-order logic with binary quantifiers like *most*, and *the*, combining with two predicates, *F* and *G*, to make a sentence (e.g. *most Fs are G*). We can also imagine expressions that are mixed between connectives and predicates—*connecticates*—that take a name and a sentence to make a sentence. When studying the logic of knowledge for multiple agents it's natural to theorize with a primitive device meaning 'knows that' taking a name in the first argument, and a sentence in the second.[1] From here the possibilities are boundless. One can similarly imagine adding devices that combine with connectives to make sentences, or with predicates to make predicates, or with any of the devices described above to make any other device described above.

The languages described above are all instances of *typed* languages, which is the topic of part I of this book. Expressions of these languages are divided up into different grammatical categories, *types*, and there are rules concerning how expressions of different types can be combined—you cannot, for instance, combine a name with a binary connective. Almost all of the formal languages you are likely to have encountered already will be typed languages—from the simplest, the language of propositional logic, to any of the richer languages described above. Natural languages are also typed languages—expressions of natural languages belong to different grammatical categories and there are rules about how these expressions can be combined grammatically. But things were not always this way. Two of the founders of modern logic, Gottlob Frege and Giuseppe Peano, took very different approaches to logical languages: while Frege's system was rigidly typed, Peano had no grammatical distinction like that between predicates, sentences and names—one could put any expression next to any other.[2] For a while both traditions developed in parallel, with authors like Russell, Hilbert and his colleagues, and later Church following Frege and theorizing in typed languages, and Schönfinkel, Curry, and Fitch following Peano. But ultimately the former approach won out, and research in logic today is primarily conducted in a typed languages.[3]

While the typed framework is, in this general sense, quite orthodox, the most familiar typed languages (first-order languages) do not make use of expressions that go very far up the hierarchy of typed expressions. Yet this extra generality seems to be very useful for philosophical theorizing. We can illustrate this with a couple of examples. In the philosophy of modality, one typically distinguishes operator expressions that are used to express modality, like such as 'it's necessary that', 'it *has* to be true that' and so on, from those that do not, such as 'it's not the case that' or 'John said that'. But this is not just a distinction in language—many philosophers believe there is a corresponding distinction in reality. In order to express these metaphysical distinctions in the object language it is therefore necessary to have a device, 'Nec', that combines with an operator to form a sentence. If we had operators $\neg$ and $\Box$ for negation and a particular kind of necessity we could form claims like $\neg(\mathrm{Nec}\,\neg)$ and

$\mathrm{Nec}\,\Box$ to say that the latter but not the former is a kind of necessity. Another example: metaphysicians will often talk about what vocabulary is fundamental or 'joint carving' (see also 'perfectly natural' in Lewis (1983) and 'structural' in Sider (2011)). It would be prejudiced to assume prior to inquiry that, say, only first-order predicates could be fundamental or not.[4] These distinctions must be applicable to words of many different grammatical categories: perhaps the predicates of physics (e.g. 'is an electron') are fundamental, or the tense operators are fundamental, or the quantifiers. But, as before, this distinction between expressions in language is to be explained in terms of more basic distinctions in reality, and to express these we need higher-order predicates. Let E be a first-order predicate meaning 'is an electron'. We need a higher-order predicate $\mathrm{Fun}_{\mathrm{pred}}$ that combines with a first-order predicate to make a sentence in order to state the relevant claim about fundamentality, $\mathrm{Fun}_{\mathrm{pred}}\,E$. In order to make an analogous fundamentality claim about necessity, $\mathrm{Fun}_{\mathrm{op}}\,\Box$ we need a higher-order predicate $\mathrm{Fun}_{\mathrm{op}}$ that can combine with an operator to make a sentence. And once we have these higher-order fundamentality predicates we can ask whether they themselves are fundamental, and the analogous questions about reality will require higher-order predicates that combine with predicates of the type of $\mathrm{Fun}_{\mathrm{pred}}$ and $\mathrm{Fun}_{\mathrm{op}}$ to make sentences. Further questions can be asked about the fundamentality of these new predicates, and so higher-order predicates of 'arbitrary order' are needed.[5]

0.2 Generalizations

Apart from the sorts of expressions they admit, the chief difference between propositional languages and first-order languages—the feature that makes first-order languages more *expressive*—is that in the latter one can express generalizations. Specifically, one can express generality in name position. This is a book about higher-order languages, the things we can say in them, and how to reason in and about them. The chief expressive advantage of higher-order languages over first-order languages is simply that they can express more kinds of generalizations: they can express generality in the position of sentences, predicates, connectives, and so on. They can express generality in any grammatical position whatsoever.

In order to understand what this means, we will begin examining ordinary first-order generality in name position. Consider a simple subject-predicate sentence, such as

Socrates is wise.

This would be formalized as *Fa*. By using the first-order quantifiers we can express generality in the position that 'Socrates' occupies by replacing this name with a first-order variable and prefixing with the universal or existential quantifier.

$\forall x$, x is wise.

$\exists x$, x is wise.

For now, we adopt an informal hybrid of English and logical notation. In ordinary English these sentences mean, approximately, 'everything is wise' and 'something is wise' respectively.

The sense in which these claims express generality in name position is that they bear a special logical relationship to their *instances*. First we identify the position we are generalizing

into (the position of the name 'Socrates' above) and we form a sentence that has a gap in this position—i.e. that is variable in that position

. . . is wise

This defines a class of sentences parameterized by the different names we can substitute into the gap. We might call this a 'sentential function', for it defines a class of sentences parameterized by the name we plug into the gap. (Early practitioners of higher-order logic, who were not very careful about use and mention, would call this a 'propositional function', but they meant the same thing.) These are the instances of the generalization:

Socrates is wise, Plato is wise, Aristotle is wise, (. . .and so on)

The logical relationship is this. From the universal claim, any instance can be inferred: from the claim that $\forall x$, x is wise ('everything is wise') one can infer that Socrates is wise, for example. This is often called the 'universal elimination rule'. Conversely, if you can prove an "arbitrary" instance—let's say you can prove that Socrates is wise without making assumptions specific to Socrates—then that reasoning would apply just as well to any other individual, and we may infer the universal claim $\forall x$, x is wise. This is often called the 'universal introduction rule'. A similar pair of elimination and introduction rules characterize the existential quantifier by its relationship to its instances.

The universal and existential sentences do not mention particular individuals by name, but rather generalize from sentences naming particular individuals—'Socrates is wise', 'Aristotle is wise', and so on. It is clear enough, then, what it means to say that the first-order quantifiers express generality with respect to name position, for it is names—not predicates, sentences, or what have you—that vary between the instances.

There are several ways to get a handle on the quantifiers of first-order logic. An approximation, utilized above, is to paraphrase them into English using quantificational idioms like 'everything' and 'something'. But these are imperfect because we often—indeed, some would argue *always*—understand these quantifiers as restricted in some way or other. These restricted uses of the words 'something' and 'everything' are not inferentially connected to their instances in the right way to express the universal and existential generalizations of first-order logic. The professor can say, in an introductory logic course, 'everything will be on the exam' and say something true; it does not therefore follow that Gödel's incompleteness theorems will be on the exam—the professors generalization was restricted to topics covered in this introductory logic class. So the restricted universal does not imply all of its instances—it is not completely *general*.

By contrast, we will understand the universal generalization of first-order logic as being completely general—from a true universal generalization all instances must follow. Another way into understanding generalizations is by their logical relationship to their instances. Someone who already understands the quantifier-free fragment of a first-order language—the sentences that do not contain the symbols '$\forall$' and '$\exists$'—may extend their understanding to the fully quantified language using the rules we described above. These rules in fact *pin down* the quantifiers up to logical equivalence. We'll record this with the following informal theorem (we will have the tools to offer a more precise version of this later in this book).

Theorem 0.1 (Harris's theorem for name generalizations)**.** *Suppose* $\forall_1$ *and* $\forall_2$ *are two generalizing devices, in the sense that they both satisfy the universal introduction and elimination*

principles. Consider a sentence of the form . . . x . . ., where the variable x takes the position that a name could ordinarily occupy. Then '$\forall_1 x \ldots x \ldots$' and '$\forall_2 x \ldots x \ldots$' are logically equivalent in the sense that they can be inferred from one another.

We can illustrate the style of argument with our example. Given $\forall_1 x$, *x* is wise, we can infer that Socrates is wise, using the fact that we can infer any instance from a general claim (and that $\forall_1$ expresses general claims). Since we showed that Socrates is wise from assumptions that did not depend on any feature specific to Socrates—it would have worked equally well for an *arbitrary* individual—we can infer $\forall_2 x$, *x* is wise. The inference from $\forall_2 x$, *x* is wise to $\forall_1 x$, *x* is wise can be demonstrated in exactly the same way.[6] A parallel argument is available for the existential quantifiers.

Remark 0.1. Some philosophers maintain that there is no use of the word 'everything' and 'something' for which the move from 'everything is *F*' to '*a* is *F*', and from '*a* is *F*' to 'something is *F*' are legitimate. Sometimes this stems from the view that all uses of 'everything' and 'something' are restricted—no matter what we do to prime the hearer to a wide reading of these terms, there is always a wider reading available.[7] In this case, these inferences fail in the same way that we cannot move from 'everything will be on the exam' to 'Gödel's incompleteness theorems will be on the exam' in our earlier example. Others maintain there is a widest use of 'everything' and 'something', but that we can make true assertions involving fictional names, like 'the ancient Greeks worshiped Zeus' without being able to infer 'the ancient Greeks worshiped something'. These philosophers will instead adopt a weaker-than-classical logic—a 'free logic'—for reasoning about the quantificational words.

These views are usually being motivated by the behaviour of the quantificational idioms of natural language. By contrast, we have characterized the symbols $\forall$ and $\exists$ by their inferential role, leaving it open how exactly they relate to the quantificational idioms of natural language. (Of course, practitioners of free logic will often use the same symbols, $\forall$ and $\exists$, in the formal study of these logics, but this is a purely notational point.) It is, of course, a substantive theoretical posit that *any* device satisfies this logical role. Nonetheless, it is a posit that earns its keep: the ability to form unrestricted generalizations is incredibly useful for theoretical disciplines, such as logic and metaphysics, that require a certain level of generality rarely needed in ordinary conversation. Furthermore, nothing the free logicians say contradict this posit: it is one thing to claim, for instance, that the inference from 'the ancient Greeks worshiped Zeus' to 'the ancient Greeks worshiped something' is no good, it is another to claim that there is no generalizing device we could substitute for 'something' which would make the inference OK. On the contrary, it is very hard to say something that contradicts our theoretical posit: the claim 'all uses of 'everything' are restricted' is no good, for to be restricted is usually spelled out in terms of unrestricted quantification. To be restricted is for there to be something (in the unrestricted sense) not in the range of that quantifier. One cannot use a restricted quantifier to spell the notion out either, for every restricted quantifier is unrestricted by its own lights.[8]

0.3 Higher-order generalizations

We have explained how the first-order universal and existential quantifiers, $\forall x$ and $\exists x$, express generality in name position. We might, by analogy, introduce a pair of devices—predicate quantifiers—that let us express generality in predicate position. This time we express generality in the position that 'is wise' occupies in 'Socrates is wise'. So this time

we replace this predicate with a 'predicate variable', X, and prefix the result with a universal or existential quantifier to create a generalization in predicate position.

$\forall X$, Socrates Xs.

$\exists X$, Socrates Xs.

The sense in which these claims express generality in predicate position is completely analogous. They bear a special logical relationship to their *instances*. As before we identify the position we are generalizing into (the position of the predicate 'is wise' above) and we form a sentence that has a gap in this position—i.e. a sentence that is variable with respect to that predicate:

Socrates . . .s

This defines a class of sentences parameterized by the different predicates we can substitute into the gap, and these constitute the instances of the predicate generalization.

Socrates is wise, Socrates talks, Socrates is old, (. . .and so on)

The logical relationship is also exactly the analogous. From the universal claim any instance can be inferred: from the claim that $\forall X$, Socrates Xs one can infer that Socrates talks. Conversely, if you can prove an 'arbitrary' instance—let's say you can prove that Socrates talks without making assumptions specific to talking—then that reasoning would apply just as well to any other instance, and we may infer the universal claim $\forall X$, Socrates Xs. The elimination and introduction rules for the predicate existential quantifier are also completely analogous to the existential name quantifier.

Here we use the same symbols, $\forall$ and $\exists$, to express generality in predicate position as we did for generalizing into name position, to emphasize the analogy in their logical behaviour. However, they are not the same devices: in fact, on one way of regimenting them, both sorts of quantifiers belong to grammatical categories, indeed belong to *different* grammatical categories, and so are as different as a predicate is different from a name.

We do not have ready approximations of these generalizations in English—not even the imperfect context-sensitive approximations we had for first-order generalizations.[9] But that is OK. The handle we have on these generalizations does not have to go by way of first translating them into an already understood natural language—after all, a child comes at some point to understand first-order quantificational phrases without already having a natural language to translate them into. We can come to understand predicate generalizations in exactly the same way that we came to grips with generality in name position above, namely by their logical relationship to sentences of the language that do not have predicate quantifiers. As with generalizations in name position, these predicate generalizations are pinned down, up to logical equivalence, by these relationships.

Theorem 0.2 (Harris's theorem for predicate generalizations)**.** *Suppose* $\forall_1$ *and* $\forall_2$ *are two devices that express generality in predicate position, in the sense that they both satisfy the universal introduction and elimination principles. Consider a sentence of the form* $\ldots X \ldots$*, where the variable X takes the position that a predicate would ordinarily occupy. Then* '$\forall_1 X \ldots X \ldots$' *and* '$\forall_2 X \ldots X \ldots$' *are logically equivalent in the sense that they can be inferred from one another.*

There is nothing special about the grammatical category of predicates and names. We can, using this same sort of logical analogizing, introduce devices that generalize into the

position of sentences, binary connectives, adverbs, or of quantificational phrases themselves. In general, we will call generalizations in positions different from name position *higher-order generalizations*. Given a sentence containing expressions of several grammatical categories, such as

It is possible that Socrates talks

we can form a sentential function by inserting a gap into any one of these expressions—a sentence that is variable in that position:

1. It's possible that . . . is wise
2. It's possible that Socrates . . .
3. It's possible that . . .
4. . . . Socrates is wise

As we have seen, a name quantifier lets us make generalizations in the position of the ellipses in 1, logically connecting it to all the possible ways of replacing the ellipses with names, and a predicate generalization is analogously connected to the ways of filling out the ellipses with arbitrary predicates. So it is not hard to spell out the role of a generalization into sentence position, or operator position. A sentential generalization, which we might write '$\forall p$ it's possible that p', is logically connected to the results of replacing the ellipses in three with different sentences, and an operator generalization with the results of replacing the ellipses in four with different operator expressions.

Before we move on, let us look at another logical expression that looks like it might be susceptible to this style of logical analogizing: the identity relation. Identity is also pinned down, up to logical equivalence, by a pair of principles. The introduction principle lets us always infer 'a is identical to a' whatever name we substitute for a. The elimination principle (Leibniz's law) lets us infer, from the assumption 'a is identical to b' that 'if $\ldots a \ldots$ then $\ldots b \ldots$', where '$\ldots a \ldots$' represents a sentence in which a appears in the position of a name.

Theorem 0.3 (Harris's theorem for first-order identity)**.** *Suppose* $=_1$ *and* $=_2$ *are two binary predicates that satisfy the identity introduction and elimination principles. Then* $a =_1 b$ *and* $a =_2 b$ *are logically equivalent, whenever a and b are names, in the sense that they can be inferred from one another.*

Proof. Suppose $a =_1 b$. Since $a =_2 a$ we can infer by Leibniz's law for $=_1$ that $a =_2 b$ replacing the second a with b. The converse direction is proved in the same way. □

First-order identity takes two names as arguments and produces a sentence. However, we could imagine that there was also an identity *connective* which, like conjunction, combines with two sentences to form a sentence, that is subject to the same pair of introduction and elimination rules. For any grammatical type whatsoever, we could introduce a binary operation that combines with a pair of expressions of that type to form a sentence and satisfies the introduction and elimination rules. Some of these may be approximated in English: we might write

To be wise is to be aware of how little one knows

to gloss the result of applying the predicate version of this operation to 'is wise' and 'is aware of how little one knows', but we do not need to rely too much on our pretheoretic grip given our stipulations pin down the operation up to logical equivalence.

0.4 Abstraction

Apart from the ability to generalize, another key expressive device in the higher-order languages explored in this book is the ability to form new meaningful expressions by *abstraction*. Abstraction is a process by which we can form complex predicates from a uniformly parameterized class of sentences, and by which we can do similar things with expressions of other grammatical types. The process is best illustrated in two steps. Take, for instance, a complex sentence like

Socrates is wise and Socrates is old.

As we did previously, we can take a particular word, such as 'Socrates', and replace it with a gap forming a sentential function

. . . is wise and . . . is old.

This sentential function associates 'Aristotle' with the sentence 'Aristotle is wise and Aristotle is old', it associates 'Plato' with 'Plato is wise and Plato is old', and so on. According to the abstraction hypothesis, one can now introduce a meaningful predicate which, when applied to any name *a*, yields a sentence synonymous with '*a* is wise and *a* is old'. In English a good candidate would be the complex predicate 'is wise and old', for plausibly '*a* is wise and old' is synonymous with '*a* is wise and *a* is old'.

More generally: to every sentential function, associating meaningful names to meaningful sentences, there exists a meaningful predicate which, when applied to a name, is synonymous with the sentence associated with the name by sentential function. This process of abstraction, if it is legitimate, can be applied in other grammatical types. For instance, we can form a sentential function by putting a gap where a sentence is:

. . . and Socrates is old

This sentential function associates with each meaningful sentence another sentence—for instance, it associates 'snow is white' with the sentence 'snow is white and Socrates is old'. By abstraction there corresponds to this sentential function a meaningful operator expression that, when applied to a sentence, *A*, yields a sentence synonymous with '*A* and Socrates is old'. Or we could abstract in the position of a predicate

Socrates . . .

giving us a class of sentences—'Socrates walks', 'Socrates talks', etc—parameterized by predicate we insert into the gap. The theory of abstraction tells us that there is a quantifier phrase that combines with a predicate, *F*, to make a sentence synonymous with 'Socrates *F*s'.

The process of abstraction also allows us to create new binary relations, and expressions of higher arity. One way to do this is to abstract from expressions that aren't sentences to begin

with. For instance, by removing 'Socrates' from the complex unary predicate 'diligently studied under Socrates' we obtain a predicate function

diligently studied under . . .

which yields a unary predicate whenever the ellipses are replaced by a name. So by abstraction we obtain a *binary* predicate which, when supplied with a name *a* yields a unary predicate synonymous with the unary predicate 'diligently studied under *a*'. We can also abstract on sentential functions of two arguments, like

. . . is wise and ___ is old

which associates each pair of names with a sentence—e.g. 'Socrates' and 'Aristotle' with 'Socrates is wise and Aristotle is old'. The result of abstracting in both positions yields a binary relation which when applied to a pair of names *a* and *b* is synonymous with the sentence '*a* is wise and *b* is old'.

The reader may wonder if this process of abstraction is well-defined. One way it could fail to be well-defined would be if there could be two non-synonymous predicates that, when each applied to any name *a* yielded a sentence synonymous with '*a* is old and *a* is wise'. If this sort of thing can happen, we need a further stipulation about the behaviour of abstracted expressions. If we abstract from the sentential function

. . . talks

there may be several non-synonymous predicates which, like the predicate 'talks', yields a sentence synonymous with '*a* talks' when applied to *a*. If there is a choice here, we might as well take a synonym of 'talks' to be the abstracted predicate. We might make the further stipulation that abstraction in degenerate cases like these is always synonymous with the predicate you started with. It turns out, given this further stipulation (and a suitably strong principle governing substitution of synonyms) that we can prove an analogue of Harris's theorem for abstraction (Proposition 3): any two methods of abstraction that satisfy our stipulations will always yield synonymous abstracted expressions.

Another way abstraction could fail to be well-defined is if no predicate can be introduced into the language that satisfies the stipulations. How could this process of abstraction fail to produce such a predicate? Let's look at an example. The logical atomists liked to treat logical words like 'not', 'and' and 'all', not as stand-alone expressions but as "syncategorematic". Here's an analogy: when we combine a name with a predicate by applying the former to the latter we do so by prefixing the predicate to the name *Fa*. A notational variant that makes it clear that application is the mode of combination would be to write something like $\mathrm{app}(F, a)$. In adopting this notation we aren't treating 'app' as a meaningful word on its own—it does not stand to anything in reality as the words *F* and *a* do—it is simply indicating how *F* and *a* should be combined. In either notation, application is syncategorematic. This is how the logical atomists thought about the logical words. Take our example 'Socrates is wise and Socrates is old'. According to the atomists, the word 'and' does not stand for anything in reality in the same way that the word 'Socrates' does, rather it indicates how the two sentences 'Socrates is wise' and 'Socrates is old' should be combined—by conjoining them as opposed, say, to applying one to the other. Words like 'and' and 'not', therefore, do not correspond to meaningful operators and connectives in their own right, any more than

application does. If definitions by abstraction were legitimate, by contrast, we could introduce a self-standing negation operator by abstraction on the sentential function 'not . . .', and by similar methods we could introduce a conjunction connective. The logical atomists rejected these stand-alone connectives, essentially, because it committed them to a richer *higher-order* ontology: they reject certain higher-order generalizations into operator position and connective position that follow, by existential generalization, from true sentences involving these expressions.[10] For instance

$\forall p$, for it to not be the case that p is for it to not be the case that p

entails the existential

$\exists X, \forall p$, for Xp is for it to not be the case that p

Note that this existential doesn't imply the existence of any individuals—it is not committal about 'ontology' as it is usually studied, relating to questions formulated using the first-order existential quantifier. However, once one has accepted the higher-order framework there is, in addition to the study of ontology, a parallel domain of inquiry this time concerned with existential quantification into the positions of predicates, sentences and other expressions, which we might call 'higher-order ontology' by analogy.

Remark 0.2. Indeed, the reader may have noticed that if there are no restrictions on abstraction we could abstract from the two-place sentential function obtained by replacing 'Socrates' and 'talks' with gaps in 'Socrates talks'. The result is a sentential function taking a name and a predicate as an argument

. . . ___s

The resulting binary operation when applied to a name and a predicate, a and F yields something synonymous with 'a Fs'—i.e. a stand-alone operation of application.

The idea of unrestricted abstraction is closely related to other puzzles in metaphysics. The *standard* formalism for notating abstracted predicates, which will be presented in Chapter 3, actually provides us with *two* ways to abstract a binary predicate from the sentential function:

. . . loves ___

These two abstracted binary predicates are not merely notational variants of one another. According to the naïve theory of abstraction, they are non-synonymous mutual 'converses'. The idea of a converse seems intimately bound up with the notion of order in which the relation receives its arguments. But what does this really mean? It is especially obscure if reality itself does not have a left-to-right ordering mirroring the ordering of a predicate in a standard linear notation. The higher-order commitment to converses is a hotly contested issue in metaphysics. But by admitting unrestricted kinds of abstraction we can introduce binary predicates and prove higher-order existential statements like the following, using higher-order existential generalization:

$\exists R$ for John to R Mary is for Mary to love John

A higher-order 'ontological commitment' we obtain from having meaningful terms in the language corresponding to converses, and our prior commitment to higher-order existential generalization.

The first half of this book takes it for granted that this unrestricted process of abstraction is in good standing, and takes the reader through the many sorts of things we can do with it. In part III of the book, however, we consider languages in which the method of abstraction is restricted in certain ways that allow us to be more neutral about matters of higher-order ontology without relinquishing the introduction and elimination rules for the quantifiers.

0.5 Some things that higher-order generalizations are *not*

Higher-order logic is *useful*. We'll provide some examples shortly, but before we do that let's briefly look at some other sorts of sentences that are often employed by philosophers to do similar work. These alternatives are sometimes confused with higher-order generalizations, so we will use this as an opportunity to point out some important distinctions and common pitfalls.

The most common mistake is to identify a higher-order generalization with a first-order generalization restricted to an appropriate sort of abstract object. So let's begin by looking at the similarities and differences between a predicate generalization and a first-order generalization restricted to properties. Properties are a special sort of individual. There is some philosophical dispute both about whether they exist at all (nominalists say they don't, platonists say they do), and about their nature if they do. These include questions like: 'are they abstract objects?', 'do they have locations?', 'are there logically complex properties?', or 'are properties reducible to sets?'. But the crucial point is that we use first-order generalizations to theorize about them. The view that all properties are located—i.e. $\forall x$ if x is a property, then x is located—is generalizing into the position of a name, not a predicate. From this generalization, we can infer that if wisdom is a property, then wisdom is located, that if Socrates is a property, then Socrates is located, and so on, by substituting the particular names 'wisdom', 'Socrates', and so on, for the variable x. Its instances differ over the names that replace the variable x.

To theorize about properties, we need some primitives for talking about them. Presumably, we need a first-order predicate 'is a property' to draw the distinction between things that are properties and those that are not. We also need to help ourselves to a binary first-order predicate 'instantiates' relating an individual to property. Now there are supposed to be connections between sentences involving names for properties, like 'wisdom', and sentences involving predicates, like 'is wise'. For instance:

Socrates instantiates wisdom if and only if Socrates is wise.

These biconditionals connect questions about which properties Socrates instantiates to the truth of the instances of a predicate generalization. This suggests that we might be able to come up with a first-order property-theoretic sentence that has the same logical power as a predicate generalization. Namely, we could use

$\forall x$, if x is property then Socrates instantiates x

$\exists x$, x is property and Socrates instantiates x

to express the same sort of generality as we did with '$\forall X$, Socrates Xs' and '$\exists X$, Socrates Xs'. The general strategy for paraphrasing higher-order generalizations using first-order generalizations is this: replace every predicate variable X with the open predicate 'instantiates x' where x is a first-order variable, and replace all occurrences of '$\forall X\ldots$' and '$\exists X\ldots$' with restricted first-order quantifiers '$\forall x$ if x is a property…' and '$\exists x$, x is a property and…'.

If these sentences have the logical role of a higher-order generalization then they should satisfy the quantifier introduction and elimination inferences. For instance, for the universal elimination rule we need the following inference to be a good one:

$\forall x$, if x is property then Socrates instantiates x
Therefore, Socrates is wise

The dual inference for the existential (existential introduction) is:

Socrates is wise.
Therefore, $\exists x$, x is a property and Socrates instantiates x.

These inferences are guaranteed, provided that for each instance—Socrates is wise, Socrates is happy, etc—there is a corresponding property—wisdom, happiness, etc.—such that Socrates instantiating it is equivalent to the corresponding instance.

But these inferences have a poor reputation. Whereas 'Socrates is wise' all by itself suffices for the truth of the existential predicate generalization '$\exists X$, Socrates Xs', it does not suffice for the truth of '$\exists x$, x is a property and Socrates instantiates x'. In order for the latter to be true, there have to be these further special abstract objects—properties. For the nominalist, then, these first-order generalizations do not have the logical role of a predicate generalization: the former would be vacuously true even when Socrates isn't wise, and the latter vacuously false even when Socrates is wise.

Bertrand Russell famously revealed a more fundamental problem with these inferences. An instance of existential generalization is:

$\forall y$, y doesn't instantiate itself if and only if y doesn't instantiate itself.
Therefore, $\exists X$, $\forall y$, y Xs if and only if y doesn't instantiate itself.

Of course, the premise is unassailable, so by the existential introduction rule for predicate generalizations, the conclusion should be true. The proposed first-order replacement of this conclusion is, by contrast, inconsistent:

$\exists x$, x is a property and $\forall y$, y instantiates x if and only if y doesn't instantiate itself.

The reason is this. Suppose x is instantiated by all and only those things that do not instantiate themselves. Then x must instantiate itself if and only if it does not instantiate itself.

It follows, then, that first-order generalizations over properties cannot be used to approximate generalizations into predicate position: a first-order generalization will always overlook some instances of the second-order generalization.

It is worth noting that some authors use the word 'type theory' and 'higher-order logic' somewhat differently from the way it is used in this book. According to this alternative use of these terms, type theory and higher-order logic refer to a particular framework for

theorizing about first-order properties. Consider, for example, the following excerpt from the Stanford Encyclopedia of Philosophy entry on properties (Orilia and Swoyer (2020), Section 7.3.)

> Type theory has never gained unanimous consensus and its many problematic aspects are well-known [. . .]. Just to mention a few, the type-theoretical hierarchy imposed on properties appears to be highly artificial and multiplies properties ad infinitum (e.g., since presumably properties are abstract, for any property P of type n, there is an abstractness of type $n + 1$ that P exemplifies). Moreover, many cases of self-exemplification are innocuous and common [. . .]. For example, the property of being a property is itself a property, so it exemplifies itself. There also seem to be transcendental relations. A transcendental relation like *thinks about* is one that can relate quite different types of things: Hans can think about Vienna and he can think about triangularity. But typed theories cannot accommodate transcendental properties without several epicycles.

The term 'type theory', as it is used in this book, is merely a theory systematizing the rules that describe how names, predicates, operators, sentences, and so forth combine. The sorts of limitations it imposes—for instance, that one cannot apply a name to an operator—are completely common sense and implicit in the logical languages the reader is likely already familiar with. By contrast, Orilia and Swoyer are using the phrase 'type theory' to refer to a very specific first-order theory of properties: one in which properties are stratified into different levels—levels resembling but distinct from the grammatical types of expressions—and in which the instantiation relation cannot hold between properties of different levels. Leveled at this theory of properties these criticisms seem apt. They make less sense when targeting the use of typed languages in our sense, which includes widely used languages like the propositional calculus.

A related, and common refrain is that higher-order logic is really a property theory or set theory in 'sheeps clothing', to use Willard van Orman Quine's metaphor.[11] Quine essentially understands a sentence of the form '$\exists X$, Socrates Xs' as asserting that Socrates instantiates a property (or belongs to some set), and so understood these sentences of course do engender a commitment to properties. In this book, however, we interpret sentences like '$\exists X$, Socrates Xs' as a higher-order generalization. Do higher-order generalizations commit you to properties in any interesting sense? The term 'ontological commitment' has a fair amount of baggage in philosophy, which I won't attempt to untangle here.[12] But whatever we mean by it, entailments seem to transmit ontological commitments:

> If A entails B and B commits you to Fs (sets, unicorns, etc.), then A commits you to Fs as well.

This lets us infer, for instance, that

> '$\exists x$, x is wise' commits us to nothing that 'Socrates is wise' does not already commit us to.

The reason is simple: by the rules governing existential generalization, 'Socrates is wise' entails '$\exists x$, x is wise', so everything the latter entails the former entails, by the transitivity of entailment. Thus, by entirely analogous reasoning

'$\exists X$, Socrates Xs' commits us to nothing that 'Socrates is wise' does not already commit us to.

In particular, this tells us that if '$\exists X$, Socrates Xs' commits us to sets or properties, then so does 'Socrates is wise'. But what does the claim *that Socrates is wise* commit us to ontologically? Presumably, it commits us to wise people, and to Socrates. But it does not commit us to sets, properties or other abstract objects. After all, a nominalist can obviously classify people as wise or not wise without giving up their nominalism. If sentences like 'Socrates is wise' do commit you to sets, then very little is free of this ontological commitment. Quine's quip that second-order logic is set theory in disguise may be true, but it is hardly a rebuke. By these lights biology, astronomy, and so on, are set theory in disguise too, as, indeed, is any theory that makes assertions in subject predicate form.

Let us end this section by discussing another class of statements that are closely related to higher-order generalizations: conjunctions and disjunctions. Like the universal claim '$\forall x$, x is wise' , the infinite conjunction

Socrates is wise and Aristotle is wise and Plato is wise and . . .

bears a tight logical relationship to the list of instances above. Similarly the infinite disjunction 'Socrates is wise or Aristotle is wise or . . .' is secured by any disjunct, much as the corresponding existential is secured by any instance. However, there is an important logical disanalogy. The infinite conjunction above is logically weaker than the universal claim $\forall x$, x is wise. This point essentially goes back to Russell (who is in turn drawing on Hume): you will not be able to infer the universal claim without an extra quantificational premise to the effect that Socrates, Aristotle, Plato and so on, are all the things there are.[13] Indeed, it is arguably possible for the conjunction to be true while the universal false: suppose there had been new individuals that do not in fact exist. Perhaps, at such a possibility, all the actually existing individuals are wise, securing the truth of the conjunction, but the new individuals may fail to be wise, making the corresponding universal generalization false. I have illustrated this point with a first-order generalization, but this point applies just as forcefully to higher-order generalizations in other positions. Having said this, however, it is worth noting that disjunctions and higher-order existentials are in other ways more alike than the latter is to existential quantification over properties. The disjunction 'Socrates is wise or Socrates talks or Socrates walks. . .', like '$\exists X$ Socrates Xs', clearly does not entail the existence of any properties.

0.6 Higher-order generalizations in philosophy

A central contention of this book is that higher-order generalizations are useful for philosophical theorizing. I think the best way to appreciate this point is to examine a few representative debates in philosophy.[14]

Let's begin with an important theme of 20th century philosophy: the notion of metaphysical necessity as it is found, most notably, in Kripke's work.[15] A principal component of Kripke's conception of metaphysical necessity is that it is supposed to be 'necessity in the highest degree'. What is the proper way to formalize this claim? Instead of answering this question directly, it is best to begin by saying more about the role the claim plays in philosophical theorizing. Among other things, we can appeal to material conditionals like the following:

If it's metaphysically necessary that this table is mostly made of wood, then it's physically necessary that this table is mostly made of wood.

The slogan that 'metaphysical necessity is necessity in the highest-degree' is not itself an interesting philosophical thesis without concrete implications like the one above. It is significant to the extent that we can draw these consequences from it, for it is these consequences that are used in philosophical applications, not the slogan itself. So a proper formalization of the thesis must entail conditionals like this one.

The above conditional is just one example of many. Any sentence obtained from this sentence by replacing 'it's physically necessary that' with some other operator expressing a kind of necessity should be true as well. Similarly, any sentence obtained by replacing 'this table is mostly made of wood' with another sentence should also be true. We can express this using the higher-order predicate of operators Nec introduced earlier: every sentence obtained by replacing the ellipses below with an *arbitrary* operator expression should be true

If $\mathrm{Nec}\ldots$ and it's metaphysically necessary that this table is mostly made of wood, then ... this table is mostly made of wood.

Sentences of this form are all instances of a higher-order generalization which universally generalizes into the position of the ellipses above. Similarly, the result of replacing 'this table is mostly made of wood' with any other sentence in the above will also yield a truth, and sentences of this form are all instances of a further higher-order generalization, this time in sentence position. Thus, the claim that metaphysical necessity is the broadest kind of necessity is partially captured by a doubly quantified higher-order generalization. Writing '$\Box$' for metaphysical necessity:

$$\forall X(\mathrm{Nec}\, X \rightarrow \forall p(\Box p \rightarrow Xp))$$

where X is an operator variable, and p a sentence variable.[16]

Observe that the question of whether there are notions, like logical necessity, that outstrip metaphysical necessity is quite orthogonal to the debate between nominalism and platonism. In principle, the nominalist and the platonist could accept or reject the thesis that metaphysical necessity is necessity in the highest degree. So we cannot properly capture this idea with a first-order generalization restricted to a special sort of abstract entity, a *necessity*, for the truth of *that* sort of generalization is not orthogonal to the nominalism-platonism dispute. Moreover, depending on how we respond to Russell's paradox, a generalization over abstract objects may also fail to have the concrete implications that constitute the claim's philosophical role.

Another area where higher-order generalizations are useful is in the formulation of supervenience theses. For concreteness, we can take the thesis that the mental supervenes on the physical. This thesis ought to imply that if Mary is in pain we should be able to fill out the ellipses below with a predicate stated in physical terms that applies to Mary, and in a way that makes the conditional come out true:

It's metaphysically necessary that if Mary ... then Mary is in pain

So it seems like the proper way to spell this out is with a higher-order existential generalization into predicate position. More generally, the supervenience thesis suggests there ought

to be such a filling out of the ellipses above whenever we replace 'is in pain' with any other mental predicate, suggesting we are looking for a $\forall\exists$ sentence with alternating higher-order quantifiers. Corresponding to the distinction between predicates that use only physical language and those that don't, we might posit a corresponding distinction in reality: let Phys be a higher-order predicate that combines with a first-order predicate in order to make this distinction, let Ment do the analogous thing for 'mental', and let m be a name for Mary. So the supervenience thesis can be captured by a higher-order generalization:

$$\forall X(\mathrm{Ment}\, X \wedge Xm \rightarrow \exists Y(\mathrm{Phys}\, Y \wedge Ym \wedge \Box(Ym \rightarrow Xm)))$$

Often supervenience theses are formulated in terms of properties. Could Mary and Nora instantiate the same physical properties while instantiating different mental properties? The nominalist thinks not: no two people can instantiate different mental properties because there are no properties to instantiate. Yet the question of supervenience still seems like a substantive one: the nominalist might still wonder whether Mary and Nora could be physically alike—that Mary's c-fibres are firing iff Nora's are, and so on—yet differ over whether they are in pain. This nominalist can express such counterexamples with a higher-order generalization: $\forall X(\mathrm{Phys}\, X \rightarrow (Xm \leftrightarrow Xn))$ conjoined with the claim that Mary is in pain but Nora isn't $Fm \wedge \neg Fn$. Nominalism is an extreme view about the existence of properties, but property-theoretic formulations fail to track the issues of substance on many less extreme views. Perhaps the best solution to Russell's paradox and related paradoxes is to reject the existence of certain properties concerning intentional notions, including many mental properties. Property-theoretic formulations of supervenience are therefore entangled in all of these extraneous issues, whereas the higher-order formulations are closer to the philosophical action.

Metaphysicians will often talk about 'qualitatively indiscernible' situations or objects. Two objects are indiscernible when they cannot be distinguished without making reference to particular individuals. Qualitative predicates include, for instance, 'is round', 'is heavier than something', 'self-instantiates', whereas non-qualitative predicates include 'is Jupiter shaped', 'is heavier than Jack', 'instantiates wisdom'. Leibniz, for instance, thought that qualitatively indiscernible objects had to be the very same object. By contrast, Max Black thought that two identically proportioned iron balls, in otherwise empty space, would be qualitatively indiscernible yet distinct Black (1952). What is the philosophical cash value of the claim that a and b are qualitatively indiscernible? From this claim, we can infer any of the following biconditionals

a is round if and only if b is round.

a is heavier than something if and only if b is heavier than something.

a doesn't instantiate itself if and only if b doesn't instantiate itself.

By now, I hope, it is clear that the proper formalization of qualitative indiscernibility involves a higher-order generalization:

$$\forall X(\mathrm{Qual}\, X \rightarrow (Xa \leftrightarrow Xb))$$

The claim that a and b instantiate the same qualitative properties does not suffice, for it does not, without taking a stand on orthogonal issues, let us infer the above biconditionals.

0.7 Semantics and model theory for higher-order languages

It does not appear to be possible to state the intended meanings of sentences expressing first-order generalizations—that is, to furnish a first-order language with a "semantics"—without using generalizations in the statement of these meanings. It does not seem promising, for example, to describe the semantics for a first-order language in a propositional metalanguage. For the same sorts of reasons, we should not expect to be able to describe the semantics of sentences expressing higher-order generalizations in a language that only has first-order generalizations. In this case, it is less obviously impossible—much of contemporary mathematics can be stated in the first-order language of set theory, so one has a lot more to work with than in a propositional language. However, for reasons we have already discussed, first-order generalizations over sets seem no more promising than first-order generalization over properties as approximations of higher-order generalizations.[17]

Nonetheless, first-order set theory is useful for a very different enterprise. Suppose that you want to know whether your favourite higher-order theory contains a contradiction: can you derive a contradiction from your favoured principles using the background logical rules? Or perhaps we just want to see whether your favourite principles let you prove some other undesirable consequence that falls short of a contradiction. It is natural, then, to want to have general mathematical tools for probing when one sentence is derivable from some others.

A natural strategy for showing that a sentence cannot be derived from some others using a given set of axioms and rules to find a property that the assumptions and axioms have, that is preserved by the rules, but which is not had by the sentence in question. For if the axioms and the assumptions have the property, and the property is preserved by the rules then everything derivable from the assumptions will have the property; any sentence that does not have the property will not be derivable from the assumptions. In order for this strategy to be practically useful, we need to isolate a class of properties which are automatically preserved by the logical axioms and rules, and the standard answer to this demand is *model theory*. Model theory typically proceeds by defining a class of mathematical structures, *models* (usually made out of sets, but this is not essential) and defining a special relation between models and sentences, *truth in*, in such a way that for each model M, the logical axioms and rules automatically preserve *being true in M*.

Now model theory should be distinguished sharply from the project, mentioned at the outset, of describing the intended meanings of a language. As a case in point, the model theory for propositional logic consists of certain functions that map sentences to the numbers 1 and 0, and a sentence is true in one of these models when it is mapped to 1. The originators of this method, Emil Post and Ludwig Wittgenstein, did not think that the actual meanings of sentences were numbers (or truth values). Nonetheless, the truth-value method is perfectly suited to establishing what follows from what in propositional logic: it is sound and complete—a sentence is derivable from premises if and only if it is true in every model that makes the premises true.[18] Part IV of this book is devoted to the model theory of higher-order logics; some further remarks on higher-order semantics and its relation to model theory may be found in Section 15.5.

0.8 Glossing higher-order generalizations in English

Having distinguished higher-order quantification from first-order quantification over properties and sets, we face a more practical problem of glossing claims like '$\exists X$ Socrates Xs'

in ordinary English, which is not a higher-order language. As we have already suggested, the handle we have on these generalizations does not have to go by way of first translating them into English. However, sometimes it is easier to say something suggestive in English that brings a sentence of higher-order logic to mind than to squint at a series of logical symbols.

Some higher-order generalizations arguably do have natural language analogues. For instance, Arthur Prior suggests that in the inference from 'She met him in Paris' to 'She met him somewhere', we are replacing a prepositional phrase with the quantificational phrase 'somewhere', and so 'somewhere' is best understood as expressing a higher-order generalization into the position of a prepositional phrase, and not a generalization into name position.[19] On the other hand, while we do not seem to have something that expresses generality in predicate position in English, once we have the concept of a predicate generalization we can say things like 'Socrates somethings' or 'Socrates whatevers' and make it clear enough what one means. Using the analogy with the prepositional phrases like *somehow*, *however*, *somewhere*, *wherever* and *anywhere* Prior suggests we simply concoct terms out of the analogous sentential 'wh'-word, *whether*, to play the role of sentential quantification: *somewhether*, *anywhether* and *everywhether*. In order to make quantificational statements we need some natural language device that does the work of sentential variables. Since pronouns play the role of first-order variables in English, we need something that stands to sentences as pronouns stand to nouns: prosentences (Grover, 1992). Again, by analogy with the way in which we have prepositional phrases like *there*, which can be bound by quantificational expressions like *anywhere*—as in *if Alice met Bob anywhere, she met Carol there too*—Prior suggests the word *thether*. Thus we might write 'if anywhether then thether' to capture the higher-order generalization $\forall p(p \rightarrow p)$, or 'anywhether Alice thinks that, it's necessary that' for $\forall p(Tp \rightarrow \Box p)$.

None of these solutions is particularly elegant, and none of them is fully general. A more common practice for glossing sentences of higher-order logic in the philosophical literature is to use a first-order property generalization, like 'Socrates has some property' to *indicate*, without being *synonymous* with, a higher-order generalization; in this case, '$\exists X$ Socrates Xs'. This has the disadvantage that it can lead some to think that higher-order logic is a framework for theorizing about first-order properties. However, having clarified the true relationship between sentences higher-order generalizations and property-theoretic generalizations no confusion should arise from this practice.

0.9 How to read this book

This book should certainly not be your introduction to logic. Familiarity with the syntax and model theory of propositional and first-order logic will be assumed. Apart from this, however, there are no hard prerequisites. There are parts of the book where the reader would benefit from knowing some elementary modal logic, especially Chapters 7 and 8 and to a lesser extent Chapters 17 and 18. While the book is self-contained in this respect, and defines basic notions like that of a *normal modal logic* and a *Kripke frame*, the book does not attempt to teach these concepts. Chapters 1–4 and 6 of Cresswell and Hughes (1996) will be more than sufficient; there are also routes through the book that do not require any modal logic.

While this book is intended to take the reader progressively through some logical and meta-theoretical concepts in higher-order logic, chapters do not always depend on

all the chapters prior to them. There are consequently a couple of routes through the book that one can focus on depending on your goals. One route focuses on the higher-order logic *Classicism*, its applications to modal metaphysics and its model theory, and the other on general λ-languages and their applications to structured theories of propositions.

- Classicism and modal metaphysics: The reader should read part I of the book for an introduction to the simply typed λ-calculus, and part II for higher-order logic, the system Classicism, and the modal notions and principles that can be expressed in Classicism. Part IV of the book covers model theory for a variety of languages (Chapters 14–18), but the reader should pay special attention to Chapter 18 on the model theory of Classicism.
- Structure: The reader should read part I of the book for the simply typed λ-calculus, and Chapters 4 and 5 from part II for higher-order logic. They may then skip ahead to part III to learn about general lambda languages and the applications to structured theories of reality. The chapters on model theory (Chapters 14–17) and Appendix A and B will equip the reader with the necessary tools for modeling these theories.

This book also contains exercises. It is recommended that the reader have a pen and paper at hand while working through this book, so that exercises can be attempted as the reader proceeds. There are two sorts of exercises in this book. Those labeled *comprehension checks* are not supposed to be very challenging, they simply offer the reader an opportunity to apply a concept they have just encountered and can be attempted in the head. *Exercises*, by contrast, are longer and sometimes more challenging and will require pen and paper. Exercises are also an opportunity to explore concepts and principles that we have not had space to cover fully in the book. Many of the exercises contain statements of important facts and should be considered as much a resource for such facts as the numbered theorems, propositions and lemmas found throughout the book.

0.10 Other resources

Higher-order logic and type theory is a vast subject with practitioners belonging to a variety of disciplines: mathematical logic,[20] intuitionistic mathematics,[21] category theory,[22] theorem proving,[23] the foundations of programming languages,[24] linguistics,[25] and in philosophy, the philosophy of mathematics,[26] semantics and metaphysics.[27] As a result any book on higher-order logic will by necessity contain many omissions. In the case of this book, these omissions are for the most part well-covered elsewhere. The reader may consult the references provided here for further details.

Higher-order logic originated with mathematicians, and for a while higher-order logic, not set theory, was the preferred language for foundational mathematics. Actual higher-order mathematics has a long and prestigious lineage, tracing back to Frege (1879), and continuing with Peano (1889),[28] Whitehead and Russell (1910-1913), Zermelo (1908),[29] Hilbert and Ackermann (1928),[30] Bernays and Schönfinkel (1928), Tarski (1931),[31] Carnap (1947), Church (1940), Kreisel (1967), Henkin (1950), Montague (1965, 2014), Montague (2014), Friedman (1975c), Simpson (2009).[32] Opposing the higher-order mathematicians were Skolem and Gödel, who propounded the use of first-order logic in mathematics, an attitude that Quine advocated for more generally in philosophy.[33] I will end these prefatory remarks by listing some topics in higher-order philosophy that have not been

treated as directly in this book as I would have liked—I include with them some references to further resources in the endnotes: plural logic,[34] ramified type theory,[35] the metaphysics of grounding,[36] higher-order contingentism,[37] and applications of higher-order logic to propositional attitudes.[38] While discussions of these topics are limited, my hope is that this book will equip the reader with the necessary tools to explore these topics on their own.

Endnotes

1. See Prior (1971) p. 135.
2. Frege (1879), Peano (1889). We'll return to these systems briefly in Section 1.4.
3. There are several possible explanations for this. One possible reason is that early untyped logical languages were found to be inconsistent. But I think a more fundamental reason is that they are quite hard to interpret; as we have noted already, natural languages are typed languages and so it is not always possible to translate an expression of an untyped language into an antecedently understood sentence of a natural language. If we explain that the symbol F means *walks* and $\neg$ means *it's not the case that*, then expressions like $F\neg$ don't seem to correspond to anything comprehensible.
4. Cf Sider (2011) Chapter 6, Dorr and Hawthorne (2013).
5. See Sider (2011) Section 7.13.
6. The reader should consult Harris (1982) for more discussion, and a more precise formulation of this argument.
7. Philosophers have been moved to this picture from a variety of motivations, from making sense of ontological disputes, to the paradoxes of set theory. Many of these motivations are discussed in the collection Rayo and Uzquiano (2006).
8. These sorts of worries are spelled out further in Williamson (2003).
9. We do seem to be able to express generality in the position of an adjective—like 'wise' in 'Socrates is wise; so Socrates is something'—but not into the position of the whole predicate 'is wise'.
10. Defenses of this position on the connectives can be found in Russell (1918), Wittgenstein (1922) and Ramsey (1927). Russell and Ramsey were often explicit in expressing these ideas in higher-order terms.
11. Quine (1970); see the section heading in Chapter 5.
12. A particular treatment of the notion, that seems faithful at least to the way Quine uses the term, states that a claim, P, commits you to Fs just in case P entails the existential sentence $\exists xFx$. This analysis substantiates the claim that entailments transmit ontological commitments.
13. Russell (1918) pp. 69–70.
14. The examples and discussion in this section draw heavily on Bacon (forthcoming).
15. Kripke (1980).
16. We will return to the proper formalization of 'necessity in the highest degree' in Chapter 7. The above formalization says, putting it roughly, that the extension of metaphysical necessity is included in the extension of any other necessity. But this is too weak: presumably we require this inclusion to hold necessarily. This inclusion must hold of metaphysical necessity, but to be suitably strong the inclusion must be necessary in any other sense of necessary: i.e. the proper formalization of the claim is: $\forall Y(\mathrm{Nec}\ Y \rightarrow\ Y\forall X(\mathrm{Nec}\, X \rightarrow \forall p(\Box p \rightarrow Xp)))$.
17. The naïve strategy for paraphrasing higher-order generalizations in predicate position with first-order quantification over sets is to mimic the strategy we adopted with properties, this time replacing each predicate variable, X, with the open predicate 'belongs to x'. This strategy may initially seem *less* promising, because sets, unlike properties, are extensional. More sophisticated paraphrases avoid commitments to extensionalism—e.g. to accommodate intensionality we could quantify instead over functions from worlds to sets—but ultimately they are all subject to the same problems we discussed in relation to the property-theoretic approximations of higher-order generalizations: they take a stand on platonism, whereas higher-order generalizations do not, and do not play the same logical role due to Russell's paradox.
18. This method of showing logical independence between sentences using unintended interpretations was used extensively in logic and in geometry by David Hilbert and his followers. Hilbert reportedly once remarked "One must be able to say at all times–instead of points, straight lines, and planes–tables, chairs, and beer mugs".
19. Prior (1971) Chapter 3.
20. See the references in the next paragraph.

21. See especially intuitionistic type theories, originating with Martin-Löf (1972) and continuing recently as homotopy type theory Univalent Foundations Program (2013).
22. Bell (1982), Lambek and Scott (1988), MacLane and Moerdijk (2012).
23. See for instance Gordon and Melham (1993), Brown (2007).
24. Mitchell (1996).
25. Carpenter (1997), Jacobson (1999), Heim and Kratzer (1998), Barker and Shan (2014).
26. Linnebo (2013), Fine (2002a), Studd (2013), Scambler (2021), Goodsell (2022) for combine modal and higher-order resources in the philosophy of mathematics. Other papers in the philosophy of logic and mathematics involving higher-order logic include: Burgess (2005), Wright and Hale (2001), Boolos (1984), McGee (1997), Shapiro (1987), Rayo and Williamson (2003), Uzquiano (1999).
27. See, for instance, Dorr (2016), Bacon (2020), Jones (2018),Trueman (2020), and the citations below.
28. In Peano's original axiomatization of arithmetic the induction principle was formalized as a single second-order generalization (in modern presentations it is presented in a first-order context by an axiom schema with infinitely many instances). There is more discussion of Frege and Peano in Section 1.4.
29. This paper contained the first axiomatization of modern set theory. Modern mathematicians often use 'Zermelo Fraenkel set theory', or 'ZF', to refer to a first-order theory, however Zermelo's original system was second order.
30. While Hilbert and Ackermann (1928) are often cited as the first modern treatment of first-order logic, the book covers in its four chapters propositional logic, the monadic predicate logic (a version of propositional logic where the letters are interpreted as monadic predicates), predicate logic, and higher-order logic.
31. Tarski's original definition of truth, and the original foundational papers in the discipline that would later become model theory were formulated in higher-order languages, as opposed to the first-order language of set theory.
32. The reader should note that there is a very fuzzy distinction (which I haven't tried to draw) between 'first-order' higher-order mathematics, as it were, and mathematical work on the meta-theory of higher-order logic, exemplified in Church (1940), Henkin (1950). A selection of themes in higher-order mathematics can be found in Bell (2022), Väänänen (2012).
33. For a good overview of how the default framework in mathematics moved from higher-order logic to first-order logic see Moore (1988).
34. Plural logic was an important stepping stone to the rehabilitation of higher-order notions in post-Quinean philosophy; see, e.g., Boolos (1984), Schein (1993).
35. A venerable list of philosophers have expressed sympathy towards the ramified theory of types, including Bertrand Russell, Alonzo Church, David Kaplan, Saul Kripke, and Harold Hodes. There is a brief discussion of ramified type theories in Section 11.1. See Church (1976), Hodes (2015), Hatcher (1982) for a modern presentation of the ramified theory. Some critical discussion can be found in Bacon et al. (2016), Uzquiano (forthcoming).
36. Higher-order logic is especially pertinent here in relation to the Russell-Myhill paradox, and other puzzles of ground. The reader should consult Fine (2010), Krämer (2013), Fritz (forthcoming a), Fritz (forthcoming b), Fritz (2020), Litland (2020), Goodman (2022), for a start into this literature.
37. See for instance Fine (1977), Williamson (2013), Stalnaker (2012), Fritz and Goodman (2016). Further references are given in Section 8.4.
38. These papers include Bacon and Russell (2019), Caie et al. (2020), Yli-Vakkuri and Hawthorne (2021). The literature on Prior's paradox is also relevant here: see Prior (1961), Priest (1991), Rapaport et al. (1988), Tucker and Thomason (2011), Bacon et al. (2016), Bacon (2021), Bacon and Uzquiano (2018), Uzquiano (2021).

Part I

Typed languages

1

Typed languages

This chapter introduces the reader to the notion of a type, a typed language, the notion of a signature for a typed language (Sections 1.1 and 1.2). Philosophical confusions concerning the proper way to theorize about the semantics of typed languages are addressed in Section 1.3. Section 1.4 treats some extensions of the simply typed languages explored in this book, and the possibility of completely type-free languages is also raised and discussed.

1.1 Types

When one studies a logical language, such as the language of propositional or predicate logic, one usually begins by describing its syntax: the sorts of expressions that make up the language, and how they can be combined to make further expressions. In predicate logic, the basic sorts of expressions might include:

- Singular terms, such as the constants a, b, c.
- Predicates of various arity, such as a unary predicate F, a binary relation R, and so on.
- The truth-functional connectives: $\wedge, \vee, \neg$.
- The quantifiers, $\forall, \exists$.
- Complex sentences built from the above.

These are subject to clear rules about how they can be combined: predicates may be combined with singular terms to form sentences, such as Fa, and connectives with sentences, such as $\neg Fa$, and so on. But one cannot apply a predicate to a connective or a connective to a singular term, for example: $F\neg$ and $\neg a$ are not well-formed.

Type theory may be seen as a way of generalizing and systematizing these relationships. Its basic insight is to classify expressions in a language by their 'inputs'—the sorts of expressions they can be grammatically applied to—and their 'outputs'—the sorts of expression that result from such an application.

More concretely, the type of an expression is simply something that specifies the types of expressions the expression can be applied to, or take as arguments (if any), and what the type of the result is. For instance, a unary operator expression like *it is not the case that*, will take something with the type of a sentence as argument—e.g. *London is in France*—and yield something with the type of a sentence as a result: *it is not the case that London is in France*. A unary predicate—e.g. *is tall*—would take something with the type of a singular term, *John*, and yield a sentence, *John is tall*. Of course, there are some expressions in language that never take arguments themselves, expressions that only ever occur as arguments. For example, it

DOI: 10.4324/9781003039181-3

is never grammatical to apply a singular term or a sentence to another expression. Singular terms and sentences have this special status in pretty much all formal languages studied by logicians.[1] (A notable exception will be discussed at the end of the chapter.)

Thus type theory may be expressed in terms of two basic types: e, the type assigned to singular terms, and t, the type assigned to sentences. Mnemonically, think of e in terms of expressions standing for *entities*, and t as the *truth evaluable* expressions.[2] For any types, σ and τ, there is another type $(\sigma \to \tau)$, which corresponds to the expressions that may be applied to a single argument expression of type σ and produce an expression of type τ.

Definition 1.1 (Types)**.** *The types are defined as follows:*

- *e and t are types.*
- *If σ and τ are types, so is $(\sigma \to \tau)$.*

The types e and t are called base types*, and types of the form $(\sigma \to \tau)$ are* functional types*.*

Convention 1.1 (Has type)**.** *Depending on the context we write $: \sigma$ as short for 'has type σ' or ', of type σ,'.*

In what follows we adopt the convention of associating type brackets to the right, so that, $(\sigma \to (\tau \to \rho)$ can be shortened to $(\sigma \to \tau \to \rho)$ (but, to prevent ambiguity, $((\sigma \to \tau) \to \rho)$ cannot be so shortened). We also follow the usual convention of omitting the outermost brackets, writing simply $\sigma \to \tau \to \rho$. More generally, this means that we can just write $\sigma_1 \to \sigma_2 \to \cdots \to \sigma_n \to \tau$ instead of $(\sigma_1 \to (\sigma_2 \to \cdots \to (\sigma_n \to \tau) \ldots))$.

Exercise 1.1. *Using Definition 1.1, show that $(t \to ((e \to t) \to t))$ is a type, and rewrite it applying the bracketing conventions.*

To illustrate the intended interpretations of the types, we can try to assign approximate types to expressions in English. (Note: this correspondence between simple type theory and English grammar is not perfect, but it will suffice for the present pedagogical purposes.[3]) Starting with the base types, we have:

John, *London*: type e

John is tall, *London is not in France*: type t

This falls out of our identification of e with singular terms and t with sentences. As discussed earlier, a unary predicate, like *is tall*, is something that can be applied to a singular term to produce a sentence.

is tall: type $e \to t$

Generally, $e \to t$ corresponds to the type of unary predicates. Similarly, operator expressions combine with sentences to form sentences. For instance, the negation operator in English might be associated with a type:

it is not the case that: type $t \to t$

A conspicuous limitation of our type theory is that all expressions of functional type are unary: they combine with a single argument to produce something else. Yet many expressions in English, such as transitive verbs, appear to be relational, taking multiple arguments; e.g. *loves* and *is taller than*. The restriction to unary functional types isn't a real limitation, however. We can represent binary predicates by treating them as unary devices that combine with a single expression to form a unary predicate. The binary relation *loves*, which combines with two singular terms to form a sentence, may then be associated with a type as follows:

loves: type $e \to e \to t$.

Recall that this type is short for $(e \to (e \to t))$, given our bracketing conventions. So *loves* combines with something of type e, say *John*, to form a predicate, *loves John*, which can then be combined with another singular term, *Mary*, to form a sentence. It is sometimes helpful to depict the construction of a complex expression as follows:

$$\frac{M : \sigma \qquad N : \tau}{P : \rho}$$

This should be read as follows: if you have the things above the line, then you can make the things below the line using the syntactic rules. Thus we may depict the construction of the above sentence like this:

$$\cfrac{\cfrac{\text{loves} : e \to e \to t \qquad \text{John} : e}{\text{loves John} : e \to t} \qquad \text{Mary} : e}{\text{Mary loves John} : t}$$

The specific syntactic rules that license this derivation will be given later.

This way of representing binary relations—as a device that combines with a name to form a unary predicate—is often called *currying* (it is analogous to an operation on binary functions often referred to by the same name). This operation was introduced by Gottlob Frege, and was later employed by Moses Schönfinkel and, most extensively, by Haskell Curry from which it gets its name.[4]

The representation of binary relations via currying turns out to be beneficial, for it allows us to construct the complex predicate *loves John*. We would not be able to do this straightforwardly if *loves* had to take both of its arguments simultaneously. This corresponds to the fact that English, and other natural languages, are *binary branching*: every complex expression decomposes into two subexpressions, an expression taking a single argument, and that single argument, respectively.

Remark 1.1. Notice a subtlety that is not captured in our type theory, namely that *loves* combines with its first argument, *John*, by prefixing (adjoining on the left), and to its second argument, *Mary* by postfixing it to the expression (adjoining on the right). This suggests that a completely faithful representation of English would have at least two distinct ways of applying an expression to another one, and indeed, more complex type theories can be developed to capture this.

Our gloss of $\sigma \to \tau$ as the type of expressions that yield things of type τ when applied to things of type σ becomes ambiguous once there are two ways of applying something to an expression of type σ. As a result, there are two functional types associated with a given

σ and τ: $\sigma \rightarrow \tau$ and $\tau \leftarrow \sigma$. Since *loves* combines with a name on the right to form an expression that combines with a name on the left to form a sentence, it would receive the type $e \rightarrow (t \leftarrow e)$ in this system. See, e.g., Lambek (1958), Moortgat (2020) for further details.

Similarly, binary connectives like *and* may be assigned curried types:

and: type $t \rightarrow t \rightarrow t$

Note, of course, the operation of currying extends to relations of arbitrary arity. The general recipe is this: assuming we know how to curry n-ary predicates in this type system, $n+1$-ary predicates can be identified with unary devices that yield a curried of n-ary predicates when applied to an argument of the appropriate type. Our bracketing conventions are designed to make it very easy to see what types the arguments of a curried n-ary relation are. Any type contained in brackets will signal an argument to the relation, as well as any base types not appearing in brackets in the function's type, apart from the right-most. For instance, an expression of type $t \rightarrow (e \rightarrow t) \rightarrow e \rightarrow (t \rightarrow t) \rightarrow t$ is a curried four place relation that takes arguments of type t, $e \rightarrow t$, e and $t \rightarrow t$, in that order, to produce a result of type t

Finally, we can consider quantificational expressions, like *everyone* or *some dog*. We know that *barks* is a predicate, of type $e \rightarrow t$, and may be combined with *some dog* to form sentence, *some dog barks*. It follows that *some dog* can have exactly one of two types: it could be occurring as the argument of *barks*, and thus have type e, or it could take *barks* as its argument, and thus have type $(e \rightarrow t) \rightarrow t$. There are well-known problems with treating quantifier phrases as singular terms: for instance, it would seem to license the evidently fallacious inference from *nothing travels faster than light* to *something (namely nothing) travels faster than light* by the principle of existential introduction for singular terms from predicate logic. This leaves us with the other option:

everyone, *some dog*: type $(e \rightarrow t) \rightarrow t$

We will say more about the representation of quantifiers in type theory shortly.

Exercise 1.2.

a. Consider expressions like not, sometimes *and* clearly. *These expressions combine with a predicate, like* is tall, *to form another predicate, like* is not tall. *What type should we associate with these expressions, given the approximate association of types to expressions of English outlined above?*

b. What type should be associated with the English expression believes that*?*

Predicate logic generalizes propositional logic in one way, by introducing new types of expressions like singular terms and predicates. Modal logic extends propositional logic in a different way by adding new operator expressions and connectives. But there are many other ways to generalize. Once we have predicates, we might countenance expressions that combine with predicates to form new predicates: *predicate modifiers*, that have type $(e \rightarrow t) \rightarrow (e \rightarrow t)$.[5] Or, once we have quantifiers, we might think about determiners: *some dog* looks like it decomposes into *some* and *dog*, where *some* must have the type of something that takes a predicate to a quantifier phrase.

some: type $(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$

More exotic types are not occupied in English, but seem like they could be, either by a more expressive natural language or by introducing them to a formal language. For instance, one might investigate operations that take operator expressions to operator expressions—the analogue of predicate modifiers for operator expressions. Type theory is the appropriate setting for studying all these generalizations: it abstracts the general rules that appear to govern all the particular cases we have mentioned above.

loves, *some* and *and* may all be thought of as relational expressions. In the first case, we have a relational expression accepting two individual terms, say *Mary* and *John*, to produce sentence stating a relation between Mary and John. *some* takes two predicates, *F* and *G*, and states the relation that *some F Gs*. And similarly for conjunction. More generally, a (binary) relational expression at type σ is an expression of type $\sigma \to \sigma \to t$.

Our examples considered thus far involve relational expressions taking two things of the same type as arguments, but this is not an essential restriction. For instance, *believes that* might be thought of as a relational expression between something of type *e* and of type *t*. For any two types, σ, τ one can consider relational expressions of type $\sigma \to \tau \to t$. We might also consider relational expressions of higher arity: for any sequence of types $\sigma_1, \ldots, \sigma_n$, there is a relational type $\sigma_1 \to \cdots \to \sigma_n \to t$. This motivates the following definition:

Definition 1.2 (Relational Types)**.** *The relational types are defined as follows:*

e and t are relational types.

If σ a relational type, and τ is a relational type other than e, then $\sigma \to \tau$ is a relational type.

Exercise 1.3. *Which of the following are relational types?*

a. $e \to t \to e$

b. $e \to e$

c. $t \to e$

d. $t \to e \to t$

e. $(t \to e) \to t$

f. $(e \to t) \to t$

Equivalently, a relational type is either *e* or of the form $\sigma_1 \to \cdots \to \sigma_n \to t$ where $\sigma_1, \ldots, \sigma_n$ are relational types (the 0-ary case ensuring that *t* is a relational type). It is common to write $(\sigma_1, \ldots, \sigma_n)$ for the relational type $\sigma_1 \to \cdots \to \sigma_n \to t$.[6] The reader should convince themselves that relational types, apart from *e* itself, never end with an *e*.

It's worth noting that an expression of an arbitrary type may be thought of as either a curried relational type or a curried multi-argument functional type taking its arguments to something of type *e*. Thus, all types represent expressions taking multiple arguments (possibly none) to form an expression that is either a sentence or a singular term.

Theorem 1.1. *Every type may uniquely be written in the form $\sigma_1 \to \cdots \to \sigma_n \to t$ or $\sigma_1 \to \cdots \to \sigma_n \to e$.*

Proof. This is proven by induction on the structure of types. The base types *e* and *t* are trivially in this form (where $n = 0$).

Suppose σ and τ can be uniquely written in this form. In particular, for some $\tau_1, \ldots, \tau_n$, $\tau = \tau_1 \to \cdots \to \tau_n \to t$ or $\tau = \tau_1 \to \cdots \to \tau_n \to e$. Thus the functional type $\sigma \to \tau$ is $\sigma \to \tau_1 \to \cdots \to \tau_n \to t$ or $\sigma \to \tau_1 \to \cdots \to \tau_n \to e$, and so also has the required

form. Moreover, the source of any functional type is unique (in this case σ), so the above decomposition is unique. □

1.2 Typed languages

To illustrate the generality of simple type theory, let us begin by surveying some of the logical languages that can be represented within it. The specification of a language determines a specification of the primitive operations—or *constants*—that can be used to build sentences. In predicate logic, and elsewhere, the word 'constant' is often reserved for singular terms. In type theory, constants may have any type whatsoever:

Definition 1.3 (Signature)**.** *A* signature*, Σ, is a collection of sets, Σ^σ, for each type σ.*
The elements of Σ^σ called the constants of type σ.

To illustrate how various more familiar systems are particular instances of typed languages, let us consider the signatures of some well-known systems, beginning with the propositional calculus.

Example 1.1 (Propositional Logic)**.** *The signature of propositional logic, with propositional letters $p_0, p_1, \ldots$.*

- $\Sigma^t = \{p_0, p_1, \ldots\} \cup \{\top, \bot\}$
- $\Sigma^{t \to t} = \{\neg\}$
- $\Sigma^{t \to t \to t} = \{\wedge, \vee, \to, \leftrightarrow\}$

Following our earlier discussion, connectives like $\wedge, \to$ and so on are represented by curried binary operations on sentences of type $t \to t \to t$.

Modal logic, as described in, e.g. Cresswell and Hughes (1996), has the following signature.

Example 1.2 (Modal Logic)**.** *The signature of modal logic is like propositional logic except that $\Sigma^{t \to t}$ contains an extra constant, $\Box$.*

Quantified logics, like predicate logic, are a bit more subtle because they contain expressions that bind variables. We will discuss how this works in a little more detail later. But for now, we can merely note that we can treat the universal and existential quantifiers as we treated the English quantificational phrases *everything* and *some dog*: as *higher-order* predicates, expressions of type $(e \to t) \to t$. Intuitively, the universal quantifier combines with a unary predicate, F, of type $e \to t$ to form the sentence (type t) stating that everything is F; the existential quantifier forms the sentence stating that something is F.

Let us write $e^n \to t$ as short for $\underbrace{e \to \cdots \to e}_{n} \to t$, the type of n-ary predicates.

Example 1.3 (Predicate Logic)**.** *Consider a first-order language that has, for each arity, predicates $P^n_1, \ldots, P^n_{k_n}$ of that arity, and constants $c_1, c_2, \ldots$. It may be specified by the following typed signature:*

- $\Sigma^e = \{c_1, c_2, \ldots\}$
- $\Sigma^{e^n \to t}$ *contains* $P^n_1, \ldots, P^n_{k_n}$

- $\Sigma^{e\to e\to t}$ *also contains* $=$
- $\Sigma^{t\to t} = \{\neg\}$
- $\Sigma^{t\to t\to t} = \{\wedge, \vee, \to, \leftrightarrow\}$
- $\Sigma^{(e\to t)\to t} = \{\forall, \exists\}$

Example 1.4 (Predicate Logic with Function Symbols)**.** *Like predicate logic, except for each n we might have a supply of n-ary function symbols* $f \in \Sigma^{e^n\to e}$.

As a general notational shorthand, we will write $c : \sigma$ for $c \in \Sigma^\sigma$, and we will continue to pronounce this *c has type* σ.

Beyond these common examples, there are many other signatures we might consider that extend propositional and predicate logic: people have considered adding new first-order quantifiers, such as *there are finitely many* : $(e \to t) \to t$, determiners like *most*: $(e \to t) \to (e \to t) \to t$, new connectives such as the counterfactual conditional (of type $t \to t \to t$). It would be foolish to attempt to survey them all. But one signature will play a particularly important role in this book: the signature of *higher-order logic*. The main feature of this signature is that it introduces quantifiers that stand to other types as first-order quantifiers stand to type e, in the sense explained in Section 0.3. Exploiting the representation of first-order quantifiers as operations of type $(e \to t) \to t$, we see that the analogue for type σ should combine with a predicate of type $\sigma \to t$ to form a sentence: i.e. its type should be $(\sigma \to t) \to t$. When σ is t, for instance, the operation combines with an operator expression to form a sentence and thus may be thought of as a quantifier that quantifies into sentence position, in the same sense that a first-order quantifier quantifies into the position a singular term occupies.

Example 1.5 (Higher-order Logic)**.** *Like the signature for first-order logic except there are generalizations of identity and the quantifiers at each type. A higher-order signature may have non-logical constants of any type, in addition to the logical constants below. The signature of* pure *higher-order logic, which we will refer to as* Λ *throughout this text, refers to the signature containing only the logical constants:*

- $\Lambda^t = \{\top, \bot\}$
- $\Lambda^{\sigma\to\sigma\to t} = \{=_\sigma\}$
- $\Lambda^{t\to t} = \{\neg\}$
- $\Lambda^{t\to t\to t} = \{\wedge, \vee, \to, \leftrightarrow\}$
- $\Lambda^{(\sigma\to t)\to t} = \{\forall_\sigma, \exists_\sigma\}$

Higher-order identities will be discussed further in Chapter 4. $=_t$, for instance, is a binary connective and thus behaves grammatically like $\wedge$, combining with two sentences to form a sentence. As one might expect, the list of logical constants has some redundancy, in the sense that in the languages we'll consider some of the connectives may be defined out of the others. Some cases of definability are familiar, such as the definability of $\wedge$ from $\neg$ and $\vee$, and others less so, for instance we'll see how $=_e$ may be defined from $\forall_{e\to t}$ with the propositional connectives. We'll return to this in Chapter 4. It's also worth noting that the sense in which one thing may be defined from another is weak—the defined notions have the right extensional properties, but may not be identical to the target operation without

further assumptions—so that in some contexts it is more perspicuous to work with the larger signature.

Above we have focused exclusively on *logical signatures*: signatures for languages that can be used to make statements that can be evaluated for truth and falsity. But to illustrate the generality of the type-theoretic framework, let's consider a non-logical signature. The sorts of typed languages that computer scientists are interested in are not logical languages but *programming languages*, whose expressions denote programs. The appearance of type-theoretic ideas is most conspicuous in functional programming languages like Haskell and Lisp, but other more traditional programming languages, such as Python, have adopted elements of functional programming languages. The programming language PCF (Plotkin, 1977) is a simplified version of a functional programming language: the primitives include operations for manipulating natural numbers (0, the successor and predecessor operations) operations for forming conditional statements (if_σ which takes a number and two other values, *a* and *b* of type σ, and outputs *a* if the value of the number is 0, and *b* otherwise), and an operation for recursion, fix_σ, which produces a fixed point of a program of type $\sigma \to \sigma$:[7]

Example 1.6 (Programming Computable Functions (PCF))**.** *In this example, we interpret the e as the type of natural numbers* $\mathbb{N}$*. The signature of PCF is defined as follows*

- $\Sigma^e = \{0\}$
- $\Sigma^{e \to e} = \{\mathrm{succ}, \mathrm{pred}\}$
- $\Sigma^{e \to \sigma \to \sigma \to \sigma} = \{\mathrm{if}_\sigma\}$
- $\Sigma^{(\sigma \to \sigma) \to \sigma} = \{\mathrm{fix}_\sigma\}$

Given a signature, Σ, a *language* based on that signature will consist of a collection of expressions, or *terms*, of each type which are built out of the constants of Σ. Our target class of languages will be those based on the 'simply typed λ-calculus'. But before we present that system let us start off with a simpler class of languages, that has exactly one way of building complex expressions: *application*.

Definition 1.4 (Typed Applicative Languages)**.** *Given a signature* Σ*, the typed applicative language,* $\mathcal{A}(\Sigma)$*, based on* Σ *is a typed collection of sets,* $\mathcal{A}^\sigma(\Sigma)$*, called the* terms *of type* σ*. Writing* $M : \sigma$ *as short for* $M \in \mathcal{A}^\sigma(\Sigma)$*, we may define the terms of type* σ*, for each* σ*, simultaneously as follows:*

- $c : \sigma$ *whenever* $c \in \Sigma^\sigma$*.*
- *If* $M : \sigma \to \tau$ *and* $N : \sigma$ *then* $(MN) : \tau$*.*

only terms that can be built by the above rules are terms of the language $\mathcal{A}(\Sigma)$*.*

Notice that many familiar logical systems focus on specifying the well-formed sentences. In type theory, sentences have no special status. The focus on sentences is due to the fact that in many common systems, sentences are the only syntactic categories containing complex expressions. In predicate logic, for instance, there is no way to build complex predicates.[8] If you don't have function symbols, there is no way to build complex singular terms. Similarly, the only connectives and quantifiers are the truth-functional and universal and existential quantifiers respectively, all of which are simple.

Comprehension Check 1.1. *Pick one of the signatures from the examples above, and describe a complex term that is not of type t.*

In proving that a term is well-formed or has a particular type, it is convenient to reuse our earlier notation for representing construction rules. Things above the line represent things you have already constructed, and things below represent the things you may construct from them. Our two rules become:

$$\frac{}{c : \sigma}\ c \in \Sigma^{\sigma} \qquad \frac{M : \sigma \to \tau \qquad N : \sigma}{(MN) : \tau}$$

For instance, suppose we are considering a signature for predicate logic that contains a binary predicate $L : e \to e \to t$, informally representing the *loving* relation and a constant a standing for Alice. We can represent the proof that $(\forall(Re))$ is a term, informally meaning that Alice loves everything, as follows:

$$\dfrac{\dfrac{R : e \to e \to t \qquad a : e}{(Ra) : e \to t} \qquad \forall : (e \to t) \to t}{(\forall(Ra)) : t}$$

Exercise 1.4. *Show that the following are terms of type t in the signature of pure higher-order logic.*

a. $\exists_t \neg$

b. $\forall_{t \to t} \forall_t$

c. $\exists_t (\wedge (\forall_{t \to t} \forall_t))$

Suppose that (i) a sufficient condition for $\forall_\sigma F$ to be false is if there is some term, $a : \sigma$, such that Fa is false, (ii) a sufficient condition for $\exists_\sigma F$ to be true is that there is some term, $a : \sigma$, such that Fa is true, (iii) there is a false sentence $\bot$, and (iv) the truth-functional connectives have their usual meaning. Determine the truth values of 1-3.

Exercise 1.5. *Show that the following are terms of type t (sentences) in the signature of propositional logic.*

a. $((\wedge p_1)p_4)$

b. $((\to p_1)((\vee p_1)p_4))$

c. $((\to (\neg p_1))((\to p_1)p_2))$

Note that unlike a typical presentation of the propositional calculus, we are able to construct complex expressions of type $t \to t$ using the rules.

Exercise 1.6. *Show that there is a complex term of type $t \to t$ in the signature of the propositional calculus. (That is, a term of type $t \to t$ that is not $\neg$.)*

Consider the sentence *Alice loves Bob*, as represented by the sentence $((La)b)$.

Convention 1.2 (Bracketing Conventions)**.** *We will omit left associated brackets. If $M : \sigma_1 \to \cdots \to \sigma_n \to b$, where $b = t$ or e, and $a_i : \sigma_i$ for $i = 1, \ldots, n$, then we may write $Ma_1 \ldots a_n$ as short for $((\ldots(Ma_1)a_2)\ldots a_n)$.*

Outer brackets may always be omitted.

Thus, for instance, by omitting the outer brackets from $((\wedge p_1)(\neg p_1))$ we get $(\wedge p_1)(\neg p_1)$, and then omitting left associated brackets we get $\wedge p_1(\neg p_1)$.

Observe that our bracketing conventions for terms are opposite to the bracketing conventions for types, where brackets associated to the right may be omitted. In that case, the convention made it easy to see from the look of a type what its arguments should be when considered as the type of a curried function. Similarly, if R is a curried n-ary functional term, and $a_1, \ldots, a_n$ are arguments of the appropriate type, then the result of applying R to its arguments has a particularly simple form with this convention: $Ra_1 \ldots a_n$.

It's worth noting that different conventions would avoid this mismatch. One might write types in the reverse order, writing for instance $\tau \leftarrow \sigma$ where we write $\sigma \to \tau$ (this is essentially the original convention of Church (1940)). Alternatively, one could leave the type conventions alone and write application in the opposite order, e.g. writing aF for the result of applying a functional term F to an argument a (the conventions employed by the mathematical collective, Bourbaki).

Exercise 1.7. *Rewrite the formulas from Exercise 1.5 using our bracketing conventions.*

Note that the usual conventions for relation symbols in predicate logic are inconsistent. For non-logical predicate symbols, we place the predicate before the arguments, writing things like Rab, yet for the identity relation, we write the relation between the arguments, $a = b$. The former is called the prefix notation, and the latter infix notation. Type theory consistently uses the prefix notation across the board: n-ary on symbols, of type $\sigma_1 \to \cdots \to \sigma_n \to t$, always appear prior to their arguments. However, the infix notation for certain logical symbols, like $=, \wedge, \vee, \to$ et cetera, is so prevalent that it would be perverse to insist that we disregard it.

Convention 1.3 (Infix notation)**.** *When* $* \in \{=, \to, \wedge, \to, \leftrightarrow\}$*, we write* $(M * N)$ *instead of* $((*M)N)$.

Exercise 1.8. *Rewrite the formulas from Exercise 1.5 in infix notation.*

Exercise 1.9. *Rewrite the following formulas in infix notation, in prefix notation.*

a. $(p_1 \to p_2)$

b. $(p_1 \to (p_2 \to (p_1 \wedge p_2)))$

c. $((p_1 \vee p_2) \to p_3) \to ((p_1 \to p_3) \vee (p_2 \to p_3))$

The following exercises are designed to familiarize the reader with our bracketing conventions

Exercise 1.10. *Suppose* $R : (e \to t) \to (t \to e \to e) \to e \to (t \to e) \to (e \to e) \to e$.

a. What is Rs arity, considered as a curried relation?

b. What are the types of its arguments, and what is the type of the result of applying this relation to those arguments?

c. Reinsert the brackets omitted from Rs type.

Exercise 1.11. *Suppose* $S : ((\sigma \to \tau) \to ((\tau \to \rho) \to (\sigma \to (\tau \to \rho))))$*. Apply our bracketing conventions to the type of S. What is its arity, as a curried function, that yields results in* ρ*, and what are the types of its arguments?*

Suppose that $H : e \to t$ and $G : t \to e \to e$ and F has the type of the operation in Exercise 1.10. What is the type of FHG? We can calculate by repeatedly using our rule for calculating

types under application:

$$\dfrac{\dfrac{F : (e \to t) \to (t \to e \to e) \to e \to (t \to e) \to (e \to e) \to t \qquad H : e \to t}{(FH) : (t \to e \to e) \to e \to (t \to e) \to (e \to e) \to t} \qquad G : t \to e \to e}{((FH)G) : e \to (t \to e) \to (e \to e) \to t}$$

Exercise 1.12. *Suppose* $S : (\sigma \to \tau) \to (\tau \to \rho) \to (\sigma \to (\tau \to \rho))$, $F : \sigma \to \tau$, $G : \tau \to \rho$, $a : \sigma$, $b : \tau$.

a. Reinsert the brackets from SFGab.

b. Derive, in a similar fashion to the above, the type of SFGab;

1.3 The concept *horse* problem

With some type theory under our belt, we are now in a good position to return to some issues, touched on in the introduction, concerning how we should talk about the meanings of typed languages.

It is tempting to treat the terms of a typed language as standing for special sorts of abstract individuals: sentences (terms of type t) as standing for "propositions", $e \to t$ type predicates as standing for "properties", $t \to t$ operator expressions for "operators", and so on. And it is similarly tempting to treat the higher-order quantifiers $\forall_t$, $\forall_{e \to t}$, $\forall_{t \to t}$ as ranging over propositions, properties and operators.

To do so flirts with confusion. One way in which this confusion can manifest itself is in the idea that all of the quantifiers $\forall_\sigma$ are restrictions of the first-order quantifier $\forall_e$. For instance, to say that $\forall_t$ quantifies over propositions only is just to say that it is a restriction of the first-order quantifier to a special sort of individual, a *proposition*. An analysis of the phrase 'every proposition' is offered in the following remark.

Remark 1.2. The English determiner 'every' combines with two predicates to form a sentence, and thus can be assigned the semantic type $(e \to t) \to (e \to t) \to t$. The expression 'is a proposition' combines grammatically with a name to form a sentence, so it can be assigned the type $e \to t$. Thus the phrase 'every proposition' has the same type as the first-order quantifier, $(e \to t) \to t$. It can combine with another first-order predicate to make a sentence, as in 'every proposition is abstract'.

Similarly, if we follow the tempting idea of treating $e \to t$ predicates as 'standing for' or 'referring to' a property, one might naturally be led into thinking that the truth of the claim:

Alice is tall.

commits us not only to the existence of the individual referred to in the sentence, namely Alice, but also to the existence of a property referred to, namely being tall. But this certainly seems like a mistake: a nominalist can believe that Alice is tall on the basis of her physical height, whilst rejecting the existence of properties.

Early practitioners of higher-order logic appeared to be sensitive to this confusion, but would often talk in ways that invite it. Frege, for instance, maintained that what the predicate 'is a horse' stands for cannot be picked out with a singular term. But this is a paradoxical way of phrasing things since 'what the predicate 'is a horse' stands for' is itself a singular term, and 'stands for' seems to be a binary predicate that takes two singular terms to form a

sentence.[9] In a similar spirit, Russell, in lecture 1 of 'The Philosophy of Logical Atomism', says that the sort of thing that stands to sentences as individuals stand to names cannot take the place of the subject of a predicate:[10]

> You cannot properly name a fact. The only thing you can do is to assert it, or deny it, or desire it, or will it, or wish it, or question it, but all those are things involving the whole proposition. You can never put the sort of thing that makes a proposition to be true or false in the position of a logical subject.

This too seems paradoxical since the expression 'the sort of thing that makes a proposition to be true or false' can be the subject of a predicate, as it indeed is in the above quotation.

Here is what Russell means: names (expressions of type e), can be said to stand for or to refer to things ('a name can just name a particular', Russell (1918), p. 11). The name 'Alice' refers to a certain individual Alice. But sentences (expressions of type t) do not refer. 'Refers to', 'stands for', etc. express a relationship between two individuals, a linguistic expression and its referent. Since they grammatically combine with singular terms, their type is $e \to e \to t$. According to Russell, the meaningfulness of a whole sentence, like 'Alice is tall', must be explained in terms of a different semantic primitive. You can assert that Alice is tall, you can wish that Alice were tall, and so on, but a sentence cannot 'refer to Alice is tall'. This is grammatical nonsense: a sentence, 'Alice is tall', is taking the place that grammatically must be filled with a name. One needs instead something that can take a whole sentence in its second argument place: one needs an expression of type $e \to t \to t$. I suggest we could approximate this notion using the complex English expression 'means that':[11]

x means that P

where x is a singular term, and P is a whole sentence. Letting x be the singular term 'the sentence 'Alice is tall'' and letting P be the sentence 'Alice is tall', we can say things like:

The sentence 'Alice is tall' means that Alice is tall

This should have the same sort of semantic explanatory power as an analogous claim about reference, such as 'Alice' refers to Alice.

Parallel considerations can be made about predicates and expressions of other types. The meaningfulness of a predicate must be stated using a term of type $e \to (e \to t) \to t$. Perhaps one of the following English expressions could approximate it:[12]

x means to F.
x ascribes F

Where x is a singular term, and F is a predicate. So letting x be the quotation name 'the predicate 'is tall'', and F the predicate (in the passive voice) 'be tall', we can say things like

The predicate 'is tall' means to be tall.
The predicate 'is tall' ascribes being tall.

More generally, we need an expression of type $e \to \sigma \to t$ to theorize about the meaningfulness of expressions of type σ.

Russell is explicit that we need an infinite number of semantic primitives for each semantic type ('a strictly infinite number of things that "meaning" may mean.' Russell (1918) , p. 12). He elaborates:

> The word "Socrates," you will say, means a certain man; the word "mortal" means a certain quality; and the sentence "Socrates is a mortal" means a certain fact. But these three sorts of meaning are entirely distinct, and you will get into the most hopeless contradictions if you think the word 'meaning' has the same meaning in each of these three cases. It is very important not to suppose that [. . .] there is just one sort of relation of the symbol to what is symbolized. A name would be a proper symbol to use for a person, a sentence [. . .] is the proper symbol for a fact.

Or, in 'Logical Atomism' (Russell (1918) p. 141):

> [. . .] Meaning is a different relation for different types. The way to mean a fact is to assert it; the way to mean a simple is to name it. Obviously naming is different from asserting and similar differences exist where more advanced types are concerned, though language has no means of expressing the differences.

These quotes illustrate the primary awkwardness that arises when talking about typed languages in English. The expressions 'is a fact', 'is a proposition', 'is a property', 'is a concept', and so on are all first-order $e \to t$ predicates, since they combine grammatically with a singular term to make a sentence—'Alice is a proposition' is a grammatical, albeit false, sentence. So expressing it this way makes it tempting to read Russell as talking about a special kind of individual when he talks about facts, or to read Frege as talking about a special sort of individual when he talks about concepts. (Russell is, of course, fully aware of this: he also writes 'When I say "facts cannot be named", this is, strictly speaking, nonsense. What can be said without falling into nonsense is: "The symbol for a fact is not a name".' Russell (1918) p. 141.)

The proper way for Frege and Russell to express themselves would not be to employ singular quantification restricted to propositions or concepts, but to use quantification into sentence position or predicate position (that is, to employ the primitive quantifiers $\forall_t$, or $\forall_{e \to t}$). Of course, we do not have such expressions in English. We can quantify into *some* positions other than the position of a singular term in English, but there are some grammatical positions, such as sentence position, for which English has no corresponding quantifier. But as we have noted already, there is no perfect solution to the practical problem of paraphrasing higher-order quantification in English. We have thus, somewhat reluctantly, followed Frege and Russell in using first-order quantificational expressions of English, like 'every proposition', to indicate sentences of a higher-order language that would involve quantification into sentence position. Let's explicitly codify this with the following loose convention:

Convention 1.4. *For each type σ, we shall use the first-order predicate 'type σ entity' for the sorts of things that expressions of type σ "stand for". When σ is one of e, t, $e \to t$, $e \to e \to t$ or $t \to t$ we will use the nouns 'individual', 'proposition', 'property', 'relation' and 'operator'. Occasionally we will 'property' and 'relation' more generally for type $\sigma \to t$ and $\sigma \to \sigma \to t$ entities.*

When we provide an informal English gloss using these predicates we indicate a unique higher-order sentence that does not involve these first-order predicates. Thus expressions like 'every proposition', 'some property', 'all operators', and so on, are to be understood as indicating a sentence of a higher-order language employing the quantifiers $\forall_t$, $\forall_{e \to t}$, $\forall_{t \to t}$, and so on.

Note, of course, that in even stating this convention I have begun to employ the convention; however I think the proposal is clear enough.

1.4 Alternative type systems

The types we have described in this chapter are called 'functional types': like functions, which have a unique 'domain' and 'codomain', every expression has a unique source and target—i.e. a single type of expressions corresponding to the things those expressions may combine with, and the type of expression that results from such a combination. Although this is the type system we will pursue throughout this book, there are several variations one might consider that we will now briefly survey. We've assumed there are two base types, e and t. However, one could consider systems that contain fewer or more base types. We've also assumed that there is only one type constructor, $\rightarrow$, for making functional types, but we could imagine having more or different type constructors. Here we will briefly look at ways of generalizing along both of these axes.

In fact, the first formulation of higher-order logic due to Frege had only one base type.[13] Frege thought there were only two propositions, the true and the false, and that they are just two entities among a broader class of individuals of the more familiar sort. Since the things ordinarily classified as propositions are a subclass of the individuals it makes sense to identify the Fregean base type with type e rather than type t. We could represent Frege's types as a subset of our functional type system as follows:

e is a Fregean type.

If σ and τ are Fregean types then so is $\sigma \rightarrow \tau$.

Thus, for Frege, the addition operation $+ : e \rightarrow e \rightarrow e$ has exactly the same type as the conjunction connective $\wedge : e \rightarrow e \rightarrow e$, which in turn has the same type as a binary relation $R : e \rightarrow e \rightarrow e$. Frege's decision to blur together the e and t types has some interesting consequences. Of course, it fails to predict the most basic facts about grammaticality: for instance the distinction between the grammatical sentences 'it's not the case that Gottlob talks' and 'Gottlob talks', and the ungrammatical 'it's not the case that Gottlob' and 'Gottlob talks walks'. For Frege, these are in equal standing since 'Gottlob' and 'Gottlob talks' have the same grammatical type. Of course, Frege was not doing natural language grammar, he was creating a system for doing logic and metaphysics. In this respect, then, Frege's metaphysics is radically different from the metaphysical picture implicit in the use of modern logical systems, such as first-order logic, as well as natural language.

The view that there is only a single base type must also have a way to make sense of the distinction between asserting and naming. This book is composed of a sequence of sentences, which according to Frege is just a sequence of proper names for truth values. Some proper names, such as 'Gottlob', are used exclusively for naming objects, whereas expressions referring to truth values can be used to name a truth value, but can also be used to *assert* them. If you were to simply write a sentence like 'it's raining', according to Frege, it is not clear which of the two things you are doing. To assert it you must do something more than name, which compels Frege's to introduce his infamous judgement stroke: to indicate that a sentence is being used to assert a truth value, rather than name it, you write $\vdash A$ instead of A.[14]

The function of this symbol is a little perplexing. One might naively hope to read $\vdash$ as 'I assert that ...' so that '$\vdash$ it's raining' simply means 'I assert that it's raining'. But note that on this interpretation $\vdash A$ is just another sentence, and so is no different to a name for another truth value, the reference of which depends on whether the truth value A refers to has been asserted. A better interpretation is that $\vdash$ functions like a question mark, in that it changes the illocutionary force of an utterance of a Fregean name. Much like a question mark can turn a statement into a question (the door should be open?) or certain intonation can turn a statement into a command (*the door should be closed!*), the judgement stroke indicates a naming expression is being used to make a statement. There is, however, a more radical interpretation of Frege's judgement stroke—arguably found in Curry and Feys (1958) (Chapter 1 §E2)—that involves interpreting him in the modern type system with two base types, e and t. According to this interpretation, most of the expressions in the Frege (1879) and Frege (1893) are function symbols and singular terms, and the only sentences to be found in these works are of the form 't is the True', $\vdash t$, where t is one of these singular terms. That is to say, we treat $\vdash$ as the only genuine predicate of type in the system—i.e. the only expression with type $e \to t$—and indeed, the only constant with a type involving t. We cannot conjoin or negate sentences in the traditional sense, although this effect can be achieved inside the scope of the $\vdash$ predicate using suitable function symbols. Readers interested in what can be done in Frege's type system should consult Aczel (1980), who exploits features of Frege's type system and his account of judgement to provide a natural response to Russell's paradox. Kripke (2014) explores a Fregean treatment of first-order logic which does away with the distinction between formulas and terms.

One could also have the view that t is the only base type. The thought here is that, unlike Frege, we could posit a greater number of propositions and identify the things we ordinarily think of as individuals as being special sorts of proposition.[15] It is also possible to consider type systems with more than two base types. To account for the syntactic behaviour of English it is often thought to be necessary to posit three base types to account for the proper ways in which phrases combine: N, NP and S, corresponding to nouns, noun phrases, and sentences. Linguists typically do not take this three-way division metaphysically seriously, in the sense that they will distinguish the *syntactic* type of a natural language phrase from its *semantic type*: a noun like 'cat' syntactically has the base type N, but has a non-base semantic type $e \to t$, which is the same semantic type as for a predicate like 'is a cat', which is not a noun and so has a different syntactic type from 'cat'. Nonetheless, one might consider a metaphysical view in which reality more closely mirrors the syntax of English and in which there is a deep and important difference between the entities in reality corresponding to nouns and the entities corresponding to predicates. Montague (1973) is another example of a system with three base types. Montague follows Frege in holding that there are only two propositions and accounts for intensionality by positing a third base type, s, of possible worlds. Adopting an idea of Frege's, developed in Church (1951), every expression is associated with a plain reference in type σ but also a 'sense' of type $s \to \sigma$ which can be the reference of a term in an intensional context.

In the spirit of probing the assumptions behind the present type system, it's very natural to ask why there must be any base types at all. Indeed, there is no *formal* reason a typed language must have base types. We have assumed that the types of predicates, operator expressions, and so on are complex because they can be applied to other expressions, whereas the types of names and sentences are simple because they cannot. But perhaps this is simply because English does not contain any expressions to which sentences and names

can be applied, and so perhaps it is possible to extend English with expressions to which sentences and names can be applied. Clearly, sentences can be combined in some way with expressions in the original type system—namely expressions of type $t \to \tau$—but we could rule this possibility out by checking that these new expressions don't combine with any predicates with types of the form $(t \to \tau) \to t$. Once we have seen that it is possible that sentences have a complex type of the form $\sigma \to \tau$, it is similarly coherent to suppose that σ and τ are complex, and this can be continued ad infinitum circumventing the need for things to bottom out with base types. Let's introduce a more general notion of a type system that allows for this sort of non-well-foundedness.

Definition 1.5 (Generalized functional type system)**.** *A generalized type system consists of:*

A set T of types.
A binary function $\to$: $T \times T \to T$ subject to the constraint that:

$(\sigma \to \tau) = (\sigma' \to \tau')$ *if and only if* $\sigma = \sigma'$ *and* $\tau = \tau'$ *for every* $\sigma, \sigma', \tau, \tau' \in T$.

The functional type system of this chapter satisfies these conditions, but generalized type systems allow for non-well-founded types such as circles—for instance, one can have a types satisfying equations like $\sigma = (\sigma \to \tau)$, $\sigma = (\sigma \to \sigma)$ and so on—as well as infinite descending chains. Types of the former kind are sometimes studied by computer scientists interested in 'recursive types'. Types of the form $\sigma = (\sigma \to \sigma)$ have special significance in relation to what's known as the untyped λ-calculus.[16] For more on recursive type see Mitchell (1996) Sections 2.6.3 and 7.4, and for the untyped λ-calculus the reader should consult Hindley and Seldin (2008).

Another dimension along which we might generalize is to introduce other type constructors as well as, or instead of $\to$. In Remark 1.1, we already considered adding another type constructor $\leftarrow$ to account for the two-sided behaviour of English predicates. This idea originated with Lambek (1958) and has mainly been used in linguistics to model natural language syntax, in which it is not given a metaphysically serious interpretation. Moortgat (2020) is a good reference. Another common generalization of the functional type theory in this book is to add product types—indeed, we will actually need product types in Chapter 17. These behave much like the $A \times B$ notation from set theory for representing the set of ordered pairs (a, b) where $a \in A$ and $b \in B$. One can obtain product types by adding to the rules in Section 1.1 a clause:

If σ and τ are types, then so is $(\sigma \times \tau)$.

It is important not to think of entities of type $(\sigma \times \tau)$ as ordered pairs, since ordered pairs are special sorts of sets, which are in turn individuals. Nonetheless, the formal analogy with ordered pairs can be helpful. When a typed language has product types it is natural to consider signatures containing constants that intuitively can be interpreted as returning the first and second coordinate of an ordered pair, and an operation that given two entities of type σ and τ returns the pair of them:

$\pi_1 : \sigma \times \tau \to \sigma$
$\pi_2 : \sigma \times \tau \to \tau$
$p : \sigma \to \tau \to (\sigma \times \tau)$

Product types give us an alternative way to represent binary relations, for now a binary relation can also be treated as having type $R : (e \times e) \to t$.

The treatment of relations in functional type theory is in fact one point where we made a fairly substantive representational decision. Since every expression with a functional type has a unique source and target, we treated binary relations as higher-order functions that take one argument at a time. Note that even when we add product types to the functional type system, all expressions combine with at most one argument, which will now sometimes be a pair entity. There is an alternative type system that treats n-ary relations as taking all their arguments at once. Given types $\sigma_1, \ldots, \sigma_n$ it is possible to form a type of n-ary relations, $(\sigma_1, \ldots, \sigma_n)$ between entities of these types.[17] (Note that above we used the notation $(\sigma_1, \ldots, \sigma_n)$ as an abbreviation for a specific functional type, $\sigma_1 \to \cdots \to \sigma_n \to t$; in this section, we are temporarily reusing the notation.) Indeed, once you have the resources to make relational types like these, you can make all the functional relational types (see Definition 1.2), so this type constructor is usually taken to replace the functional type constructor $\to$.

Definition 1.6 (Relational type system)**.** *The relational types are specified as follows*

e is a relational type.

If $\sigma_1, \ldots, \sigma_n$ are relational types, then so is $(\sigma_1, \ldots, \sigma_n)$.

Observe that type t is replaced by the type $()$ obtained by setting $n = 0$ in the second rule: sentences are 0-ary predicates.

A signature, Σ, for a relational type system may be defined exactly as before, except that constants now have relational types, not functional types. Since this type of system no longer has functional types, the rules for combining expressions no longer apply. Instead, we have the rules:

$$\frac{}{c : \sigma}\ c \in \Sigma^{\sigma} \qquad \frac{M : (\sigma_1 \ldots \sigma_k) \qquad N_1 : \sigma_1, \ldots, N_k : \sigma_k}{(MN_1 \ldots N_k) : ()}$$

Observe that in a relational type system with this as the only rule the only complex expressions are sentences; it is possible to combine a relationally typed language with the λ-device introduced in the next chapter to form complex expressions with other types.[18]

There is a straightforward way in which the types of the relational type system correspond with the functional relational types: since $(\sigma_1, \ldots, \sigma_n)$ represents an n-ary relation between entities of type $\sigma_1, \ldots, \sigma_n$, it is serving the same function as the functional type $\sigma_1 \to \cdots \to \sigma_n \to t$. Similarly the result of applying a relation M of relational type $(\sigma_1, \ldots, \sigma_k)$ to some arguments to get $(MN_1 \ldots N_k)$ is equivalent to successively applying a functionally typed term M' corresponding to M to terms $N'_1, \ldots, N'_k$ corresponding to $N_1, \ldots, N_k$, yielding $(\ldots((M'N'_1)N'_2)\ldots N'_k)$. On this basis, we can set up a translation from a relationally typed applicative language to a functionally typed one. It is not possible to directly translate in the other direction because a functionally typed signature can have non-relational types (i.e. types ending in an e), and also because in a functionally typed language we can fill the first k arguments of an n-ary relation leaving an $n - k$-ary relation behind, whereas the only rule we've provided for a relationally typed applicative language is for filling all of the argument places of a relation. To deal with the second issue we could extend the rules for combining relations in relational type theory to allow one to fill the first k arguments of an

n-ary relation, where $1 \leq k \leq n$:

$$\frac{M : (\sigma_1, \ldots, \sigma_n) \qquad N_1 : \sigma_1, \ldots, N_k : \sigma_k}{(MN_1 \ldots N_k) : (\sigma_{k+1}, \ldots, \sigma_n)}$$

The first problem can be circumvented by restricting to functionally typed signatures that only contain constants with relational types (i.e. functional types ending in a t). The terms in any such functional language will then all have relational types, and we can on this basis translate back into a relational signature with the above more general rule of application. A rigorous specification of this translation, in a context where the relational and functional type theorists also have λ-terms, can be found in Dorr (2016) Appendix A1 and A2.

Another historically important complication of the type system we have considered in this chapter is the ramified theory of types of Whitehead and Russell (1910-1913). According to this theory, the grammatical types are individuated not only by their inputs and outputs but come stratified into infinitely many different 'levels', indexed by the natural numbers. This is usually developed as a variant of a relational type system: there is a ramified type of individuals, e, of level 0, and whenever $\sigma_1 \ldots \sigma_n$ are ramified types, there is a ramified type $(\sigma_1, \ldots, \sigma_n)/n$ of level n for $n \geq 1$. The relationship between the type of an expression and the sorts of things it can be applied to becomes more complex, and there are a plethora of ways it can be developed.[19] Ramified type theories quickly fell out of favour due to the fact that they were cumbersome and appeared to be unnecessarily prohibitive: systematic theorizing is nearly impossible in these frameworks.[20] For modern expositions of ramified type theory the reader should look to Chapter 4 of Hatcher (1982), Uzquiano (forthcoming) and Hodes (2015).

We have only scratched the surface of ways in which the type system here can be generalized. One direction generalization we haven't touched upon concerns the idea that a single expression can belong to multiple types. Systems that allow for subtyping, where one type is 'contained' in another, have applications in computer science and have been studied in some detail in that context; see for instance Mitchell (1996) Section 10.2. Subtyping could be employed by a Fregean to reintroduce a type of proposition, t, as a subtype of e. Cumulative type theory is another framework in which a single term can have multiple types (see Linnebo and Rayo (2012)). Related to the idea of an expression belonging to multiple types is the concept of a polymorphic type. In this book, we will frequently encounter typed families of expressions that behave in similar ways at each type. For each type σ, one might have a term I^σ, introduced as a constant or by definition, of type $\sigma \to \sigma$ interpreted as the identity operation on that type. Since the behaviour of identity is the same for each σ, one might suspect there is really a single identity operation, which takes a *type* as an argument, and then returns the identity operation of that type. This idea can be made precise in type theories with polymorphic and dependent types. This is a vast topic that goes well beyond the scope of this book, but the interested reader can consult Mitchell (1996) Chapter 9, and Wilhelm (2022) contains some philosophical applications.

In this chapter, we have seen that many of the familiar logical languages for conducting philosophy—propositional logic, propositional modal logic, first-order logic, and so on—are implicitly typed. It is natural, then, to wonder whether it is possible to create a logical language that is not implicitly typed. Such languages may seem very unfamiliar to the modern reader, but were actually deeply embedded in the history of logic. The first axiomatic treatment of arithmetic in Peano (1889) was untyped, in the sense that the usual rules concerning

which symbols can be combined with which are not present. Things we would ordinarily distinguish as predicates, names and sentences are all referred to uniformly by Peano as 'signs', and there are no restrictions mentioned on the proper way to combine signs.[21] When Peano states logical laws like $a \rightarrow a$ he then always includes the qualification that a refers to a proposition and not, say, a number. Thus it would appear that according to Peano (1889) the closest thing to a distinction between a name and a sentence is a semantic distinction concerning what sort of object the sign refers to, and not one of grammar. The distinction between propositions and other objects is made in the object language—Peano has what we would call 'predicate' signs, N for numbers, K for classes, P for propositions, and so on—so that Peano can write things like 'if a is a proposition, then for x to be in the class of things such that a just is for it to be the case that a'

$$a \in P \rightarrow (x \in \{x \mid a\} = a) \text{ (Proposition 59),}$$

Notice that, like Frege, Peano places the identity sign = between both names of propositions and non-propositions. Like Frege, Peano identifies $a = b$ and $a \leftrightarrow b$ when a and b are propositions (Proposition 3 of Peano (1889)) so that we can infer:

$$a \in P \rightarrow (x \in \{x \mid a\} \leftrightarrow a)$$

Here a is appearing in (what we would ordinarily think of as) name and sentence position simultaneously. Peano and his followers had a deep influence on subsequent logicians like Schönfinkel (1924) (the inventor of combinatory logic), Church (1932), Church (1933), Curry and Feys (1958), and Fitch (1936) who developed these untyped systems in parallel with the typed systems studied by Russell, Whitehead and Hilbert which later became the dominant logical framework.[22]

The systematic study of these untyped systems began in earnest by two important figures of modern type theory: Haskell Curry and Alonzo Church, in the forms of illative combinatory logic, and the illative untyped λ-calculus.[23] Both of these systems are logical languages in which the expressions do not have types at all. What are such languages like? In an illative combinatory logic, a signature simply consists of a set of typeless constants, Σ. The rule for combining expressions to make new ones is very simple: any term can be applied to any other term.

$$\frac{M \qquad N}{(MN)}$$

In order to be a logical language it must contain at least the logical connectives, such as $\rightarrow$, $\wedge$ and $\neg$. Curry also required that there be constants I, C, K and W that perform certain structural operations (C, for instance, maps an entity to its converse—see also the typed entities of the same name in Chapter 3) whereas Church achieved the same effect by his λ device.

As with Frege, these systems did not have a distinction between names and sentences and so must confront the same interpretational issues that Frege faces. So, like Frege, 'it's not the case that Gottlob' is grammatical. However, Frege still had a type system in place; a predicate of type $e \rightarrow e$ can be applied to a name, but not to itself. By contrast in Curry and Church's systems any way of combining constants by application is grammatical 'walks walks', 'Batman (snow is white it's not the case that)', and so on.[24] Sometimes the result of

putting expressions together by application results in an expression that denotes a proposition, and sometimes it simply results in a name for an individual that is not a proposition, such as a table or chair. And indeed, sometimes it can be a very non-trivial matter to determine whether an expression denotes a proposition or not—just as it was a discovery to find that 'Hesperus' refers to a planet, not a star—and thus it is often quite non-obvious whether an expression of this language is the sort of thing that is appropriate for assertion. This is in stark contrast with the typed languages described in this chapter. We do not need any extra-linguistic information to tell whether an expression is a sentence and thus appropriate for assertion, consideration, wondering about, and so on. To know whether something is a sentence one just needs to know the rules for application and the types of the constants.

A more serious issue for the illative untyped systems of Church and Curry, which they also shared with Frege's system, is that they appear to be inconsistent with classical logic. For any expression M, there is the expression $\neg(MM)$ saying that M doesn't apply to itself. Both Curry and Church allow one to abstract from this general pattern and form the property, r, of not applying to yourself.[25] Since it is perfectly grammatical to apply this operation to itself, we can coherently ask whether this property applies to itself, formalized (rr). The problem is that if it does apply to itself then it's one of the things that doesn't apply to itself, so it can't apply to itself. But if it doesn't apply to itself, then it has the property of not applying to itself so it does apply to itself.

Now this isn't quite the end of the story. The issue of inconsistency interacts with the first issue of which expressions can be asserted in subtle ways. For instance, what does classical logic even mean in this context? Normally we take any grammatical instance of a tautological schema, like $(p \wedge q) \rightarrow q$, to be a commitment of classical logic, but in the present case grammatical instances would include expressions like 'if Gottlob and Haskell then Gottlob'. Since the theories of Curry, Church and Frege were already in the business of restricting the instances of classically valid schemas, perhaps the inconsistency with the unrestricted classical schemas isn't so bad—perhaps (rr) is an expression like 'Gottlob' which is not appropriate for assertion and so should not be subject to classical laws. The matter is quite complicated, and there are many illative untyped combinatory and λ-calculi that deal with these issues in different ways.[26] Suffice it to say, however, that rejecting types requires one to accept a new and radical conception of logic and reasoning.

Endnotes

1. In natural language grammar it's sometimes posited that there are three base types, corresponding to sentences (e.g. *Fido is a dog*), noun phrases (e.g. *Fido*) and nouns (e.g. *dog*). In formal languages, and in natural language semantics, however nouns like *dog* are typically formalized or interpreted with a complex type, like that of a predicate.
2. Note that, in linguistics, the types e and t are not typically assigned to expressions, but to sets and functions, with t being assigned to the two-element set $\{true, false\}$. This is not consistent with the present terminological conventions.
3. Type systems adequate for natural language grammar are more complex: the types we have associated with English expressions above are roughly what linguists would call the expressions *semantic type*, which can be distinct from the expressions *grammatical type*. As mentioned in endnote 26, natural language grammar usually assumes the existence of three base types: nouns, noun phrases, and sentences. A noun like 'cat', for instance, has a base grammatical type, but is associated with the non-base semantic type, namely $e \rightarrow t$.
4. Frege (1893) Section 36, Schönfinkel (1924), Curry and Feys (1958).
5. Notice that $(e \rightarrow t) \rightarrow (e \rightarrow t)$ can be thought of as a unary operation taking predicates to predicates, or as a curried binary operation taking a predicate and a singular term to a sentence. Judicious choices about when to

apply the bracketing conventions can indicate how to think about it. The second way of thinking can be made salient by strictly applying the conventions: $(e \to t) \to e \to t$.

6. This convention is present as early as Hilbert and Ackermann (1928).
7. For instance, addition could be defined from these operations using λ-notation: $\lambda k(\mathrm{fix}_{e\to e}(\lambda f \lambda n.(\mathrm{if}(n)(k)(\mathrm{succ}(f(\mathrm{pred}\, n))))))$. The λ-notation will be introduced in Chapter 3.
8. You can build complex open sentences, but these are sentences, not predicates.
9. This observation famously led Frege to say some very puzzling things, such as 'the concept *horse* is not a concept'.
10. Russell is talking of facts here rather than propositions, since at this point in his writing he believed that false propositions were logical constructions, but this does not affect the wider point.
11. The English phrase seems to be complex and to involve a binary relation *means* of type $e \to e \to t$ and an operation *that* of type $t \to e$ that takes a sentence to a singular term (for instance, *that grass is green*). Since the existence of such an operation can lead to logical paradoxes when combined with other assumptions, it is thus perhaps best to take the semantic notion of type $e \to t \to t$ as primitive, and treat the English phrase 'means that' as an approximation at best.
12. Similar remarks apply here. The English expression 'means to' appears to be made up of 'means' of type $e \to e \to t$, and 'to' of type $(e \to t) \to e$, since 'to' combines with the predicate 'be tall' to make a singular term 'to be tall'. An operation like 'to', of type $(e \to t) \to e$, was famously responsible for the inconsistency in Frege's Basic Law V, so it is perhaps best to treat this way of formulating the relevant semantic primitive in English as an approximation (or to simply adopt the second English formulation).
13. See Frege (1879), Frege (1893).
14. Frege writes "We have already said that in a mere equation there is as yet no assertion; "2+3=5" only designates a truth value, without its being said which of the two it is [. . .] we therefore require another special sign to be able to assert something as true." Frege and Furth (1966) p. 37.
15. Something along these lines is considered in Bacon (2019a) as a way of making sense of ontological nihilism—the view that there are no individuals. There a type system is proposed that has only one base type t, although in that case ordinary sentences involving quantification over individuals are paraphrased in terms of quantification over $t \to t$ operators rather than propositions.
16. One can model recursive types by positing strict identities between types like $\sigma = (\sigma \to \sigma)$ as we've done above, or by simply having a pair of operations $\mathrm{fold} : \sigma \to (\sigma \to \sigma)$ and $\mathrm{unfold} : (\sigma \to \sigma) \to \sigma$ where, putting it roughly, unfold undoes fold (i.e. unfold is a right inverse of fold).
17. This sort of type system, and the corresponding notation, was used in early texts on higher-order logic, such as Hilbert and Ackermann (1928), for instance.
18. Myhill (1958), Orey (1959), and Gallin (1975) use relationally typed languages with λ-terms.
19. The system of Whitehead and Russell (1910-1913) appears to be non-cumulative, although it is not precisely stated. A more precise, but cumulative version is outlined in Church (1976).
20. One of the motivations for ramified type theory was to ban impredicative quantification. However this ban can be achieved in the simple theory of types using hierarchies of restricted quantifiers with standard types, avoiding many of the inconveniences of ramified frameworks. (NB: some authors use 'ramification' to refer to the latter sorts of approaches as well as the more prohibitive ones.)
21. See the translation in Kennedy (2012). Peano begins (Section II) by introducing conventions regarding how to bracket arbitrary strings of signs: see, for instance, his example of $((ab)(cd))$ being legitimate for any signs a, b, c and d whatsoever.
22. Little remains of Schönfinkel's work apart from two published papers, and a notebook that mentions the influence of Peano (see Wolfram (2021)). Church attributes the λ-notation (in the untyped context) to Peano and Frege, (see Church's dictionary entries in Burge and Enderton (2019)) and Curry and Feys (1958), Feys (1953) attribute the origins of combinatory logic to Peano and Burali-Forti. See Seldin and especially Cardone and Hindley (2006) for a useful overview of the history.
23. Curry and Feys (1958), Church (1932).
24. For an illuminating discussion of the semantic status of ungrammatical sentences like these see Magidor (2009).
25. We can think of the class of expressions of the form $\neg(MM)$ as being given by a single expression $\neg(xx)$ parameterized by a variable x, which yields the elements of the class when x is replaced each term M. Both Curry and Church have resources that allow one to introduce a property for every expression parameterized by a free variable like this (much like the $\mapsto$ notation lets you introduce a function for any mathematical expression parameterized by a variable). Curry achieves this in a fairly complicated manner, by way of the I, C, K and W constants, whereas Church uses the λ-notation which behaves a little bit like the $\mapsto$ notation for functions. In Church's formalism, the property of not applying to yourself is $\lambda x.\neg(xx)$. The untyped λ-calculus may also be

interpreted in Frege's Grundgesetze: given a term M with a free variable x a Churchean λ-term $\lambda x.M$ can be simulated as a Fregean value-range-term, $\grave{\varepsilon}.M$ which binds occurrences of the singular variable ε, and Church's notion of application, which he writes MN, can be simulated in Frege's Grundgesetze using his primitive binary application function symbol, $\frown$, for applying value-ranges to arguments which he would write $N \frown M$; each logical constant gets an interpretation this way too, for instance negation would be $\grave{\varepsilon}(\neg\, \varepsilon)$, and the paradoxical Russell term would be $\grave{\varepsilon}(\neg\, \varepsilon \frown \varepsilon)$. This gives us an alternative way of generating inconsistency in Frege's system of the Grundgesetze by way of the inconsistency of the untyped λ-calculus.

26. For some relevant literature see Curry and Feys (1958) , Fitch (1974), Scott (1975), Bunder (1979), Plotkin (1990), Barendregt et al. (1993), Caie (2020), Czajka (2013). The illative untyped systems can also be related to Frege's project in Frege (1893) when you take into account that Frege has operations mapping the type of properties—$e \to e$ for him—to individuals, which lets you simulate the untyped λ-calculus in a broadly Fregean system. See Aczel (1980).

2

An informal introduction to abstraction

In a typed applicative language, the only way to combine expressions is by application. This makes these languages pretty limited. We might therefore wish to explore languages with other devices for combining expressions.

To illustrate, consider the signature of propositional modal logic, of Example 1.2. That language contains two operator constants, $\Box : t \to t$ and $\neg : t \to t$. The impossibility operator, 'it's necessarily false that', seems to be something we ought to be able to define from the operator expressions 'it's necessary that' and 'it's not the case that' alone. Indeed, given any term $A : t$, we may create another term $\Box(\neg A) : t$ in the language, informally corresponding to the claim that it's necessarily false that A. But note that this observation differs importantly from the idea that the language contains a defined *operator expression*, corresponding to impossibility, for it merely states a closure condition on expressions of type t: it says that if you can build one term of type t, A, then you can build another, $\Box(\neg A)$. Metaphysically, we might think of this committing us to further propositions, without guaranteeing the existence of any further *operators*, beyond necessity and negation. Indeed, the only operator expressions you can construct in a typed applicative language from sentences, $\Box$ and $\neg$ are $\Box$ and $\neg$. Thus we do not have any defined operators expressions, and in particular, we don't have impossibility.

What we would like is some complex operator expression, $(\Box\neg) : t \to t$, such that $(\Box\neg)A$ means the same as $\Box(\neg A)$. Of course, $(\Box\neg)$ is not well-formed, as (MN) is our notation for application: $\Box$, having type $t \to t$, must take something of type t as argument, not $t \to t$. We need a different way of combining two things: something that takes an expression of type $M : t \to t$, and another expression of type $N : t \to t$, and produces a third term of type $t \to t$, which informally corresponds to the operation of applying N and then M in that order.

Here is another example. Consider the signature that contains a binary relation $L : e \to e \to t$, informally meaning 'loves'. It's a binary relation, but there's also a related unary predicate, informally corresponding to loving oneself, that one might expect to also be definable. Again, given any term $a : e$ (for *Alice*, say) we can formulate the sentence $Laa : t$, stating that *Alice loves herself*. But this again corresponds to a hypothesis about the existence of propositions: if you have an individual, like *Alice*, and a binary relation, *loves*, you can form the proposition that *Alice loves herself*. It does not ensure that there is the unary property *loves herself*. A metaphysician who wants to posit a further unary property of *loving oneself* needs a convenient notation that takes an expression $R : e \to e \to t$ and produces a new unary predicate $R^w : e \to t$ of bearing R to oneself—the *contraction*, or *reflexization* of R. A similar line of thought motivates a notation for converse relations: with L alone one can express the claim that *Alice loves John* and that *Alice is loved by John*, but we need something

DOI: 10.4324/9781003039181-4

more to have a separate binary predicate, L^c, corresponding to the relation of *being loved by* (the converse of *loving*).

Comprehension Check 2.1. *Consider the typed applicative language generated by the signature* $L : e \to e \to t$ *and* $a : e$*, with the above interpretations. Which of the following can be defined? (i) a predicate of type* $e \to t$ *corresponding to* being loved by Alice*, (ii) a predicate of type* $e \to t$ *corresponding to* loving Alice.

2.1 Abstraction

All of these examples of complex expressions, and more, can be formed by a process of *abstraction* from expressions we already take to be meaningful in an applicative language. The idea is this. Suppose we have a meaningful sentence, such as 'Socrates is old and Socrates is wise'. It is then possible to consider the result of putting a gap where a particular expression used to be in that sentence. Replacing the name 'Socrates' with a gap yields:

... is old and ... is wise.

Inserting a name into this gap yields a sentence, so that this gappy sentence provides us with a uniform method of associating an arbitrary type e expression with a corresponding type t expression:

Socrates ⇒ Socrates is old and Socrates is wise.
Aristotle ⇒ Aristotle is old and Aristotle is wise.
Plato ⇒ Plato is old and Plato is wise.
etc...

According to the hypothesis that we can introduce meaningful new predicates by abstraction, to any uniformly parameterized class of meaningful sentences like this, there is a corresponding meaningful predicate. In the present case, we can introduce a new $e \to t$ predicate, meaning *is old and wise*, that combines with a name, a, to make a sentence that means the same as 'a is old and a is wise'. Of course, we can leave gaps in the positions of other types of expressions. For instance, by punching out the predicate from 'Socrates talks' we get

Socrates ...

If abstraction were legitimate, there would correspond to this gappy sentence a higher-order predicate, of type $(e \to t) \to t$, that combines with an ordinary $(e \to t)$ predicate, F, to make a sentence that means Socrates Fs. Similarly, we can leave gaps in expressions that are not sentences. By leaving a gap in the complex $e \to t$ predicate, 'John loves', we can make

... loves

the gappy expression is something that associates each expression of type e to a $e \to t$ predicate, e.g. takes 'Socrates' to 'Socrates loves', and so the abstracted expression would have type $e \to e \to t$.

The hypothesis that this method corresponds to a way of introducing meaningful predicates can now be informally stated:

The Abstraction Hypothesis An interpreted applicative language can be meaningfully closed under definitions by abstraction.

This is somewhat imprecise, and we will have to wait until later to make it precise. The reader should remember, too, that the talk of the meanings of expressions of a typed language, and thus also meaningfulness, is delicate (recall Section 1.3).

We can see how the abstraction hypothesis accommodates the examples we opened with. In any applicative language with unary operators $\Box$ and $\neg$, we can make a sentence with a gap '$\Box\neg \ldots$'—'it's necessarily false that . . . '—associating sentences with sentences, e.g. '1=0' with 'it's necessarily false that 1=0'. The result of abstraction then yields the required $t \to t$ expression. By leaving a gap in two places we can make '. . . loves . . . ', and the corresponding definition by abstraction yields the desired reflexization of 'loves', meaning *loves oneself*.

The Abstraction Hypothesis is a substantive one. Later in the book we will encounter philosophical views that reject the existence of things like converses and reflexizations, and so we will consider weakening the hypothesis in various ways. However, it is only possible to understand the philosophical reasons for rejecting the hypothesis once it has been sufficiently developed. So it makes sense to start with the full strength hypothesis before we consider weakenings. (Weakenings will be the subject of Chapters 9 and 10). On the other hand, there is also much to be said in favour of the hypothesis: it is a theoretically powerful posit.

Remark 2.1. For those readers familiar with mathematical functions, the two ways of combining expressions discussed are reminiscent of operations on functions. Given a pair of functions $f : A \to A$, and $g : A \to A$, we can produce their composition $f \circ g : A \to A$: the function which maps an element $a \in A$ to $f(g(a))$, the result of applying one of these functions after the other. It is common to represent 'the function that maps a to $\ldots a \ldots$' with the notation $a \mapsto \ldots a \ldots$, thus allowing us to define the composition $f \circ g$ more compactly as $a \mapsto f(g(a))$. The second operation of forming a unary reflexization of a binary relation is also reminiscent of an operation that takes a binary function $f : A \times A \to B$ and produces the unary function that maps an element $a \in A$ to the result of feeding a to f in both of its arguments. For example, reflexizing the binary operation of addition yields the operation of doubling. We can similarly notate this $a \mapsto f(a, a)$. The $\mapsto$ notation for talking about functions also closely resembles the notation we will use to formalize abstraction. However, despite these similarities, there are some important differences. Mathematicians often identify functions with sets. Grammatically, symbols referring to functions are of type e (the set-theorists words for functions should therefore be sharply distinguished from *function symbols*, such as those in signature of Example 1.4, which have complex types like $e \to e$). The analogous device for creating complex operators and predicates, by contrast, serves a very different purpose, and will belong to a different type. Moreover, the analogy with functions is precarious: many principles that govern functions correspond to highly contentious principles of higher-order logic. For instance, no two functions can agree on all their arguments and yet be different overall, and there is a function for any functional relation between two domains. Both of these principles correspond to substantive theses of higher-order logic (see Section 6.5), so we should be cautious about basing our intuitions about properties and relations too closely on functions.

2.2 Introducing λ

Let's first find a notation for representing definitions by abstraction. We introduce this notation informally for now. It will be defined properly in the next chapter. First, instead of expressions with a gap where another expression of type σ should be, we'll make use of *variables*. A variable of type σ behaves syntactically exactly like a constant of type σ behaves: we could therefore add variables to an applicative language by simply expanding the signature. However, it is often useful to keep variables and constants separate. We'll generally use letters towards the end of the alphabet, like x, y, z, X, Y, Z for variables. Expressions containing variables are often called *open* expressions, whereas those with no variables are called *closed*. Instead of the gappy sentence '... is old and ... is wise' we'll thus use the open sentence:

x is old and x is wise

where x is a variable of type e. This is a sentence that is variable in the name position because it defines a class of sentences parameterized by the names you could substitute for x: 'Socrates is old and Socrates is wise', 'Aristotle is old and Aristotle is wise', etc.

To notate the predicate we obtain by abstracting from this parameterized class of sentences, we will prefix this variable sentence with a λx.

$\lambda x.\ x$ is old and x is wise.

While the open expression 'x is old and x is wise' will be considered as having the same type as a sentence, namely type t, the λ-expression above is a predicate.

The ability to abstract gives us a whole host of new ways of combining expressions to make new expressions. Let's have a look at some of them. Recall, first, that in an applicative language, we could already combine things by application

$$\frac{M : \sigma \to \tau \qquad N : \sigma}{(MN) : \tau}$$

Again, we read this as saying that if you have the things above the line, then you can make the things below the line.

Composition: given $F : \sigma \to \tau$ and $G : \tau \to \rho$, we may *compose* them to produce a term $G \circ F : \sigma \to \rho$. Applied to a term $a : \sigma$ it is synonymous with $G(Fa)$

$$\frac{F : \sigma \to \tau \qquad G : \tau \to \rho}{G \circ F : \sigma \to \rho}$$

Note the reversal in $G \circ F$, relative to the order in which they are applied: this is because we are following standard conventions of placing predicates to the left of the things they are applied to, whereas we depict the source and target of a functional type on the left and right respectively. Some computer scientists introduce an operation ;, and write $F; G$ for $G \circ F$, reflecting direction of the arrows in the types of F and G.[1] The composition of F and G is defined by abstraction:

$G \circ F := \lambda x.(G(Fx))$

where x is a variable of type σ.[2] Here we use the symbol $:=$ to indicate that we are giving a definition.

The reflexizations of binary (curried) operations are now expressible too.

Reflexization: given $R : \sigma \to \sigma \to \tau$ we may form the reflexized term $R^w : \sigma \to \tau$. Applied to a term $a : \sigma$ it is synonymous with Raa

$$\frac{R : \sigma \to \sigma \to \tau}{R^w : \sigma \to \tau}$$

It is defined by abstraction as follows

$$R^w := \lambda x.Rxx$$

Exercise 2.1. *Let* $\mathrm{add} : e \to e \to e$ *be the curried function that represents addition on the numbers: i.e.* $\mathrm{add}\, nm = n + m$ *when n and m are numbers (and is 0 whenever one of the arguments is not a number).*

a. What is the type of $\lambda x.\mathrm{add}\, xx$ *(i.e the type of* add^w*)?*

b. Describe $\lambda x.\mathrm{add}\, xx$ *in more familiar terms.*

We'll end with two limiting cases of abstraction. We can take an expression, say 'Socrates', and make a gap where the entire expression is, '. . . '. The resulting operation is an $e \to e$ function symbol which, when applied to a name, is synonymous with that name. More generally:

Identity: for any type σ, there is an identity operation of type $I^\sigma : \sigma \to \sigma$. For any term $a : \sigma$, $I^\sigma a$ is synonymous with a.

$$\frac{}{I^\sigma : \sigma \to \sigma}$$

here we have nothing above the line. This simply means you do not need any expressions to construct the identity operation. The operation is defined as follows:

$$I^\sigma := \lambda x.x$$

where x is a variable of type σ.

Another limiting case of abstraction occurs when we abstract a variable that doesn't occur in an expression. For instance, from the sentence 'snow is white', we can form the vacuous predicate 'λx.snow is white' (denoting the property of *being such that snow is white*) which applied to an arbitrary name is synonymous with 'snow is white'.

Vacuous abstraction: given $a : \tau$ we may form the vacuous operation $a^k : \sigma \to \tau$. Applied to a term $b : \sigma$ it is synonymous with a

$$\frac{a : \tau}{a^k : \sigma \to \tau}$$

Exercise 2.2. *Define* a^k *using* λ.

Remark 2.2. Above I have described the behaviour of certain expressions, such as I^e (that is $\lambda x.x$), by saying what they are synonymous with when applied to certain other expressions. We might also want to say what the thing that I^e denotes, the operation of *identity*, does to each individual (it maps each individual to itself). To state these ideas carefully and in full generality one would need to use higher-order generalizations.

2.3 Multiple abstraction and currying

Note that by using variables, instead of mere gaps, we increase the number of things we can do with abstraction. For a start, we can make sentences with multiple sorts of gaps, like 'x is old and y is wise'. We could represent this by representing gaps with different sorts of ellipses:

... is old and ___ is wise

but we would quickly run out of different ways to notate these differences. Once we can do this, we can pinpoint particular gaps that we want to abstract from, yielding an expression with one less kind of gap. For instance, abstracting in the position of x, yielding an $e \to t$ predicate with one gap left, '$\lambda x.$ x is old and y is wise': *being such that you are old and y is wise*. This is thus a class of unary predicates parameterized the value of y:

Socrates $\Rightarrow$ $\lambda x.$ x is old and Socrates is wise.

Plato $\Rightarrow$ $\lambda x.$ x is old and Plato is wise

Aristotle $\Rightarrow$ $\lambda x.$ x is old and Aristotle is wise

and so on...

Thus we can then abstract again, now into the position of y, making an $e \to e \to t$ expression: '$\lambda y.(\lambda x.$ x is old and y is wise)'.

Multiply abstracted expressions are pervasive in this book. The first use we will have of them is in the representation of 'curried' binary and higher arity operations. Consider a term like $5 + 3$, that you might find in the first-order language of arithmetic. In such a context you will find $+$ flanked by two numerical terms making an e type expression: let's pretend we don't know anything about the syntactic behaviour of $+$ in isolation (perhaps it is syncategorematic). By abstracting into the position of the 3, we can form the $e \to e$ function symbol of adding 5: $\lambda y.5 + y$. More generally, for any given number n, there is a unary function of *adding n*, which we may write $\mathrm{add}\, n$. It is defined as follows:

$$\mathrm{add}\, n := \lambda y.n + y.$$

Where y is a variable of type e. We can then think of addition *itself* as the higher-order function that takes a number, n, as input and outputs the function of adding n. This is what we have called the 'curried' perspective on addition. Its type would thus be:

$$\mathrm{add} : e \to (e \to e).$$

We can define this higher-order operation as follows. Since *adding n* is a class of unary functions uniformly parameterized by the value of n, we can use abstraction to introduce an operation mapping n to the function of adding n.

$$\mathrm{add} := \lambda x.\, \mathrm{add}\, x$$

of course, $\mathrm{add}\, x$ is itself a λ-term, $\lambda y.x + y$, so in full:

$$\mathrm{add} := \lambda x.(\lambda y.x + y)$$

Note that the result of applying *adding n* to *m* is of course just $n + m$. Thus there is a straightforward connection between our original variable expression of type *e*, $x + y$, and the explicit addition function add:

$$\mathrm{add}\, xy \equiv x + y$$

One should read the $\equiv$ sign here as representing synonymy. We might think of the above equation as a specification of the desired applicative behaviour of the function symbol add, and our definition in terms of λ as a solution to this specification. Indeed, abstraction ensures we are always able to solve equational specifications like this. Suppose you wanted a unary $e \to e$ function, *f*, that adds 4 to its input. I could specify this behaviour equationally as follows:

$$fx \equiv x + 4$$

This equation thus tells us we're looking for a function symbol that, when applied to an arbitrary name, *a*, yields something synonymous with $a + 4$. Abstraction lets us define this function symbol explicitly: $f := \lambda x.x + 4$. Or suppose I want a unary predicate that satisfies

$$Fx \equiv Ox \wedge Wx$$

The equation tells us we're looking for a predicate that, when applied to an arbitrary name is synonymous with $Oa \wedge Wa$ (*a* is old and *a* is wise). We can turn this into a explicit definition $F := \lambda x.(Ox \wedge Wx)$. Addition is binary, so the specification of its applicative behaviour is spelled out by an equation in two parameters:

$$\mathrm{add}\, xy \equiv x + y$$

More generally, if we wanted an operation $F : \sigma_1 \to \cdots \to \sigma_n \to \tau$, where τ is a base type, we may attempt to specify *F* with an equation:

$$Fx_1 \ldots x_n = M$$

where *M* is some expression involving the variables $x_1, \ldots, x_n$. In general, the idea is to keep applying *F* to arguments $x_1, x_2, \ldots$ of appropriate types, until you no longer have an expression which can be applied to anything (i.e. you have a name or a sentence), and then explicitly say what the result is in terms of these variables.

The skill of being able to explicitly define an operation from such a specification is an important one that we will need again and again, so it will be worth pausing for a minute to attend to how these definitions really work. Given a specification of the behaviour of an operation, its definition in terms of λ can actually be calculated quite systematically. In general, if you want something defined by an equation $Fx = whatever$, then we know *F* can be defined as $\lambda x.whatever$. We can thus systematically obtain explicit definitions of operations from our equational specifications by moving the *x* on the left-hand-side of the equation to the right-hand-side, as follows:

$$\frac{fx \equiv x + 4}{f \equiv \lambda x.x + 4}$$

For binary curried functions, it's the same, but now we can apply the process twice:

$$\frac{\dfrac{\mathrm{add}\,xy \equiv x + y}{\mathrm{add}\,x \equiv (\lambda y.x + y)}}{\mathrm{add} \equiv (\lambda x.(\lambda y.x + y))}$$

Thus we have in general

$$\frac{Fx_1 \ldots x_n \equiv M}{F \equiv \lambda x_1.(\ldots(\lambda x_n.M)}$$

Multiple abstraction also lets us create converse relations. Take the sentence 'John loves Mary'. By abstracting out 'John', we get a unary predicate denoting the property of *loving Mary*. If we abstract out 'Mary' from this we simply get something denoting the *loving* relation. Suppose, now, we abstract in a different order: abstracting 'Mary' first we get the unary predicate denoting *being loved by John*. If we then abstract 'John', we get something denoting the converse of *loves*, the relation of *being loved by*. We can also arrive at this same result using the equational method above. We can specify the behaviour we want from the converse of a relation L: $L^c xy \equiv Lyx$. Thus

$$\frac{\dfrac{L^c xy \equiv Lyx}{L^c x \equiv (\lambda y.Lyx)}}{L^c \equiv (\lambda x.(\lambda y.Lyx))}$$

In summary, this gives us another way in which abstraction increases the expressivity of an applicative language.

Converses: given $R : \sigma \to \tau \to \rho$ we may form the converse operation $R^c : \tau \to \sigma \to \rho$. Applied to terms $a : \tau, b : \sigma$ it is synonymous with Rba

$$\frac{R : \sigma \to \tau \to \rho}{R^c : \tau \to \sigma \to \rho}$$

$$R^c := (\lambda x.(\lambda y.Ryx))$$

where y is a variable of type σ and x of type τ

Exercise 2.3. *Suppose that you have a binary predicate T meaning* taller than *and the truth-functional connectives. Suppose I want a ternary relation R such that Rxyz means 'y is between x and z in height'. Using the equational method find an explicit definition of R in terms of T and abstraction.*

2.4 Getting more abstract

This is just the beginning. Once we have the ability to abstract multiple times, we can keep abstracting away from an expression until we have a completely abstracted term—a *pure* λ-term. For instance, take the sentence:

It's necessary that it's necessary that 1=1.

Replacing the necessity operators with an operator variable and abstracting we get $\lambda X.X(X(1 = 1))$. This takes an operator, X, and maps it to the proposition you get by applying it to 1=1 twice. If we abstract out the sentence 1=1, we get a pure expression $\lambda p.(\lambda X.(X(Xp)))$ that takes a proposition and an operator to the result of applying that operation twice

Comprehension Check 2.2. *What is the type of* $\lambda p.(\lambda X.(X(Xp)))$*?*

Exercise 2.4. *Consider the analogous operation, d, that takes an individual* $x : e$ *and a function* $Y : e \to e$ *and applies Y to x twice:* $d := \lambda x.(\lambda Y.(Y(Yx))$

a. *Let* $\mathrm{suc} : e \to e$ *be the function that maps each positive whole number to its successor and maps non-numbers to themselves. What is* $d\ \mathrm{suc}\ 4$*?*

b. *Describe the* $e \to e$ *function* $d\ \mathrm{suc}$ *in more familiar terms.*

Indeed, each of the new ways of combining terms described above—composition, reflexization, and so on—corresponds to a higher type operation that can be defined explicitly by abstraction. Recall that composition lets us combine two operations, $X : \tau \to \rho$ and $Y : \sigma \to \tau$ to make $X \circ Y : \sigma \to \rho$. Thus we can create a higher-order operation that takes two operations as arguments, X and Y, and outputs $X \circ Y$. Its type would thus be $(\tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)$. The standard name for this operation is B. We can specify its behaviour equationally—$BXY \equiv X \circ Y$—so $By := \lambda X.(\lambda Y.X \circ Y)$. Of course, $X \circ Y$ is short for a λ term, so its full definition is

$$B := \lambda X.(\lambda Y.(\lambda x.X(Yx)))$$

In the following exercises, you will do the same for the other modes of combination we have encountered thus far. These further applications of abstraction ought to give the reader a taste of the sorts of things we can do with abstraction.

Exercise 2.5. *Define an operation* $W : (\sigma \to \sigma \to \tau) \to (\sigma \to \tau)$ *that, when takes a binary operation to its reflexization. Applied* $R : \sigma \to \sigma \to \tau$ *yields* $R^w : \sigma \to \tau$ *(i.e.* $\lambda x.Rxx$*.)*

Exercise 2.6. *Define an operation* $C : (\sigma \to \tau \to \rho) \to (\tau \to \sigma \to \rho)$ *that, when takes a binary operation to its converse. Applied to* $R : \sigma \to \tau \to \rho$ *yields* $R^w : \tau \to \sigma \to \rho$ *(i.e.* $\lambda xy.Ryx$*.)*

Exercise 2.7. *Define an operation* $K : \tau \to \sigma \to \tau$ *that, when applied to* $a : \tau$ *yields* $a^k : \sigma \to \tau$ *(i.e.* $\lambda x.a$*.)*

Endnotes

1. This mismatch could be avoided by employing less familiar conventions. Church (1940) reverses the order of source and target: he would write $(\tau\sigma)$ for the type we depict as $(\sigma \to \tau)$. The mathematicians Bourbaki instead placed functions to the right of their arguments. Note the F; G notation can be confusing when combined with the convention of writing functions and predicates to the left of their arguments, since $(F; G)a$ means the same as $G(Fa)$.
2. Here we assume that F and G do not contain any variables. Fussing about variables is something we defer to the next chapter.

3

λ-languages

Having introduced the idea of abstraction, we can now proceed to introduce abstraction into typed applicative languages. The result is a *simply typed λ-language*. The full λ-language over a given signature and the central notion of a combinator are introduced in Sections 3.1 and 3.2. Sections 3.3 and 3.4 introduce some key principles of synonymy for λ-languages, α, β and η, and it is shown that they pin down the meaning of the λ-terms up to synonymy. Sections 3.5 and 3.6 introduce an alternative variable-free formalism, combinatory languages, and some meaning-preserving translations between the λ-language and combinatory language are described.

3.1 The full λ-language

Implicit our notation for abstraction is the use of dummy variables to indicate what happens to an arbitrary argument. Thus in order to formally specify a language with λ we need to officially introduce variables into our language. Variables, like constants, are associated with unique types. We write *Var* for the set of all variables and Var^σ for the set of variables of type σ. It is always assumed that there are infinitely many variables in Var^σ for any type σ.

Definition 3.1 (Typed λ-Languages)**.** *Given a signature Σ, the typed λ-language, $\mathcal{L}(\Sigma)$, based on Σ is a typed collection of sets, $\mathcal{L}^\sigma(\Sigma)$, called the* terms *of type σ. Writing $M : \sigma$ as short for $M \in \mathcal{L}^\sigma(\Sigma)$, we may define the terms of type σ, for each σ, simultaneously as follows:*

- $c : \sigma$ *whenever* $c \in \Sigma^\sigma$.
- $x : \sigma$ *whenever* $x \in Var^\sigma$.
- *If* $M : \sigma \to \tau$ *and* $N : \sigma$ *then* $(MN) : \tau$.
- *If* $M : \tau$ *and* $x \in Var^\sigma$, $(\lambda x.M) : \sigma \to \tau$.

Only terms that can be built by the above rules are terms of the language $\mathcal{L}(\Sigma)$.

From now on we will primarily be concerned with λ-languages, so that talk about languages without qualification can be assumed to be about λ-languages. To ease readability, I'll lay out some rules of thumb about how we will use letters and capitalizations to denote terms.

Convention 3.1. *Terms will often be denoted using lower and upper case Latin letters.*

Letters towards the end of the alphabet, w, x, y, z will be used for variables of types other than t. We will use p, q, r, s for variables of type t. In general, we will try to use lower case letters

DOI: 10.4324/9781003039181-5

for terms that are going to appear in argument position within a larger term or they have type e or t, and upper case otherwise.

Sometimes we will want to talk about an arbitrary expression, or an arbitrary expression of a given type, in which case we use a 'metalinguistic variable' that ranges over expressions. These will typically involve the letters $a, b, c, d, e, F, G, H, L, M, N, P, Q, R, S$.

Thus for instance, we might write Xyz for a relational variable $X : e \to e \to t, y : e, z : e$. In general, we will use judgement about what is most readable, and what accords with existing conventions. We will not introduce Latin letters for established symbols, such as $\neg$, $\Box$ and $\exists$. We will introduce more conventions for omitting unnecessary parenthesis shortly, however we will immediately adopt the policy of omitting the outermost parentheses of a term since doing so never introduces ambiguity.

Exercise 3.1. *Show that the following are terms of* $\mathcal{L}(\Sigma)$*, where* Σ *contains one constant* $R : e \to e \to t$*, and state their types.*

a. $(\lambda x.Yx)$*, where* $Y \in Var^{e \to t}$ *and* $x \in Var^{e}$.

b. $(\lambda y.(\lambda x.((Ry)x)))$ *where* $x, y \in Var^{e}$.

c. $(\lambda x.x)$ *where* $x \in Var^{e}$.

Exercise 3.2. *Choose concrete types* σ *and* τ *(written explicitly in terms of e and t) such that if X and y have types* σ *and* τ*, the string* $(\lambda X.(\lambda y.((Xy)y)))$ *is a term.*

After you've done this, think of the possible types you must assign X and y so that $(\lambda y.((Xy)y))$ *is a well-formed term and has type* $e \to t$*. Give an example.*

Exercise 3.3. *Argue that the variables appearing in* $Y((\lambda Y.(\lambda x.Yx))Z)$ *cannot be assigned types that make it into a well-typed term.* [i]

Apart from the rule for variables, which is exactly parallel to the rule for constants, we have one new rule: the rule of *λ-abstraction*. The rule formalizes our informal remarks of the previous section. From a term A, of type t, that contains a variable, x, of type e, we can abstract to make $\lambda x.A$, a predicate, of type $e \to t$, which we may pronounce 'is an x such that A'. If A abbreviates the variable sentence 'x is wise and x is old', we might pronounce the formula and corresponding to the λ abstract:

$\lambda x.A$: is an x such that x is wise and x is old.

It is often possible to find less awkward English expressions that mean the same as $\lambda x.A$, not involving variables: in this case, the predicate 'is wise and old'. When A is not a sentence, or x is not a variable of type e, English paraphrases of $\lambda x.A$ are harder to come by but their meanings may be apprehended by analogy with the above case.

We have been employing the distinction between open and closed sentences. Lets now try and make that notion precise. Like the quantifiers of predicate logic, λ is a variable binder. We should have analogous notions of scope and of a variable being free.

A term of the form $\lambda x.M$ is called a λ-abstract or a λ-term. M is called the scope of the abstract.

For instance, $(\lambda y.((Ry)x)$ is the scope of $\lambda x.(\lambda y.((Ry)x))$.

Definition 3.2 (Free variable)**.** *We define the set of free variables in a term,* $FV(M)$*, inductively as follows:*

- $FV(c) = \emptyset$ *for* $c \in \Sigma$*;*
- $FV(x) = \{x\}$ *for* $x \in Var$*;*
- $FV(MN) = FV(M) \cup FV(N)$*;*
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$.

A term, M, is closed *if* $FV(M) = \emptyset$ *and* open *otherwise.*

Comprehension Check 3.1. *What are the free variables in (i)* $X(\lambda z.z)$*, (ii)* Xyz*, (iii)* $Y((\lambda Y.(\lambda x.Yx))Zw)$*?*

It's worth pausing for a minute to emphasize the difference between an open sentence and a predicate. In textbooks on predicate logic its common to treat the two interchangeably. If, in predicate logic, you had predicate constants W and O, for *is wise* and *is old* respectively, then it's common to talk as though one could define in the language the complex predicate *is wise and old*. Certainly one can construct open formulae Wx and Ox, and thus the corresponding conjunctive formula $Wx \wedge Ox$, and for many purposes, this is good enough. But an open formula is not the same as a predicate. After all, we distinguish the predicate W from the open sentence Wx: one can grammatically combine the former, but not the latter, with a singular term to get a formula, and one can grammatically combine the latter, but not the former, with an operator to make another formula. Indeed, one gets the open sentence Wx from the predicate W by doing just that: combining it with the singular variable x. A predicate *being wise and old*, if it existed in the language, ought to be the sort of thing one could combine with a singular term: $Wx \wedge Ox$ is not that sort of thing—it is already a complete (albeit open) sentence. The missing link is the device of λ-abstraction which turns an open sentence into a predicate. These remarks apply more generally to an open term M of type τ in a free variable x and its abstract $\lambda x.M : \sigma \to \tau$.

The full λ-language in the signature of modal logic (Example 1.2) contains the λ-term, $\lambda p.\Box(\neg p)$—the impossibility operator—discussed in the last chapter. Similarly, any full λ-language over a signature containing a relation $L : e \to e \to t$ contains $\lambda x.Lxx$. Continuing that line of discussion, let's look at some other things we can now represent.

There is a conspicuous mismatch between logical treatments of negation, conjunction, necessity, and so forth, and their natural language counterparts. For instance, in English negation and necessity are treated as adverbs. They combine with a predicate to make another predicate: for instance, the word *not* combines with a predicate *is tall* to make *is not tall*. The word *necessarily* is similar. The types of these expressions presumably should be represented by $(e \to t) \to (e \to t)$. Yet in logical treatments of these expressions, negation and necessity are treated as operators, of type $t \to t$, and regimented in English with the slightly awkward locutions: *it's not the case that* and *it's necessarily the case that*. λ allows us to define predicate modifier negation from sentence operator negation, and similarly for predicate necessity and operator necessity.

To figure out how to define the operation of predicate negation, let's first consider what it ought to do to a fixed predicate, $T : e \to t$, meaning tall. (The case of necessity proceeds in parallel.) The result should be another predicate, meaning *is not tall*. To figure out how to define *is not tall*, we finally consider what it should do to a given term $a : e$ meaning Alice. The claim that Alice is not tall can be stated using operator negation:

$\neg(Ta)$

The predicate of being *not tall*, then, must take an arbitrary x to $\neg(Tx)$, and so may be defined via λ abstraction:

$$\lambda x.\neg(Tx)$$

Generally, for an arbitrary predicate, Y, the predicate negation of Y is $\lambda x.\neg(Yx)$. Finally, predicate negation must take an arbitrary predicate, Y, to the predicate negation of Y, and so may be defined by λ abstraction:

$$\lambda Y.(\lambda x.(\neg(Yx))$$

We could arrive at this same result by giving a specification of the behaviour we wanted from predicate negation: we want an operation $N : (e \to t) \to (e \to t)$ such that $NYx \equiv \neg(Yx)$ thus, moving the x and then the Y to the other side, we obtain $N := \lambda Y.(\lambda x.(\neg(Yx))$.

Comprehension Check 3.2. *Given that* $\neg : t \to t$, $Y : e \to t$ *and* $x : e$, *show that the type of predicate negation is* $(e \to t) \to (e \to t)$ *by Definition 3.1.*

A similar phenomenon is present in English with the word *and*. The word *and* can actually be placed between expressions of many different types: *snow is white and grass is green*, *old and wise*, *quickly and purposefully*, *some professor and every student*, *Russell and Whitehead*. Let's focus on predicate conjunction. Given two predicates, W and O, we have above shown how to represent the predicate conjunction of W and O, namely $\lambda x.(Wx \wedge Ox)$. Predicate conjunction is the operation that takes two predicates, X and Y of type $e \to t$, and returns their predicate conjunction: $\lambda z.(Xz \wedge Yz)$, where $z : e$. It is a curried binary operation, and thus may be defined:

$$\lambda X.(\lambda Y.(\lambda z.(Xz \wedge Yz))))$$

The type of this operation is thus: $(e \to t) \to (e \to t) \to (e \to t)$.

We also have a notion of conjunction that takes two quantifier phrases and gives us another one. It takes, for instance, *some professor* and *every student* and gives back *some professor and every student*. If P and Q are particular quantifier phrases, of type $(e \to t) \to t$, then the quantifier conjunction of them should be something that combines with a predicate, F, to give the conjunction of the two quantificational claims: $QF \wedge PF$. Thus the quantifier conjunction of P and Q is:

$$\lambda X.(QX \wedge PX) : (e \to t) \to t$$

where X is a variable of type $e \to t$. Quantifier conjunction itself takes two quantifiers, Y and Z, of type $(e \to t) \to t$ to their quantifier conjunction $\lambda X.(YX \wedge ZX)$:

$$\lambda Y.(\lambda Z.(\lambda X.(YX \wedge ZX)))$$

where $Y, Z : (e \to t) \to t$ and $X : e \to t$. The type of quantifier conjunction is thus:

$$((e \to t) \to t) \to ((e \to t) \to t) \to ((e \to t) \to t)$$

Take a moment to satisfy yourself that the type above is correct.

Exercise 3.4. *There is also a notion of conjunction for predicate modifiers, as in:* quickly and purposefully. *(i) What should the type of this operation be? (ii) Define it explicitly using* λ *and the conjunction connective* $\wedge : t \to t \to t$.

Let's now turn to another important use of λ-abstraction in representing logical languages. Consider again the signature of predicate logic (Example 1.3). Recall that we treated the universal quantifier $\forall$ as an expression of type $(e \to t) \to t$: something that combines with a predicate to form a sentence. Notice how different this is from the treatment of quantification in predicate logic. There are two crucial differences. The first is that in predicate logic quantifiers behave syntactically like operators in that the rules for constructing sentences tells you that they combine with sentences to form sentences: if A is a sentence then $\forall x.A$ is also a sentence. The second is that quantifiers are *variable binders*: if x is free in A, then it is not free in $\forall xA$. Indeed, in predicate logic $\forall x$ is typically treated as a unit—something that only ever appears next to a variable, much like λ in λ-languages. By contrast, our treatment assigns the lonely term $\forall$ a type, and it can appear on its own as an argument: e.g. $\lambda X.X\forall$, where $X : ((e \to t) \to t) \to t$.

The typed applicative language based on the signature of Example 1.3—predicate logic—is quite restricted in what can be expressed. We noted above that without λ-abstraction, predicate logic has no complex predicates. Thus, without λ-abstraction we can only form simple quantificational claims like *something is wise*, $\exists W$, but not *something is wise and old*. To form the latter sentence we need a predicate, of type $e \to t$, to which we can apply the existential quantifier, of type $(e \to t) \to t$: $\lambda x.(Wx \wedge Ox)$. Thus we may represent *something is wise and old* as follows:

$$\exists \lambda x.(Wx \wedge Ox)$$

More generally, given a (possibly open) sentence, $A : t$, we can form a predicate $\lambda x.A : e \to t$, to which we can apply the universal first-order quantifier, $\forall \lambda x.A$, representing the claim that every x is such that A. This suggests that the universal quantifier of predicate logic can be decomposed into two steps: turning A into the predicate *is an x such that* $A(x)$, and then applying $\forall$, resulting in the sentence saying that this predicate is universally satisfied.

Convention 3.2. *In general we omit* λ*s appearing immediately after quantifiers, writing* $\forall xA$ *instead of* $\forall \lambda x.A$. *We will also omit the subscript from the quantifier when the type of the variable x is clear from context.*

This convention, in combination with the infix conventions for the logical connectives, allows us to write things that look exactly like the sentences of predicate logic we are used to.

Exercise 3.5. *Write the following sentences of predicate logic explicitly with* λ*s in the correct places.*

a. $\forall x(Fx \to Gx)$.

b. $\forall x \exists y Rxy$.

Exercise 3.6. *Suppose that your signature contains* $\neg : t \to t$ *and* $\forall : (e \to t) \to t$.

a. Given a predicate, $F : e \to t$, *create a sentence that says that F is existentially satisfied, by appealing to the idea that existential quantification is the dual of universal quantification.*

b. Define an expression of type $(e \to t) \to t$ *that corresponds to existential quantification.*

By the way, we will adopt the same conventions of omitting outer brackets and left associated brackets in application terms. For λ-expressions there are a couple of further useful abbreviations:

Convention 3.3. *We write:*

$\lambda x.MN$ *for* $(\lambda x.(MN))$
$\lambda x_1 \ldots x_n.M$ *for* $(\lambda x_1.(\lambda x_2 \ldots (\lambda x_n.M) \ldots))$

3.2 Combinators

Let us start with an important concept:

Definition 3.3 (Combinator)**.** *A* combinator *is a closed expression of* $\mathcal{L}(\emptyset)$.

In other words, a combinator is any closed expression you can build out of λ and variables alone: no constants or free variables appear in a combinator. (The fact that combinators do not contain constants follows from the fact that they are expressions of the language in the empty signature, $\mathcal{L}(\emptyset)$. Combinators are thus also expressions of $\mathcal{L}(\Sigma)$ for arbitrary signatures Σ.)

We have encountered several combinators already in Chapter 2. We now attend to them in some more detail. (The following consolidates the material in the exercises at the end of that chapter.) For any type, σ, there is an identity combinator of type $\sigma \to \sigma$, which we called I, that informally maps a thing to itself:

$$I := \lambda x.x$$

where $x : \sigma$. Similarly, there is a curried binary operation that takes a predicate, F, and a name, a, and produces the result of applying the former to the other: Fa. That is, for arbitrary predicate $X : t$ and $y : e$ it yields $Xy : t$, and thus may be defined:

$$\lambda Xy.Xy$$

recalling that we are writing λXy as short for $\lambda X \lambda y$. More generally, we have the combinator $App : (\sigma \to \tau) \to \sigma \to \tau$, which takes an argument X of type $\sigma \to \tau$ and an argument y of type σ and yields the result of applying the former to the latter, Xy. It is defined exactly as above, except one must use variables of the appropriate type.

Both application and identity (*App* and *I*) are really an infinite class of combinators, one for each choice of the types. Thus strictly we shall write I^σ to represent the I defined in terms of a variable $x : \sigma$, and $App^{\sigma\tau}$ for the combinator defined in terms of the variables of type $X : \sigma \to \tau$ and $y : \sigma$ respectively.

Earlier we showed how to define the composition of two operators, $\Box$ and $\neg$, using λ: $\lambda p.\Box(\neg p)$. More generally we showed, given two operations Y and X of type $\sigma \to \tau$ and $\tau \to \rho$, that there is another operation of type $\sigma \to \rho$ gotten by composition:

$$X \circ Y := \lambda z.X(Yz)$$

The composition operation, $\circ$, may itself be defined as a curried binary operation:

$$\lambda XY.(X \circ Y)$$

This is the combinator we called B. Spelling it out fully, without the $\circ$ abbreviation it is defined:

$$B := \lambda XYz.(X(Yz))$$

Comprehension Check 3.3. *There is really an infinite family of combinators, $B^{\sigma\tau\rho}$, for each choice of types σ, τ and ρ above. What is the type of $B^{\sigma\tau\rho}$?*

A binary relation, like *loves* $L : e \to e \to t$, has a converse, *is loved by*. If you apply the converse of *loves* to x and y, in that order, you get the claim that y loves x—i.e. Lyx. Since this is true for arbitrary x and y, the converse of may be defined:

$$L^c = \lambda xy.Lyx$$

For an arbitrary expression $R : \sigma \to \tau \to \rho$, there is another term $R^c : \tau \to \sigma \to \rho$ defined analogously: $\lambda x \lambda y.Ryx$. There is a converse operation that maps an arbitrary $X : \sigma \to \tau \to \rho$ to X^c:

$$\lambda X.X^c.$$

this combinator is called C. Expanding out the definitions it may be written:

$$C := \lambda Xyz.Xzy$$

Again, there are infinitely many operations depending on the choices of σ, τ and ρ, so the official notation is $C^{\sigma\tau\rho}$, indicating that $X : \sigma \to \tau \to \rho$, $y : \tau, z : \sigma$.

We have defined the reflexization of L above. More generally, a predicate $X : \sigma \to \sigma \to \tau$ has a reflexization $X^w : \sigma \to \tau$ defined by $\lambda z.Xzz$. The reflexization combinator, $W : (\sigma \to \sigma \to \tau) \to \sigma \to \tau$, that takes its argument and yields its reflexization, is defined by $\lambda X.X^w$. Explicitly:

$$W := \lambda Xz.Xzz.$$

W is indexed by two types, $W^{\sigma\tau}$.

Finally we have the vacuity combinator: given any $a : \sigma$, with $x \notin FV(a)$, there is the constantly a operation, $\tau \to \sigma$, defined by $a^k := \lambda y.a$, where $y : \tau$. K is the combinator that maps an arbitrary x to the constantly x operation: $\lambda x.x^k$. In other words:

$$K := \lambda xy.x$$

This combinator depends on the types of x and y, and so is officially indexed $K^{\sigma\tau}$

Notice that in the last two examples, it is slightly more perspicuous to write $\lambda X \lambda z.Xzz$ and $\lambda x \lambda y.x$, to reflect the definitions of X^w and x^k, respectively.

To familiarize yourself with constructing combinators I recommend you try the following exercises.

Exercise 3.7. *Find combinators with the following types:*

a. $(t \to t) \to t \to t$

b. $((t \to t) \to t) \to t$

c. $e \to t \to t$

d. $e \to t \to e \to t$

Exercise 3.8. *Suppose there is a combinator of type* σ*. Show, for arbitrary type* τ*, that there must also be a combinator of type (i)* $\tau \to \sigma$*, (ii)* $(\sigma \to \tau) \to \tau$*, (iii)* $(((\sigma \to \tau) \to \tau) \to \tau) \to \tau$*.*

It's worth noting that not all types contain combinators. For instance, it is impossible to create a combinator of type t. This follows from the normalization theorem below, or the Curry-Howard isomorphism discussed in Appendix A. Some types, such as $t \to t$, contain exactly one combinator. There are even types that contain infinitely many:

Exercise 3.9 (The Church numerals)**.** *Find a combinator of type* $(\sigma \to \sigma) \to \sigma \to \sigma$ *that takes its argument* $X : \sigma \to \sigma$ *and* $y : \sigma$*, and yields the result of applying the former to the latter thrice. Show that there are infinitely many combinators in the type* $(\sigma \to \sigma) \to \sigma \to \sigma$*.*

Exercise 3.10 (Parallel composition)**.** *Define a combinator of type* $(\sigma_1 \to \cdots \to \sigma_n \to \tau) \to (\rho_1 \to \sigma_1) \to \cdots \to (\rho_n \to \sigma_n) \to \rho_1 \to \cdots \to \rho_n \to \tau$*.*

I.e., a combinator which takes a curried operation $X : \sigma_1 \to \cdots \to \sigma_n \to \tau$*, and operations* $y_1 : \rho_1 \to \sigma_1, \ldots, y_n : \rho_n \to \sigma_n$*,* $x_1 : \rho_1, \ldots, x_n : \rho_n$ *and yields something of type* τ*.*

3.3 Synonymy, α, β and η

We introduced λ as a device of abstraction. Using λ we can, for instance, define a predicate from a parameterizing class of sentences, and we characterized the abstracted predicate in terms of synonymy relationships with those sentences. We now make these synonymy relationships more explicit.

Crucial to this enterprise is, of course, the notion of synonymy:

Remark 3.1 (Synonymy)**.** Here, and throughout the chapter, we take the informal notion of synonymy as primitive. However, it is subject to some natural constraints. For instance, we take it to be an equivalence relation, in the sense that every term is synonymous with itself, if M is synonymous with N then N is synonymous with M, and if M is synonymous with N and N with P, then M is synonymous with P. We also assume that synonymous terms must have the same type. Finally, we assume that substituting synonymous terms within some larger term will yield synonymous results. Crucially, our notion of synonymy is taken to be applicable to terms that contain free variables: synonymous open terms must be synonymous in the strong sense that requires that substitution of one open term, M for a synonymous one, N, in a larger term preserves synonymy even when those larger terms can bind the free variables in M and N.

Clearly, the idea of a substitution—the idea of replacing one term for another in a larger term—is key. So before we discuss our principles of synonymy, lets say a little bit about substitution. The most basic thought is this: replacing M with N yields N. We need to extend this idea to other terms that might be different from M but nonetheless contain M. If I have a simple term (i.e. a constant or variable) that isn't M, then clearly replacing M with N

does nothing since M makes no appearance in the relevant term. The result of replacing M with N in a complex term of the form (PQ) can be obtained by making the replacement in P and in Q individually, and applying the results to each other. And finally, the result of replacing M with N in $\lambda x.P$ can be achieved by making the replacement in P and then prefixing 'λx.' to the result. (Note here that in the case that M is the variable x, we don't replace the x appearing in the 'λx.' with N. The reader should check for themselves that this could yield something that isn't a λ-term.) These ideas together make for a recursive definition of substitution.

Definition 3.4 (Substitution)**.** *The result of substituting N for M in P is written* $P[N/M]$*, and can be defined recursively as follows:*

1. $M[N/M] = N$
2. $a[N/M] = a$ *when* a *is a constant or a variable* y *distinct from* M.
3. $(PQ)[N/M] = P[N/M]Q[N/M]$ *provided* (PQ) *is distinct from* M.
4. $(\lambda x.P)[N/M] = \lambda x.P[N/M]$ *provided* $(\lambda x.P)$ *is distinct from* M.

We can analogously introduce a more general notion of substitution for simultaneously replacing multiple terms, $M_1 \ldots M_k$ with $N_1 \ldots N_k$ at once in another term P, which we write $P[N_1/M_1 \ldots N_k/M_k]$.[1] Now we may state our principles of synonymy.

Firstly, note that like many mathematical notations—such as $\int_a^b x^2\,dx$, or $x \mapsto f(x)$ and so on—the λ notation employs dummy variables. The identity of a dummy variable is always irrelevant, as its sole purpose is to coordinate argument positions. For instance, we could define the composition of two functions as $x \mapsto f(g(x))$ or as $y \mapsto f(g(y))$, it doesn't matter.

α If M can be obtained from N by permuting bound variables then M and N are synonymous.

In the case that M can be obtained by permuting the bound variables in N we say that M and N are 'α-equivalent'. Thus, for example, we obtain the synonymy of the following two definitions of reflexization: $\lambda Xy.Xyy$ and $\lambda Zw.Zww$. The restriction to permutations of bound variables ensures that you can't take distinct variables to the same variable: we can't obtain $\lambda z\lambda z.Rzz$ from $\lambda x\lambda y.Rxy$ by replacing y with z and x with z, for instance. To be an instance of α, if x and y are distinct bound variables, then the variables they are replaced with must also be distinct. And, of course, a variable can only be substituted for another if it has the same type. We will make this notion of permuting bound variables precise using our notion of substitution shortly, but for now lets power on to consider two more substantive criteria of synonymy.

Consider the predicate you get from abstracting 'Socrates' from the sentence 'Socrates is wise': $\lambda x.Wx$. Applied to any name, a, this predicate should yield a sentence synonymous with 'a is wise'. Of course, the predicate 'is wise' does this too. A natural conjecture is that W and $\lambda x.Wx$ are synonymous. Now, nothing we have said rules out the possibility that there are two non-synonymous predicates that always yield synonymous sentences when applied to the same name, so we are not forced to say W and $\lambda x.Wx$ are synonymous. However, recall that in Chapter 2 we simply *introduced* the idea of abstraction by this stipulation about how the abstracted predicate behaves when applied to a name. The possibility just raised—that the applicative behaviour of a predicate may fail to single out its meaning—means that this stipulation may also fail to pick out a unique meaning for a predicate formed by abstraction.

Since $\lambda x.Wx$ is not obviously a predicate we had antecedently and so not beholden to any pretheoretic commitments, we might as well stipulate that $\lambda x.Wx$ and W are synonymous, thus clearing up any potential ambiguity. This is the simplest stipulation we could make, and it helps to pin down the meaning of $\lambda x.Wx$ when its applicative behaviour might fail to do so. In fact, we will see shortly that this further stipulation clears up ambiguity in all the λ-terms, whether they have the form $\lambda x.Mx$ or not.

η Suppose that $M : \sigma \to \tau$, $x : \sigma$, and that $x \notin FV(M)$. Then $\lambda x.Mx$ and M are synonymous terms.

The condition that x not appear free in M is essential. If $R : e \to e \to t$ and $x : e$, clearly $\lambda x.Rxx$ and Rx do not mean the same thing, the former being the property of being self related, the latter of being related to x.

In the special case where $M : e \to t$, we may illustrate the principle using our English paraphrase of expressions $\lambda x.Mx$. Let M represent the predicate *is tall*. Then we have

The predicate *is an x such that x is tall* and the predicate *is tall* are synonymous.

However, we will not conceive of η as standing or falling with the synonymy of these two English phrases. Much like our treatment of the relationship between the first-order generalizations and the quantificational idioms of English, the English paraphrase is a mere approximation of $\lambda x.Mx$.[2] We should think of the intended meaning of $\lambda x.Mx$ as being pinned down by η (at least partially), much as we thought of first-order generalizations being pinned down by the quantifier introduction and elimination rules in Section 0.2. To back this up, of course, we will need to prove a version of Harris's theorem (recall Theorem 0.1), telling us that our rules really do pin down the meanings of λ-terms. We'll return to this below in Proposition 3.

The upshot of $\lambda x.Mx$ and M meaning the same is that we can treat these terms interchangeably: if we see M or $\lambda x.Mx$ appearing in a larger term, we can replace one with the other. Thus, for instance, the following ought to be true:

John is an x such that x is tall is synonymous with *John is tall*.

Someone is an x such that x is tall is synonymous with *someone is tall*.

Definition 3.5 (η-equivalence)**.** *Two terms, M and N, are immediately η-equivalent if, for some term $P : \sigma \to \tau$ and variable $x \notin FV(P)$ of type σ, one can be obtained from the other by replacing P with $\lambda x.Px$ or conversely.*

Two terms M and N are η-equivalent if there is a chain of terms, $P_1, \ldots, P_n$, when P_i is immediately η-equivalent or α-equivalent to P_{i+1}, $M = P_1$, and $P_n = N$.

When M and N are η-equivalent we write $M \sim_\eta N$.

We build it into our definition that α-equivalent terms are η-equivalent. Note that P and $\lambda x.Px$ have the same free variables because $x \notin FV(P)$. Note also that P and $\lambda x.Px$ may be substituted even in the context of a larger term containing λs that bind the variables appearing in P or $\lambda x.Px$. Take, for instance, the term $\lambda x.\forall_e(Rx)$—*being such that you R everything*—is η-equivalent to $\lambda x.\forall_e(\lambda y.Rxy)$, since Rx and $\lambda y.Rxy$ are immediately η-equivalent, even though the x gets bound by a λ in the larger term in which the replacement is happening.

η-synonymy η-equivalent λ-terms are synonymous.

Notice that our official statement of η follows from the claim that P and $\lambda x.Px$ (i.e. immediate η-equivalents) are synonymous, and our principle that synonymous expressions can be substituted while preserving synonymy.

Exercise 3.11. *Which of the following terms are η-equivalent?*

a. $(\lambda x.Fx)a$

b. $(\lambda Y.\lambda x.Yx)Fa$

c. Fa

d. $(\lambda Y.Y)Fa$

e. $(\lambda x.\lambda Y.Yx)aF$

Exercise 3.12. *a. Recall the definitions of $I^{\sigma\to\tau}$ and $App^{\sigma\tau}$, show that they are immediately η-equivalent. Thus explain how application is definable from identity.*

b. Define a notion of application $(\sigma_1 \to \cdots \to \sigma_n \to \tau) \to (\sigma_1 \to \cdots \to \sigma_n \to \tau)$ that takes a curried n-ary operation of type $\sigma_1 \to \cdots \to \sigma_n \to \tau$, and then takes n arguments of types $\sigma_1, \ldots, \sigma_n$ in that order, and applies the operation to the arguments. Explain why it is η equivalent to the identity combinator $I^{\sigma_1\to\cdots\to\sigma_n\to\tau}$.

The final criterion of synonymy is essentially a formalization of our job description for abstraction from Chapter 2. We said that the predicate that results from abstracting from the gappy sentence '... is wise and ... is old' yields a sentence synonymous with '*a* is wise and *a* is old' when applied to any name *a*. More generally, applying an abstracted expression, such as $(\lambda x.M)$, to some term N should be synonymous with the result of plugging N into the gap in the expression we abstracted from, M. The 'gap' here is represented by a variable, x, and 'plugging into the gap' just corresponds to substituting x with N in M. Thus:

β If $x, N : \sigma$ and $M : \tau$, $(\lambda x.M)N$ is synonymous with $M[N/x]$, provided N is free for x in M.

here we write $M[N/x]$ for the result of replacing x with N in M. There are some obvious side conditions that need to be in place for this idea to make sense, which we'll fuss over shortly (that 'N is free for x in M'). We'll first get the intuitive idea across with some unproblematic examples.

As before we can illustrate β using our English paraphrase of λed predicate expressions of type $e \to t$.

Socrates is an x such that x is old and x is wise means the same as *Socrates is old and Socrates is wise*

The former paraphrases the English expression $(\lambda x.Fx \wedge Gx)a$ and the latter $Fa \wedge Ga$.

It's helpful to break this calculation into two steps. The first step is to delete the outer λ of the term being applied. Deleting the λx from $\lambda x.Fx \wedge Gx$ leaves you with the open sentence:

$Fx \wedge Gx$

The second step is to replace the free variable x with the term to be substituted—a in this case:

$Fa \wedge Ga$

Let's apply this to some other examples. Earlier we offered $\lambda p.\Box(\neg p)$ as a defined operator meaning *it's impossible that*, expressing the composition of $\Box$ with $\neg$. In order for it to serve that function it must be the case, at minimum, that if we apply it to any sentence $A : t$, we get something that means the same as $\Box(\neg A)$. This is an instance of β:

$(\lambda p.\Box(\neg p))A$ means the same as $\Box(\neg A)$

As before, we can calculate this in two steps: remove the outer λ, yielding $\Box(\neg p)$, and replace the free variable with A, to get $\Box(\neg A)$.

Comprehension Check 3.4. *We also considered the reflexization of a relation as a motivation for introducing λ. Show, using β, that $(\lambda xLxx)a$ means the same as Laa*

Like with η, the synonymy claim intuitively allows us to substitute β-equivalent expressions that occur in embedded contexts.

Definition 3.6 (β-equivalence)**.** *Two terms, M and N, are immediately β-equivalent if, for some $x, Q : \sigma$ and term $P : \tau$, one can be obtained from the other by replacing $P[Q/x]$ with $(\lambda x.P)Q$ or conversely, provided Q is free for x in P.*

Two terms M and N are β-equivalent if there is a chain of terms, $P_1, \ldots, P_n$, when P_i is immediately β-equivalent or α-equivalent to P_{i+1}, $P_1 = M$ and $P_n = N$.

When terms M and N are β-equivalent we write $M \sim_\beta N$

Thus our official statement of β is:

β-synonymy β-equivalent terms are synonymous.

As with η, this follows from the simple statement that $(\lambda x.M)N$ is synonymous with $M[N/x]$ (when N is free for x in M) and our assumption that synonymous terms can be substituted for each while preserving synonymy. Observe, provided N is free for x in M, $(\lambda x.M)N$ can be substituted for $M[N/x]$ and vice versa *even in contexts that bind free variables appearing in $(\lambda x.M)N$ and $M[N/x]$*.

Let us now make good on our promise to show that the meanings of λ-terms are pinned down uniquely by these principles, so that given the assumption that something satisfies the principles (what we earlier called The Abstraction Hypothesis) we can be sure we are using them unambiguously.[3]

Proposition 3.1 (λ-terms are pinned down by β and η.)**.** *Given β and η-synonymy, the meanings of λ-terms are pinned down uniquely.*

Proof. Consider the λ-term $\lambda x.M$. Suppose there was another term $\lambda' x.M$ that satisfied the relevant instances of β-synonymy: the result of substituting $(\lambda' x.M)a$ for $M[a/x]$ in any term P will result in a synonymous term, for any term a free for x in M. Clearly $M[x/x] = M$, thus we may substitute $(\lambda' x.M)x$ for M preserving synonymy in any context. In particular, by replacing it in $\lambda x.(\lambda' x.M)x$ we get $\lambda x.M$, so that $\lambda x.(\lambda' x.M)x$ is synonymous with $\lambda x.M$. Finally, by η-synonymy, $\lambda x.(\lambda' x.M)x$ is synonymous with $(\lambda' x.M)$, yielding that $\lambda' x.M$ and $\lambda x.M$ are synonymous by the transitivity of synonymy. □

Remark 3.2. What we have shown here is that if λ satisfies η and λ' satisfies β, then λ-abstractions and λ'-abstractions are synonymous. The argument is symmetric: we could have shown the same from the assumption that λ-satisfies β and λ' satisfies η.

It's worth noting why η and β are different. One might naïvely think that you can derive η from β. For given β one show that $(\lambda x.Mx)$ and M do the same thing to any argument (when $x \notin FV(M)$): $(\lambda x.Mx)y$ is synonymous with $Mx[y/x]$, which is just My. Thus, given β, $(\lambda x.Mx)$ and M have the same applicative behaviour. But as we have already pointed out, nothing we have said prevents two non-synonymous $\sigma \to \tau$ expressions yielding synonymous expressions when applied to any argument. (If we could model properties faithfully by functions, then two properties that spit out the same proposition when applied to the same arguments would indeed be identical. However, the principle of Functionality—discussed in Section 6.5—is a substantive principle of higher-order logic, and without something like it we cannot straightforwardly derive η from β.)

The relation between η and β can perhaps be better captured as follows. η can be thought of as saying that λ-abstraction is the left inverse of application to a (fresh) variable, and β as saying that it is the right inverse. An operation, f, is a right inverse of another, g, when doing f and then g takes you back to the thing you started with (in functional notation $g \circ f = id$). f is a left inverse of g when doing g then f gives you back the thing you started with (i.e. $f \circ g = id$). If I apply M to a fresh variable, x, to get Mx, and then apply λ-abstraction I get $\lambda x.Mx$: η tells us that this is what we started with. If we take M and λ-abstract, getting $\lambda x.M$, and then apply to the variable x, we get $(\lambda x.M)x$ which by β is $M[x/x]$, i.e. M. So β tells us this is what we started with.[4]

Exercise 3.13. *Which of the terms in Exercise 3.11 are β-equivalent?*

Exercise 3.14. *Show that $(\lambda X.\lambda y.Xy)(\lambda z.z)a$ and a are β-equivalent, by explicitly finding the chain of immediate β-equivalents.*

It is now time to fuss over the exact conditions under which β applies. Firstly, when we move from $(\lambda xM)N$ to $M[N/x]$ we don't want a free variable occurring in N to get bound by when it is substituted for x in M by λs appearing elsewhere in M.

For instance, consider the follow relation: $(\lambda x.\lambda y.Lxy)$, where L represents the binary relation *loves*. We shall pronounce $(\lambda x.\lambda yLxy)$ *loves* because it is η-equivalent, and thus synonymous with L. The result of applying *loves* to a variable y, representing an arbitrary individual, gives us the unary predicate *loves y*, parametrized by the variable y. But if we naïvely applied β to $(\lambda x.\lambda y.Lxy)y$ we would get $\lambda y.Lyy$, the relation of loving oneself. The latter is closed, and not parametrized by any variable, but is also obviously quite different from the relation of loving any particular individual. In other words, it is clearly not synonymous with *loving y*: $(\lambda x.\lambda y.Lxy)y$.

Luckily, we can always avoid this sort of pathological use of β by appealing to α. You do this by permuting the bound variables in $\lambda x.\lambda y.Lxy$ so that they do not coincide with the free variables appearing in the argument, in this case y. E.g. $\lambda x.\lambda z.Lxz$ has the same meaning as $\lambda x.\lambda y.Lxy$, by α, but we *may* apply the former y and naïvely use β to reduce it to $\lambda z.Lyz$. More generally, if I am considering a term $(\lambda x.M)N$ and M contains bound variables that appear free in N, we may simply replace the bound variables in M so that they don't coincide with any free variables in N, and substitute as we would ordinarily. This is always possible because there are infinitely many variables of any given type, and N only contains finitely many free variables (this is proved by an easy induction on our definition of *FV*).

So it remains to say exactly when β can be applied: when N doesn't have any free variables that get bound by λs appearing in M when N replaces x. In this case, we say that N is free for x in M

Definition 3.7 (Free for x)**.** *N is free for x in M is be defined recursively as follows:*

1. *N is free for x in M whenever M is a variable or constant.*
2. *N is free for x in (PQ) provided it is free for x in both P and Q*
3. *N is free for x in $\lambda y.P$ if N is free for x in P and $y \notin FV(N)$*

Because our definition of β-equivalence allows you to relabel bound variables, the trick of relabelling bound variables in M to allow us to legitimately replace x with N is built into the notion, and we generally don't have to think too much about it.

It is very common to define curried operations that take multiple arguments by a term followed by a string of λs, such as $\lambda xyz.M$. This may be subsequently applied to a string of arguments: $(\lambda xyz.M)abc$. By applying β three times we see that this is equivalent to $M[a/x, b/y, c/z]$—the result of substituting a for x, b for y and c for z in M. Since this pattern occurs frequently, it's advantageous to be able to recognize it: go back to Exercise 3.14. Let's write it down for the record:

Proposition 3.2. *$(\lambda x_1 \ldots x_n.M)a_1 \ldots a_n$ is synonymous with $M[a_1/x_1 \ldots a_n/x_n]$.*

Finally, we give the fully rigorous definition of α-equivalence, namely:

Definition 3.8 (α-equivalence)**.** *M is immediately α-equivalent to N iff one can be obtained from the other by replacing $\lambda x.P$ with $\lambda y.P[y/x]$ provided y is not free in P.*

M and N are α-equivalent iff there exists $P_1, \ldots, P_n$ where $M = P_1$, $N = P_n$ and P_i is immediately α-equivalent to P_{i+1} for $1 \leq i < n$.

Putting all of our notions of equivalence together gives us the strongest criterion of synonymy we will consider here: $\beta\eta$-equivalence. Terms are $\beta\eta$-equivalent when you can get one from the other using either β or η.

Definition 3.9 ($\beta\eta$-equivalence)**.** *M and N are $\beta\eta$ equivalent iff there is some sequence of terms $P_1, \ldots, P_n$ where $M = P_1$, $N = P_n$, and P_i is immediately η-equivalent, immediately α-equivalent or immediately β-equivalent to P_{i+1} for $1 \leq i < n$.*

When M and N are $\beta\eta$-equivalent we write $M \sim_{\beta\eta} N$.

As before we have built α into our definition of immediate η equivalence and β equivalence.

Let me end this discussion by recording a few fundamental facts about our notions of equivalence.

Proposition 3.3. *α, η, β and $\beta\eta$-equivalence are equivalence relations on terms.*

Proposition 3.4. *If M is $\alpha/\eta/\beta/\beta\eta$-equivalent to M' and N to N', then MN is $\alpha/\beta/\eta/\beta\eta$-equivalent to $M'N'$, and $\lambda x.M$ is $\alpha/\eta/\beta/\beta\eta$-equivalent to $\lambda x.M'$.*

Proposition 3.5. *If M and N are α, η, β or $\beta\eta$-equivalent, then M and N have the same type.*

Propositions 3.4 and 3.3 can be seen just by inspection of the definition of $\alpha/\eta/\beta/\beta\eta$-equivalence. Proposition 3.5 is proven by induction on term structure.

As an application of all these ideas, let's turn to the matter of when one combinator can be defined out of another. In order to investigate the issue of definability between combinators in full generality, we will need to introduce another important combinator. It is slightly less intuitive than the other combinators we have encountered.[5] Viewed as a curried operation,

it takes three arguments: a binary curried operation, X, of type $(\sigma \to \tau \to \rho)$, a unary operation Y of type $\sigma \to \tau$, and an expression z of type σ. You can explain its operation in two steps: it separately applies X and Y to z, to get $Xz : \tau \to \rho$ and $Yz : \tau$. Then it applies Xz to Yz, to form $Xz(Yz) : \rho$ (check for yourself that this type checks). Thus S may be defined:

$$S := \lambda XYz.Xz(Yz)$$

S is parametrized by the three types σ, τ and ρ, and we may write $S^{\sigma\tau\rho}$ when we need to be explicit which we mean. It is useful because on its intended interpretation the following two things mean the same:

$$\lambda x.(MN)$$
$$S(\lambda x.M)(\lambda x.N)$$

The significance of this is that it allows you to reduce the λ-abstract of a complex expression, $\lambda x.(MN)$, to the λ-abstracts of two simpler expressions, $\lambda x.M$ and $\lambda x.N$. We'll later use this fact to see how we might eliminate λs altogether in favour of combinators like S.

Let us prove these two things mean the same given β (and thus $\beta\eta$)

Proposition 3.6. *$\lambda x.(MN)$ is β equivalent to $S(\lambda x.M)(\lambda x.N)$*

Proof. 1. $S(\lambda x.M)(\lambda x.N)$
2. $(\lambda XYz.(Xz(Yz))(\lambda x.M)(\lambda x.N)$ (definition of S)
3. $\lambda z.(\lambda x.M)z((\lambda x.N)z)$ (applying β to λX and λY successively)
4. $\lambda z.(M[z/x])(N[z/x])$ (β on both λs)
5. $\lambda x.(MN)$ (by α)

□

Before we do anything with S, let's warm up with a simple example of definability. Earlier we introduced the combinator $B : (\tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)$ as the type theory equivalent of composition for functions. But in Chapter 2 we mentioned the notation $F; G : \sigma \to \rho$ as a way of composing operation $F : \sigma \to \tau$ and $G : \tau \to \rho$, in a way that respects the order of the arrows, rather than reversing them ($F; G$ just means the same as $G \circ F$). This corresponds to another combinator which is traditionally called B':

$$B' := \lambda XYz.Y(Xz) \text{ of type } (\sigma \to \tau) \to (\tau \to \rho) \to (\sigma \to \rho)$$

where $X : \sigma \to \tau$ and $Y : \tau \to \rho$.

B' is just the converse of B. So if we have the C combinator and B, we can define B' as CB. Of course, this doesn't strictly make sense until we've specified which versions of C and B we are using. Recall that $C^{\sigma\tau\rho}$ takes something of $\sigma \to \tau \to \rho$ and yields something of type $\tau \to \sigma \to \rho$. In this case, $B^{\sigma\tau\rho}$ has type $(\tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)$, and we want $B'^{\sigma\tau\rho}$ of type $(\sigma \to \tau) \to (\tau \to \rho) \to (\sigma \to \tau)$, so the relevant instance of C is: $C^{(\tau\to\rho)(\sigma\to\tau)(\sigma\to\rho)}$, which has the long type:

$$((\tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)) \to ((\sigma \to \tau) \to (\tau \to \rho) \to (\sigma \to \rho))$$

Carefully stated, then, the interdefinability result says: if you have all instances of the C combinator and the B combinator, you can define every instance of the B' combinator.

Exercise 3.15. *Show that B' is $\beta\eta$-equivalent to CB (using the above typing).*

As you can see, the type superscripts quickly become lengthy and cumbersome. So it is convenient to omit them, and say things without type superscripts, when the fully rigorous statement can be inferred by assigning types to the relevant combinators in the only way that makes sense. Sometimes this is not possible. For instance, consider another example of redundancy: once you have the S and K combinators, you may define the I combinator.

$$I = SKK$$

This example is more complicated because in order for SKK to be well typed, the two Ks have to have different type superscripts. There are also many choices of type superscripts that would make this well-typed: it turns out that every choice well define the same entity. For any type τ, the following assignment of superscripts will work:

$$I^{\sigma} = S^{\sigma(\tau\to\sigma)\sigma} K^{\sigma(\tau\to\sigma)} K^{\sigma\tau}$$

Comprehension Check 3.5. *Show that the above expression is well-typed and has type $\sigma \to \sigma$.*

Let's prove this equivalence. We choose variables $Z : \tau \to \sigma, x, z : \sigma, y : \tau, X : \sigma \to \tau \to \rho$, $Y : \sigma \to \tau$

1. $(\lambda XYz.Xz(Yz))(\lambda xZ.x)(\lambda xy.x)$
2. $\lambda z.(\lambda xZ.x)z((\lambda xy.x)z)$ (β on λX and λY)
3. $\lambda z.(\lambda Z.z)((\lambda xy.x)z)$ (β on the first λx)
4. $\lambda z.z$ (β on λZ)

Notice that in step 3 we could have applied β to the second λx, to get from $(\lambda xy.x)z$ to $\lambda y.z$. But we didn't bother because $\lambda Z.z$ eats its argument, and spits out z no matter what.

The practice of omitting type superscripts increases readability, and allows one to perform calculations on λ-terms without getting bogged down with type annotations. But it must be applied carefully, as it is possible to write down sequences of combinators without type supescripts that cannot be coherently assigned type superscripts.

Exercise 3.16. *Show that there is no way of assigning type superscripts to SII in way that makes it a term.*

We have shown that some combinators can be defined in terms of others, and we have used our informal criterion of synonymy to show that those definitions really capture the meanings of the defined combinators. The following question is now quite salient: can we get by with only finitely many different sorts of combinators? Can every combinator be defined by applying a fixed finite collection of different sorts of combinators to one another. Note that it isn't possible to get by with finitely many combinators *tout court*: a combinator (or indeed, any expression) can only be applied to finitely many combinators before it results in an expression with a base type, and thus cannot be applied any further (see Theorem 1.1). Thus one can only create finitely many terms from a finite set of terms using application alone. However, it's natural to wonder, for example, whether all combinators are definable from all the different versions of the S and K combinators (in this case the answer is 'yes'), or from the different versions of the B and W combinators (in this case the answer is 'no').

In these examples, we have only two sorts of combinators, but each sort has infinitely many instances. We treat these questions in Section 3.5. But first, a small digression.

3.4 Reduction

Before we move on I should mention that a traditional book on the λ-calculus would typically introduce the symmetric relations β, η and $\beta\eta$-equivalence by way of another asymmetric relation called *reducibility*. β-reduction is a relation that holds between a term of the form $(\lambda x.M)N$ and $M[N/x]$, but not conversely, and η-reduction between $\lambda x.Mx$ and M but not conversely.[6] These directed relations have special meaning for computer scientists modeling functional programming languages using λ-languages because they correspond more closely to the temporal direction in which a program (represented by a λ-term) executes. A program terminates when it is represented by a λ-term that reduces to a term that cannot be reduced any further—the *normal form* of that term. If the term has a base type its normal form is called the *value* or *output* of the program, which could be something like a number or boolean value presented in some standard notation.

It turns out that if the only reduction rules are β and η then *every* term of a λ-language terminates in a normal form. Since this temporal notion is less obviously relevant for philosophical applications I shall simply state the relevant definitions and results here. The interested reader should consult a standard text, such as Hindley and Seldin (2008), for more details.

Definition 3.10 (β, η and $\beta\eta$ reduction)**.** *P immediately β-reduces to Q iff we may replace some term of the form $(\lambda x.M)N$ in P with $M[N/x]$ and produce Q, provided N is free for x in M. If we can turn P into Q through some (possibly empty) chain of immediate β-reductions or relabellings of bound variables we say that P β-reduces to Q.*

P immediately η-reduces to Q iff we may replace some term of the form $(\lambda x.Mx)$ in P with M, provided $x \notin FV(M)$, and produce Q. If we can turn P into Q through some (possibly empty) chain of immediate η-reductions or relabellings of bound variables we say that P η-reduces to Q.

If we can turn P into Q through some (possibly empty) chain of immediate β or η-reductions or relabellings of bound variables we say that P $\beta\eta$-reduces to Q.

An important fact, that we shall sometimes appeal to, is that reduction is *confluent*: if two terms are β-equivalent then they β-reduce to a common term, and similarly for η and $\beta\eta$-reduction.

Theorem 3.1 (The Church-Rosser Theorem)**.** *If P and Q are β-equivalent ($\beta\eta$-equivalent) then there is some term, T, such that P and Q β-reduce ($\beta\eta$-reduce) to T.*

Definition 3.11 (Normal forms)**.** *A term P is in β (η, $\beta\eta$) normal form iff it can only be β (η, $\beta\eta$) reduced to α-equivalents of P.*

A β (η, $\beta\eta$) normal form of a term P is a term β (η, $\beta\eta$) equivalent to P that is in β (η, $\beta\eta$) normal form.

A corollary of the Church-Rosser theorem is that normal forms are unique up to α-equivalence. If S and T are two β normal forms of a term P then they are both β-equivalent to P and thus to each other. So by the Church-Rosser theorem, they both β-reduce to a common term U. By the definition of β normal form, S and T are both α-equivalent to U,

and thus to each other. This gives us the uniqueness part of the following theorem; the rest of the proof can be found in Mitchell (1996), Section 8.3.2.

Theorem 3.2 (Normalization)**.** *Every term of the full λ-language over a signature has a unique β (η, $\beta\eta$) normal form, up to α-equivalence.*

3.5 Combinatory languages

In Section 3.1 we augmented typed applicative languages with the λ device to overcome the limitations of those languages. In this section we will introduce another approach to type theory, due to Moses Schönfinkel and Haskell Curry, which instead takes combinators, like B, C, K and W, as primitive operations.[7] This approach deviates from the approach of the last section by taking combinators as basic unanalyzed constants, instead of defining them in terms of λ. The resulting theory is interesting because it does away with variables altogether, allowing one to avoid the often fiddly issues associated with manipulating variable binders.

Definition 3.12 (Combinatory languages)**.** *If $X \subseteq \{B, B', C, K, I, W, S\}$ then $CL(X, \Sigma)$ denotes the typed applicative language over the signature Σ with additional constants corresponding to all versions of combinators belonging to X. When Σ is empty we omit it. For example $CL(\{S, K\})$ contains, for each types σ, τ, ρ, constants:*

$$S^{\sigma\tau\rho} : (\sigma \to \tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$$
$$K^{\sigma\tau} : \sigma \to \tau \to \sigma$$

For now, consider the combinatory language that contains all the listed combinators. The two examples we used to illustrate the expressive weakness of applicative languages was that there was no way of constructing composed operators, or contracted relations. But in a combinatory language, the impossibility operator can simply be expressed with the composition combinator:

$$B\Box\neg$$

And the *loves oneself* can be defined in terms of *loves* using the reflexization combinator:

$$WL$$

What about the more complicated cases where we used λ to bind variables deep inside an expression. Recall, for instance, that we couldn't directly express the first-order sentence $\exists x(Fx \wedge Gx)$ without λ-abstraction. We had to use λ-abstraction to construct a predicate out of the predicates F and G, to which we could apply $\exists : (e \to t) \to t$. But it turns out we can express $\lambda x.(Fx \wedge Gx)$—$\lambda x.(\wedge Fx)(Gx)$ in prefix notation—entirely in terms of combinators, $\wedge$, F and G, without appealing to λ-abstraction.

$$S(B\wedge F)G$$

To see why the two are equivalent, consider what $B\wedge F$ means. Firstly note that we are using prefix notation: B is taking two arguments, $\wedge$ and F (as opposed to the nonsensical infix way of reading this formula, where $\wedge$ takes B and F as arguments). $B\wedge F$, is the result of

composing $F : e \to t$ with $\wedge : t \to t \to t$, the result of which takes an individual to an operator $e \to t \to t$. Specifically, the composition of $\wedge$ and F is the operation that takes an individual, a, to the operator, $\wedge(Fa)$: conjunction with Fa. Now SXY is the operation that takes an arbitrary individual, a, applies X to it, and Y to it separately, and then applies the former result to the latter. In the present case, applying first argument of S, $B\wedge F$, to a yields *conjunction with Fa* and applying the latter argument, G, to a, yields Ga. So if we apply *conjunction with Fa* to Ga we get the conjunction of Fa and Ga, as required.

Let's consider an example involving multiple variables being bound: $\forall x \exists y Rxy$. This can be captured as follows.

$$\forall(B\exists R)$$

$(B\exists R)$ is the result of composing $R : e \to (e \to t)$ with $\exists : (e \to t) \to t$, to produce a predicate $e \to t$ (notice that, in this bracketing, Rs codomain type is the same as the domain type of $\exists$, so this composition makes sense). It takes an arbitrary x to the proposition $\exists(Rx)$, stating that the predicate Rx is satisfied by something—i.e. x bears R to something (notice, in passing, the equivalent meanings of $\exists(Rx)$ and $\exists y(Rx)y$). Thus $\forall(B\exists R)$ is the claim that everything bears R to something.

Hopefully, the above examples make plausible the conjecture that any λ-expression can be paraphrased in terms of combinators. Indeed, unless this conjecture were true, my opening claim that combinatory languages serve as a variable-free substitute for λ-languages would be false advertising. To make good on this conjecture, we need a systematic way of translating terms expressed in terms of λs into terms expressed in terms of combinators. At this point, we'll only aim to provide a mapping from one kind of term to the other in a way that intuitively preserves meaning, as encoded by our assumptions η and β. As usual, there are assumptions being made when we say one sort of term 'means the same as another': we'll bear down on these assumptions later.

Since $CL(X, \Sigma)$ does not contain variables, its terms are all closed. In order to raise the question of what it means to eliminate variables, it is convenient to consider a combinatory language that also contains variables. The reason for needing an intermediary language with variables, in broad strokes, is this. Ultimately we want to produce a translation taking λ-terms, such as $\lambda x.\neg(Fx)$ to combinatory terms not involving λ. Such a translation will most naturally be defined inductively, in terms of the structure of the λ-terms. So, for instance, to translate $\lambda x.\neg(Fx)$ we first need to find a combinatory term that translates $\neg(Fx)$. But of course, this term has a free variable x, so its combinatory paraphrase also needs to have a free variable x. Of course, once we know how to translate every λ-term with a combinatory term with the same free variables, we get as a special case a translation of every closed λ-term in our combinatory language without variables.

So let $CL^v(X, \Sigma)$ denote the combinatory language you get by adding a collection of typed variables Var to the signature with infinitely many variables of each type (i.e. treating them like new constants, subject to the same term formation rules). What does it mean to express λ-abstraction using combinators alone? Suppose P is a combinator expression that contains exactly one free variable, x. The hope would be to find another term of a combinatory language that contains no free variables (or λs), only combinators, that 'means the same as' $\lambda x.P$ would, were we operating a language with λ-abstraction treating all occurrences of combinators in P as though they were metalinguistic abbreviations for their corresponding definitions in the full λ-language ($K^{\sigma\tau}$ is short for $\lambda xy.x$, etc.). Actually, the goal we've

just described isn't quite general enough, if we want to paraphrase expressions involving multiple λs binding different variables. What we would like, in general, is:

> Given a combinatory term P involving variables $x, y_1, \ldots, y_n$, find another combinatory term, written $[x].P$, involving only the variables $y_1, \ldots, y_n$ that, informally, means the same as the λ expression $\lambda x.P$ does in $\mathcal{L}(\Sigma)$.

Note that $[x].P$ is not a term that will anywhere involve the symbol $[x]$: it is simply a shorthand for an expression involving combinators in X, and one less variable than P. For this reason, we'll refer to it as *ersatz abstraction*. Once this operation is defined, we should be able to find a translation for any λ-term in a combinatory language with variables by simply replacing λs with the above combinatory substitute. And, as emphasized earlier, as a special case we will be able to find a paraphrase of any *closed* λ-term in a combinatory language without variables, vindicating the claim that we can do away with variables altogether.

In what follows we shall be talking about translations between combinatory and λ-languages. In order to help us keep track we shall keep to the convention of using P and Q for combinatory terms, and M and N for λ-terms.

Let's begin with the set of combinators $X = \{S, K, I\}$. If P is a term of a combinatory language with variables, then it is either (i) a variable $x \in Var^\sigma$, (ii) a constant (combinatory or otherwise), $c \in \Sigma \cup \{S, K, I\}$, (iii) of the form (PQ) for term $P : \sigma \to \tau$ and $Q : \sigma$. Let's consider these in turn. Suppose it is a variable. Then it is either the very variable we are attempting to λ, namely $x : \sigma$, or it isn't. If it is, then $[x].P$ may simply be defined as the identity combinator:

$$[x].x = I^\sigma$$

Suppose that the variable is a different variable, $y : \tau$. Then $[x].y$, informally is the operation of type $\tau \to \sigma$ that constantly outputs y, regardless of the input. So:

$$[x].y = K^{\sigma\tau} y$$

Similarly, when P is a constant, $c : \tau$ we have:

$$[x].c = K^{\sigma\tau} c$$

Finally, we have the case (PQ). We may suppose that by induction we already know how to paraphrase $[x].P$ and $[x].Q$. $[x].(PQ)$ is the operation that takes an argument a, and results in $(PQ)[a/x]$. This is equivalent to substituting a for x in P and Q separately: $(P[a/x])(Q[a/x])$. And $P[a/x]$ is just the result of applying $[x].P$ to a, and $Q[a/x]$ the result of applying $[x].Q$ to a. Thus we have shown that operation $[x].(PQ)$ can be redescribed as follows: applying it to a is the same as separately applying $[x].P$ to a, and $[x].Q$ to a, and then applying the former result to the latter. This is exactly what S does.

$$[x].(PQ) = S([x].P)([x].Q)$$

Indeed, this corresponds to Proposition 3.6, proved earlier for λ-languages: $S(\lambda x.M)(\lambda x.N)$ means the same as $\lambda x.(MN)$.

Putting this all together:

Definition 3.13 (Ersatz abstraction)**.** *Suppose that P is a term of $CL^v(\{S, K, I\}, \Sigma)$. Then $[x].P$ is defined as follows:*

- $[x].x = I$
- $[x].y = Ky$, *and* $[x].a = Ka$ *when* $a \in \Sigma \cup \{S, K, I\}$.
- $[x].PQ = S([x].P)([x].Q)$

Exercise 3.17. *Calculate the following, assuming that* $\wedge : t \to t \to t, F : e \to t \in \Sigma$.*:*

a. $[x].Sxy$ *where* $x : t \to t \to t$, $y : t \to t$, *and* $S : (t \to t \to t) \to (t \to t) \to t \to t$

b. $[x].Fx$, *where* $F : e \to t$ *is a constant, and* $x : e$

c. $[x].(Fx \wedge Gx)$

d. $[x].SK$, *choosing appropriate type superscripts*

In what follows we will write $[x][y].P$, when P is a combinatory term, as short for the result of first calculating $[y].P$, to yield some combinatory term Q, and then calculating $[x].Q$. Notice that this order of calculation is no accident: it is not possible to calculate the outer $[x]$ before the inner $[y]$, as '$[y].P$' is not a term of a combinatory language, it is just a shorthand for one. We employ similar conventions for $[x][y][z].P$.

Exercise 3.18. *Calculate*

a. $[z].X(Yz)$

b. $[Y][z].X(Yz)$

c. Estimate the length of $[X][Y][z].X(Yz)$ *without explicitly calculating it.*

d. Use part c to explain how one can define the B combinator from S, K and I.

e. Explain how you might define the C combinator in terms of S, K and I.

How do we know that our definition of λ-abstraction really expresses λ-abstraction? What we need is a translation between $CL^v(\{S, K, I\}, \Sigma)$ and $\mathcal{L}(\Sigma)$ that takes expressions of one language in variables $x_1, \ldots, x_n$, to expressions of the other language in the same variables, that preserves meaning (in an obvious way). Then, if P in $CL^v(\{S, K, I\}, \Sigma)$ translates to $M \in \mathcal{L}(\Sigma)$, then we want to show that the term $[x].P$ in $CL^v(\{S, K, I\}, \Sigma)$ translates to something that means the same as the term $\lambda x.M$ in $\mathcal{L}(\Sigma)$. This strategy requires we have a meaning preserving translation between combinatory languages and λ-languages, and that we know when two λ-terms are synonymous. We have already given sufficient conditions for synonymy between two λ-terms. For the translation we choose something that just obviously preserves meaning—which maps the S combinator of a combinatory language to the S combinator of a λ-language, constants in the combinatory language to the same constant in the λ-language, and so on:

Definition 3.14 (Translating from combinatory languages to λ-languages)**.** *We provide a translation* $* : CL^v(\{S, K, I\}, \Sigma) \to \mathcal{L}(\Sigma)$*:*

1. $(a)^* = a$ *for* $a \in Var \cup \Sigma$
2. $(S^{\sigma\tau\rho})^* = \lambda XYz.Xz(Yz)$, *where* $X : \sigma \to \tau \to \rho$, $Y : \sigma \to \tau, z : \sigma$
3. $(K^{\sigma\tau})^* = \lambda xy.x$ *where* $x : \sigma, z : \tau$
4. $(I^{\sigma})^* = \lambda x.x$ *where* $x : \sigma$
5. $(PQ)^* = (P^*Q^*)$

An analogous translation from $CL^v(X, \Sigma)$ *to* $\mathcal{L}(\Sigma)$ *can be provided for every* $X \subseteq \{B, B', C, K, W, I, S\}$.

Remember here that the term I^σ in a combinatory language is just a basic constant, whereas in a λ-language we use I^σ as short for a complex term defined out of λs and variables.

Exercise 3.19. *Translate the following terms of* $CL^v(X, \Sigma)$ *to the full* λ*-language.*

a. SKK, assigning type superscripts in any way that makes sense (e.g. $S^{\sigma(\tau\to\sigma)\sigma}K^{\sigma(\tau\to\sigma)}K^{\sigma\tau}$*)*

b. BCK as above.

Notice that it's possible to write things that can't consistently be given type superscripts, like *SII*, and still 'apply' the translation, to get an ill-formed term made out of λs for which there's no way of assigning types to the variables.

We have postulated that this translation preserves meaning. Let's give this thesis a fancy name:

The Synonymy Thesis Any term P of $CL^v(X, \Sigma)$, is synonymous with $(P)^*$ in $\mathcal{L}(\Sigma)$

Like the Church-Turing thesis, this is stated in informal terms and cannot by proved formally. But it is a natural thing to postulate given the way we have been understanding the combinators. We will continue to take synonymy as a primitive concept in what follows: an equivalence relation that can hold both internally between two combinatory terms or two λ-terms, and between terms of the two different languages as well. Although the Synonymy Thesis cannot be proved, it can be broken up into to simpler theses. It is just equivalent to (i) the claim that the (primitive) combinatory term S is synonymous with the (defined) λ-term S, that the combinatory term K is synonymous with the λ-term K, and so on, and (ii) the claim that if the combinatory terms P and Q are synonymous with the λ-terms M and N respectively, then (PQ) is synonymous with (MN).

Definition 3.15 (Meaning preservation)**.** *A mapping taking each term* M *of* $\mathcal{L}(\Sigma)$ *to a term* M^+ *of* $CL^v(\Sigma)$ *is* meaning preserving *iff* M *and* M^+ *are synonymous for every term* M*. A mapping taking combinatory terms to* λ*-terms is meaning preserving if the converse condition holds.*

Theorem 3.3. *For any term* $P \in CL^v(\{S, K, I\}, \Sigma)$*,* $\lambda x.(P)^*$ *and* $([x].P)^*$ *are* β*-equivalent terms in the language* $\mathcal{L}(\Sigma)$*.*

Proof. We show this by induction on term structure. If P is the variable x :

$$\begin{aligned}\lambda x.(x)^* &= \lambda x.x \text{ by definition of } (x)^* \\ &= (I^\sigma)^* \text{ by definition of } {}^* \text{ for } I^\sigma \\ &= ([x].x)^* \text{ by definition of } [x] \text{ as applied to the term } x\end{aligned}$$

Using, in order, the definition of $*$ on x, on I^σ, and the definition of $[x]$ for the variable x. If P is a constant (S, K, I or from Σ) or variable distinct from x, a, then

$\lambda x.(a)^*$ is immediately β-equivalent to $(\lambda yx.y)(a)^*$
$= (K^{\sigma\tau})^*(a)^*$ by definition of $*$ on K

$= (K^{\sigma\tau}a)^*$ by definition of $*$ for application.
$= ([x].a)^*$ by definition of $[x]$

Finally, suppose we have combinatory terms $P : \sigma \to \tau$ and $Q : \sigma$, and we want to show $\lambda x.(PQ)^*$ and $([x].PQ)^*$ are β-equivalent. We may suppose for induction that $\lambda x.(P)^*$ and $([x].P)^*$ are synonymous, and $\lambda x.(Q)^*$ and $([x].Q)^*$ are β-equivalent. We have:

$\lambda x.(PQ)^* = \lambda x.(P^*Q^*)$ by definition of $*$
which is β-equivalent with $S(\lambda x.P^*)(\lambda x.Q^*)$ by Proposition 3.6
which is β-equivalent with $S([x].P)^*([x].Q)^*)$ by inductive hypothesis.
$= S^*([x].P)^*([x].Q)^*)$ by definition of $(S)^*$
$= (S([x].P)([x].Q))^*$ by definition of $*$ for application (twice)
$= ([x].(PQ))^*$ by applying definition of $[x]$. □

Given the Synonymy Thesis, and the assumption discussed in Section 3.3 that β-equivalent terms of a λ-language are synonymous, we can draw several corollaries from our theorem:

Corollary 3.1 (Ersatz β)**.** *Let $P : \sigma \to \tau$, $Q : \sigma$ be terms of $CL^v(X, \Sigma)$. Then given β and the Synonymy Thesis, $([x].P)Q$ is synonymous with $P[Q/x]$.*

Note, while we defined the notion of substitution for λ-languages, the definition applies straightforwardly to combinatory languages (we can just ignore the clauses dealing with λ-abstraction).

Proof. By the Synonymy Thesis we know that $([x].P)Q$ is synonymous with $(([x].P)Q)^*$, and by Theorem 3.3 we know that this is β-equivalent to $(\lambda x.P^*)Q^*$, which is β-equivalent to $P^*[Q^*/x]$. By a straightforward induction on the complexity of P, one can see that this is the same as $(P[Q/x])^*$, which by the Synonymy Thesis is synonymous with $P[Q/x]$ as required. □

We can also prove a version of η for ersatz abstraction using the stronger idea that $\beta\eta$-equivalent terms in a λ-language are synonymous.

Corollary 3.2 (Ersatz η)**.** *Given η and the Synonymy Thesis, $[x].Px$ is synonymous with P, when $x \notin FV(P)$.*

Proof. $[x].Px$ is $S([x]P)I$, applying the definition of $[x]$.
By the Synonymy Thesis, $S([x]P)I$ is synonymous with $(S([x]P)I)^*$, which is $S([x]P)^*I$ applying the translation. By Theorem 3.3, this is β-equivalent to $S(\lambda x.P^*)I$. This is $\beta\eta$-equivalent to P^*, which can be shown by the following chain of β-equivalent terms:

1. $S(\lambda x.P^*)I$
2. $\lambda z.(\lambda x.P^*)z(Iz)$ by β on the definition of S
3. $\lambda z.(\lambda x.P^*)zz$ by β on the definition of I
4. $\lambda z.P^*[z/x]z$ by β and λx

5. $\lambda z.P^*z$, since (i) $FV(P) = FV(P^*)$ (easy induction) and (ii) $x \notin FV(P)$. So $P^*[z/x] = P^*$
6. P^* by η

Finally, P^* is synonymous with P by the Synonymy Thesis, as required. □

Observe that there is no ersatz version of α in a combinatory language for the following simple reason: there are no bound variables. We can, however, ask whether our definition of ersatz abstraction depends on which variable we abstract on. It turns out, as one would expect, that it doesn't: $[x]P$ and $[y](P[y/x])$ define the exact same term of a combinatory language.

Exercise 3.20. *The following question is about the language* $CL^v(\{S, K, I\}, \Sigma)$. *Show that:*

a. $[x].x$ *is the same term as* $[y].y$.

b. $[x].a$ *is the same term as* $[y].a$ *when* a *is a constant or variable distinct from* x.

c. If $[x].P$ *is the same term as* $[y].P[y/x]$, *and* $[x].Q$ *is the same as* $[y].Q[y/x]$ *then* $[x].PQ$ *is the same term as* $[y].(PQ[y/x])$.

d. Conclude, by induction, that $[x]P$ *and* $[y].P[y/x]$ *are the very same term for arbitrary* P.

These facts together show that ersatz abstraction really does what we want it to.

We've now laid the groundwork for defining a translation in the other direction, allowing us to take a term of the full λ-language and find a term in a combinatory language that means the same as it.

Definition 3.16 (Translation from λ-languages to combinatory languages)**.** *We define a translation* $\dagger : \mathcal{L}(\Sigma) \to CL^v(X, \Sigma)$ *as follows:*

1. $(a)^\dagger = a$, $a \in \Sigma \cup Var$
2. $(MN)^\dagger = M^\dagger N^\dagger$
3. $(\lambda x.M)^\dagger = [x].(M)^\dagger$

Exercise 3.21.

a. Calculate $(Xz(Yz))^\dagger$.

b. Calculate $(\lambda z.Xz(Yz))^\dagger$.

c. Estimate the length of $(\lambda Yz.Xz(Yz))^\dagger$ *and* $(\lambda XYz.Xz(Yz))^\dagger$.

d. Come up with shorter translations of parts a-c into a combinatory language that have the same meanings as the corresponding λ*-terms.*

Exercise 3.22. *Calculate:*

a. $(\forall(\lambda x.\exists(\lambda y.Rxy)))^\dagger$

b. $(\exists\lambda x(Fx \wedge Gx))^\dagger$

c. $(\lambda Xy.Xy)^\dagger$

d. $(\lambda Xyz.Xzy)^\dagger$

Theorem 3.4. M *is* β*-equivalent with* $M^{\dagger *}$ *for every* $M \in \mathcal{L}(\Sigma)$.

Proof. We prove it by induction. Clearly $a^{\dagger *}$ is identical, and thus β-equivalent with a for each $a \in \Sigma \cup Var$.

Suppose M is β-equivalent to $(M)^{\dagger *}$ and N to $(N)^{\dagger *}$. Then, by Proposition 3.4, MN is β-equivalent with $(M)^{\dagger *}(N)^{\dagger *} = (M^{\dagger}N^{\dagger})^{*} = (MN)^{\dagger *}$.

Finally, suppose M is β-equivalent to $M^{\dagger *}$. $(\lambda x.M)^{\dagger *}$ is $([x].M^{\dagger})^{*}$, which is β-equivalent to with $\lambda x.M^{\dagger *}$ by Theorem 3.3. Which by Proposition 3.4 is β-equivalent to $\lambda x.M$. □

Given our assumption that β-equivalents are synonymous (i.e. β) we can appeal to Theorem 3.4 to show that $M^{\dagger *}$ and M are synonymous. As we have not stated any formal notion of equivalence between combinatory terms (i.e. an analogue of $\beta\eta$-equivalence), we do not have a straightforward analogue of Theorem 3.4 for combinatory languages: i.e. a theorem stating some precise connection between $P^{*\dagger}$ and P for terms P of a combinatory language. But given the Synonymy Thesis and β, we can use Theorem 3.4 to establish a theorem in terms of our informal notion of synonymy: that for every term of a combinatory language, P, $P^{*\dagger}$ and P are synonymous.

Corollary 3.3. *Given β and the Synonymy Thesis, $P^{*\dagger}$ is synonymous with P for every term P in $CL^{v}(X, \Sigma)$.*

Proof. By Theorem 3.4, $(P^{*})^{\dagger *}$ is β-equivalent with P^{*}. So by β they are also synonymous. If $(P^{*\dagger})^{*}$ is synonymous with P^{*} then, by the Synonymy Thesis $P^{*\dagger}$ is synonymous with P as required (as $*$ cannot map non-synonymous terms to synonymous terms). □

Corollary 3.4. *Given β and the Synonymy Thesis, $\dagger$ is meaning preserving.*

This last corollary must hold: if $M^{\dagger}$ was not synonymous with M, then $M^{\dagger *}$ would not be either, as $*$ preserves meaning (by the Synonymy Thesis). But by Theorem 3.4, they are β-equivalent and thus by β, synonymous. This would be a contradiction.

Let's take a moment to take stock. We have effectively shown that λ-languages and combinatory languages amount to the same thing expressively: anything you can express in one, you can express the other. More precisely, we have provided two meaning-preserving translations allowing you to take any term of a λ-language and come up with a synonymous term of a combinatory language, and given a term of a combinatory language, a synonymous term of a λ-language.

It's worth noting that if you open a standard textbook on combinatory logic, such as Hindley and Seldin (2008), you would get the impression that the situation is more complex that I am letting on. The two translations $*$ and $\dagger$ defined above are presented, but the translations are not presented as the inverses of one another. In particular, according to the standard account, there is a notion of equivalence on combinatory terms according to which P and $P^{*\dagger}$ are *not* equivalent. The reason for this mismatch is that the standard theory of equivalence for combinatory terms—tracing back to the work of Curry (see Curry and Feys (1958))—is significantly weaker than the formal notion of equivalence we have been adopting between λ-terms, $\beta\eta$-equivalence. For each of the combinators, Curry lists the following sufficient conditions for synonymy, where P, Q, a and b are terms of a combinatory language of the appropriate types:

S $SPQa$ is synonymous with $Pa(Qa)$

K Kab is synonymous with a

I Ia is synonymous with a

B $BPQa$ is synonymous with $P(Qa)$

C *CPab* is synonymous with *Pba*

W *WPa* is synonymous with *Paa*

These can all be derived given the Synonymy Thesis using β, as each term on the left translates to something obviously β-equivalent to the thing on the right. What is happening when it is asserted that $*$ is not the left inverse of $\dagger$ is the following: one can find a combinatory term, P, such that P cannot be shown to be equivalent to $P^{*\dagger}$ using only the principles of synonymy listed above, along with some obvious principles about synonymy.[8] But just because two terms cannot be seen to be synonymous from the above list of rules does not mean that they are *not* synonymous. For instance, we saw in Section 3.3 that (suitably typed) *SKK* and *I* are β-equivalent. But clearly the analogous synonymy judgement can't be derived from the above principles: the axiom for S can only be applied when S is supplied with three arguments, and here it only has two.

Indeed, we have shown above that given β and the Synonymy Thesis $\dagger$ and $*$ really are meaning-preserving translations that are inverse to one another (modulo synonymy). Denying this either means denying that β-equivalent terms are synonymous. Or it means denying that $*$ preserves meaning: it means, for instance, denying that I and $\lambda x.x$ mean the same thing. The latter attitude can be somewhat hard to make sense of given that we have little independent grasp of the combinator terms like I: for someone who already understands the λ-language, it's hard not to want to simply interpret it as the term you already understand as $\lambda x.x$.

Giving up β is similarly bewildering: β and η are effectively the only constraints we have put on the meanings of the λ-terms. Arguably λ-abstraction is not an operation we had prior to theorizing but something whose meaning is acquired via its theoretical role. So to lose β is to lose the only grip we had on the meaning of λ-terms. On the other hand, given β we have a very strong constraint on the meaning of a λ-terms. Indeed, as we noted in Proposition 3, we can prove that the meaning of a λ-term is pinned down by β and η.

3.6 More efficient definitions of ersatz abstraction

You may have noticed that the algorithm we gave for calculating a combinatory term $[x].P$ that behaves like the corresponding λ-abstract often produces quite lengthy expressions. Indeed, you may have found simpler ways of expressing ersatz abstracts than the thing produced by the algorithm. It turns out there are many alternative definitions of $[x].P$ that produce smaller terms, and which have other benefits.

To illustrate one of these issues, consider the translation of $[x].Fx$, where F is a constant of type $e \to t$: it becomes $S([x].F)([x].x))$, and then $S(KF)(I)$. But given that $[x].Fx$ is just supposed to mean the same as $\lambda x.Fx$, and by η this means the same as F, it would be much simpler to translate $[x].Fx$ as just F. More generally, when $x \notin FV(P)$, we could just translate $[x].Px$ as P, instead of $S([x].P)([x].x)$.

Another example of inefficiency concerns ersatz abstractions of the form $[x].M$ when $x \notin FV(M)$. If M is a constant or a variable distinct from x our original translation delivers Kc or Ky, but if it is an application, PQ, it gets translated using S—$S([x].P)([x].Q)$—and thus becomes more complex as the translation proceeds on $[x].P$ and $[x].Q$ respectively. However, when x isn't free in M, $[x].M$ means the same as KM. So in this case we could translate $[x].M$ as KM, and the translation halts.

Comprehension Check 3.6. *Let M be a term of a λ-language, where $x \notin FV(M)$. Show that $(\lambda yx.y)M$ is immediately β-equivalent to $\lambda x.M$. Explain why $(\lambda yx.y)M$ is not β-equivalent to $\lambda x.M$ when $x \in FV(M)$.*

We can actually finesse this trick a little further. We have see that we can use K instead of S in defining $[x].PQ$ when x isn't free in P or in Q. But what if x is free in P but not Q, or in Q but not P? With our new shortcut for vacuous abstraction, and the clause for application, $[x].PQ$ becomes $S([x].P)(KQ)$ or $S(KP)([x].Q)$. But we can do better. Supposing that M and N are terms of a λ-language, and S, B and C now stand for combinators in that λ-language, we have the following facts, the first of which we proved earlier (Section 3.3):

If x is free in M and N, then $\lambda x.(MN)$ is β-equivalent to $S(\lambda x.M)(\lambda x.N)$

If x does not appear free in M then $\lambda x.(MN)$ is β-equivalent to $BM(\lambda x.N)$

If x does not appear free in N then $\lambda x.(MN)$ is β-equivalent to $C(\lambda x.M)N$

This could give us an alternative way of defining $[x].(MN)$ when x appears free in only one of M and N. Namely, by defining $[x].(PQ)$ as $BP([x].Q)$ when $x \notin FV(P)$, and defining $[x].(PQ)$ as $C([x].P)Q$ when $x \notin FV(Q)$. This is beneficial, as it can lead to shorter and more convenient translations of λ-terms into combinators. Unlike our previous tricks, notice that we have now had to go outside the combinators S, K, I, and use the combinators B and C.

Let's put these simplifications together to give a more efficient algorithm for producing ersatz abstracts.

Definition 3.17 (Alternative definition of ersatz abstraction)**.** *The alternative algorithm for computing λ-abstracts can be given as follows.*

- $[x].P = KP$ *when* $x \notin FV(P)$
- $[x].Px = P$ *when* $x \notin PV(P)$
- $[x].x = I$
- $[x].(PQ) = S([x].P)([x].Q)$ *when* $x \in FV(P) \cap FV(Q)$
- $[x].(PQ) = BP([x].Q)$ *when* $x \in FV(Q)$ *and* $x \notin FV(P)$
- $[x].(PQ) = C([x].P)Q$ *when* $x \in FV(P)$ *and* $x \notin FV(Q)$

Notice how the first clause encompasses the clauses for $[x].y$ and $[x].c$ when c is a constant in our original translation.

With our original definition of $[x]$ we had an ersatz version of η, which said that $[x].Mx$ is synonymous with M when $x \notin FV(M)$. The second clause of our translation ensures a very strict version of η for ersatz abstraction: $[x].Mx$ is literally identified with M.

Exercise 3.23. *Calculate the following terms using the more efficient algorithm for ersatz abstraction.*

a. $[x].Zxy$

b. $[x].Zyx$

c. $[x].Zx(Yx)$

d. $[x][y].Zxy$

e. $[x][y].Zxx$

In the previous section, we focused on the combinatory language with the combinators $\{S, K, I\}$. However there are other combinatory bases that are complete. We noted earlier that I can be defined from S and K, thus $\{S, K\}$ is also combinatory complete.

Exercise 3.24. *Show that, with the superscripts from Section 3.3,* $(SKK)^*$ *and* I^* *are* β*-equivalent.*

More interesting combinatory bases exist. For instance, the combinators $\{B, C, K, W\}$ also form a complete combinatory basis. The strategy behind showing that some set of combinators forms a basis is always the same: to show that you can define the combinators in an already known basis from your basis. In our case, we already know that S and K form a basis.

Finding bases can often be a tricky business, so it's worth going through an example in some detail. In the above case it is clear that K can be defined, since it is simply a member of $\{B, C, K, W\}$. S can also be defined, but it is much less obvious: with suitable type superscripts it is defined by

$$B(BW)(BBC)$$

We can justify this definition as follows. Firstly, suppose that $X : \sigma \to \tau \to \rho$, $Y : \sigma \to \tau$ and $z : \sigma$. Then $\lambda z.Xz(Yz)$ can be defined as follows:

$$W(B(CX)Y)$$

It is an instructive exercise in β manipulations to see this. The derivation is given below.

1. $(\lambda Zy.Zyy)((\lambda UVz.U(Vz))((\lambda Zxy.Zyx)X)Y)$
2. $(\lambda Zy.Zyy)((\lambda UVz.U(Vz))(\lambda xy.Xyx)Y)$
3. $(\lambda Zy.Zyy)((\lambda Vz.(\lambda xy.Xyx)(Vz))Y)$
4. $(\lambda Zy.Zyy)((\lambda z.(\lambda xy.Xyx)(Yz)))$
5. $(\lambda Zy.Zyy)((\lambda z.(\lambda y.Xy(Yz))))$
6. $(\lambda y.(\lambda z.(\lambda y.Xy(Yz)))yy)$
7. $\lambda y.Xy(Yy)$

Since X (and Y) occur only once in $W(B(CX)Y)$, it is never the case that X (or Y) appears free both in M ad N, in any subterm of the form (MN). So if we apply the simplified definition of ersatz abstraction to compute $[Y].W(B(CX)Y)$, we will never encounter the clause that uses S, giving us an S-free definition of $[Y].W(B(CX)Y)$. As follows:

$$[Y].W(B(CX)Y)$$
$$BW[Y].(B(CX)Y)$$
$$BW(B(CX))$$

Finally, we compute $[X].BW(B(CX))$ using the same trick to get our definition of S, namely: $B(BW)(BBC)$.

Exercise 3.25. *Compute* $[X].BW(B(CX))$ *using the modified definition of ersatz abstraction, and explain why this means we can define S from B, C, K and W.*

Let me finally turn to another particularly nice feature of this algorithm. If you take any of the combinators, S, K, B, C, I, expressed in the λ-language, and translate them into a combinatory language using the alternative definition of ersatz abstraction, then you'll get the S, K, B, C or I constants of the combinatory language, respectively. For instance, let's translate $\lambda XYz.Xz(Yz)$. One begins by translating $\lambda z.Xz(Yz)$, by $[z].Xz(Yz)$:

1. $[z].Xz(Yz)$
2. $S([z].Xz)([z].Yz)$
3. SXY

The crucial difference between this and the original translation is that things like $[z].Xz$ don't become X, but the more complicated, $S(KX)I$. By the above calculation, $[Y][z].Xz(Yz)$ is the same as $[Y].SXY$, which for the same reason becomes SX. Finally, $[X][Y][z].Xz(Yz)$—the translation of $\lambda XYZ.Xz(Yz)$—becomes, by the above computations, $[X].SX$, and thus S.

Exercise 3.26. *Show that the translation of* $\lambda XYz.X(Yz)$ *is B,* $\lambda Xyz.Xzy$ *is C,* $\lambda xy.x$ *is K, and* $\lambda x.x$ *is I.*

Suppose we define a new translation, M^+, taking λ-terms to combinatory-terms. Like the † translation, we set $a^+ = a$ for constants and variables, $(MN)^+ = (M^+N^+)$, and $(\lambda x.M)^+ = [x].M^+$, except, unlike the † translation, in the last case we are using the refined definition of $[x]$. While we could only show that $P^{*\dagger}$ was synonymous with P, we can show that P^{*+} identical to P, in the sense of being the very same term!

Proposition 3.7. $P^{*+} = P$.

The reason is quite straightforward: we have above shown that $S^{*+} = S$, $B^{*+} = B$, and so on for each basic combinator. Moreover, if $P^{*+} = P$ and $Q^{*+} = Q$, then $(PQ)^{*+} = (P^*Q^*)^+ = (P^{*+}Q^{*+}) = PQ$. So it follows by a straightforward induction that for any term $P^{*+} = P$.

It's worth reiterating that the definition of $[x]$ builds η in by fiat. If you've done Exercise 3.26, you'll see that it is exactly this feature of the definition that ensures that $S^{*+} = S$, $B^{*+} = B$, and so on.

Endnotes

1. It may be defined as follows:
   1. $M_i[N_1/M_1 \dots N_k/M_k] = N_i$ for $i = 1 \dots k$
   2. $a[N_1/M_1 \dots N_k/M_k] = a$ when a is a constant or a variable y distinct from $M_1, \dots, M_k$.
   3. $(PQ)[N_1/M_1 \dots N_k/M_k] = P[N_1/M_1 \dots N_k/M_k]Q[N_1/M_1 \dots N_k/M_k]$
   4. $(\lambda x.P)[N_1/M_1 \dots N_k/M_k] = \lambda x.P[N_1/M_1 \dots N_k/M_k]$

   Observe that the above notion of simultaneous substitution cannot be defined by repeatedly applying single substitution, since $Rxy[y/x][z/y] = Rzz$, while $Rxy[y/x, z/y] = Ryz$.
2. For a start, 'x' is taking the position of a noun, like 'dog' in the paraphrase, but we do not think there are xs alongside dogs, so we already have reason to take the paraphrase with a pinch of salt.
3. See Bacon (2023).
4. Although this talk of left and right inverses is a little imprecise, it can be made a bit more rigorous via the categorical perspective on type theory. See Mitchell (1996) or Lambek and Scott (1988).

5. Indeed, even Haskell Curry, the name most frequently associated with combinators, felt he did not understand the S combinator, and preferred systems which treated the B, C, K and W combinators as more basic. See Seldin.
6. The relation thus seems to reflect movement from one term to a simpler term, for some sense of 'simple'. However, this notion of simplicity is hard to spell out. For instance, β-reduction doesn't reduce the number of λs: if x appears in M more than once, any λs in N get duplicated in $M[N/x]$.
7. Schönfinkel (1924), Curry and Feys (1958).
8. These include the idea that synonymy is an equivalence relation and is preserved under application.

Hints for exercises

[i] **Hint:** Argue that Z has to have the same type as Y.

Part II

Higher-order languages

4

Higher-order languages

In this chapter, we return to the signature of higher-order logic. In Section 4.1, we recall the definition of the signature of higher-order logic, and address some philosophical questions. In Section 4.2, we discuss the higher-order quantifiers and compare the present treatment to the more usual treatment of quantifiers as primitive variable binders, like λ.

4.1 Higher-order languages

Let's begin by defining the language of pure higher-order logic.

Definition 4.1 (The language of pure higher-order logic)**.** *The language of pure higher-order logic is the typed λ-language, $\mathcal{L}(\Lambda)$, based on the signature Λ of higher-order logic.*

The signature of pure higher-order logic, recall, consists of the following constants:

$\top, \bot : t$

$\neg : t \to t$

$\wedge, \vee, \to, \leftrightarrow : t \to t \to t$

$=_\sigma : \sigma \to \sigma \to t$

$\forall_\sigma, \exists_\sigma : (\sigma \to t) \to t$

We will generally prefix the word 'pure' to a language to indicate that the signature of the language contains only logical constants, like the symbols for conjunction and quantification:

Definition 4.2. *A higher-order language is a typed λ-language whose signature contains all the constants in Λ.*

Definition 4.3. *Elements of Λ will often be referred to as* logical constants*. Other constants will be referred to as* non-logical constants*.*

We can also consider other typed languages in the signature of higher-order logic and refer to them as 'higher-order languages'. For instance, the language of *SKI*-combinatory language over the signature Λ, $CL(\{S, K, I\}, \Lambda)$ is, given the synonymy thesis, capable of stating everything you can state in higher-order logic $\mathcal{L}(\Lambda)$, and conversely.

We shall adopt many of the conventions we indicated earlier when talking about logical connectives. In particular, we adopt the infix notation for $\to, \wedge, \vee, \leftrightarrow$ and $=_\sigma$, writing, for instance

$a =_\sigma b$

DOI: 10.4324/9781003039181-7

where $a, b : \sigma$, instead of $(=_\sigma a)b$.

As outlined in Convention 3.2, we generally omit λs immediately following quantifiers, writing things like:

$\exists_\sigma X.Xa$

instead of $\exists_\sigma \lambda X.Xa$.

Beyond the familiar vocabulary of the propositional calculus, and the first-order quantifiers and identity, $\forall_e, \exists_e$ and $=_e$, we also have quantifiers $\forall_\sigma, \exists_\sigma$ and identity operations $=_\sigma$ for $\sigma \neq e$. The former stand to type σ as the first-order quantifiers stand to type e. For instance, $\exists_t$ is an expression that combines with an operator expression, M, to form a sentence. More generally, $\exists_\sigma$ combines with a $\sigma \to t$ predicate to form a sentence. Their logical role should be understood by analogy with the first-order quantifiers. For instance, just as the truth of a sentence like

Fa

suffices for the truth of

$\exists_e x.Fx$

it also suffices for the truth of

$\exists_{e \to t} X.Xa$

$=_\sigma$ similarly stands to type σ as first-order identity stands to type e. So, for instance, $=_t$ is a connective, like $\wedge$, and combines with two sentences to form another sentence. It's logical role is similarly informed by an analogy with the logic of first-order identity. So, just as the truth of

$a =_e b$

suffices for the truth of an implication like

$\Box(Fa) \to \Box(Fb)$

the truth of

$A =_t B$

suffices for the truth of the implication

$\Box A \to \Box B$

You may have noticed that we have adopted a fairly large signature. It's often asserted that in the propositional calculus $\vee$ is definable from $\wedge$ and $\neg$ (or $\wedge$ from $\vee$ and $\neg$, and so on). Consequently, $\vee$ is sometimes omitted from the signature, and a convention of writing $A \vee B$ as an abbreviation for $\neg(\neg A \wedge \neg B)$ is adopted.

$A \vee B := \neg(\neg A \wedge \neg B)$

But this claim to having *defined* $\vee$ from $\neg$ and $\wedge$ is inaccurate for several reasons. Firstly, the above is a convention for defining *sentences* involving $\vee$ in terms of other sentences just involving $\neg$ and $\wedge$. This is a claim about one sort of sentence being defined in terms of another sort of sentence: it says nothing about being able to define a *connective* in terms of

other connectives. Moreover, the paraphrase in question only applies to sentences with a very particular form: a sentence where $\vee$ is flanked on either side by two other sentences A and B can be paraphrased as $\neg(\neg A \wedge \neg B)$, but if there were other ways $\vee$ could contribute to a sentence the paraphrase could not be applied.

This is not merely a verbal issue. It's true that in the propositional calculus, all occurrences of $\vee$ can be eliminated in favour of something involving $\wedge$ and $\neg$. And if that's all you care about, the proposed sense of definition is fine. But the analogous claim is not true in higher-order languages. There are occurrences of $\vee$ in sentences of higher-order logic in which $\vee$ is not flanked by two sentences, and so the envisioned strategy for replacing occurrences of $\vee$ with $\wedge$ and $\neg$ could not possibly work. Consider, for instance:

$$\exists_{(t\to t\to t)\to t} X.X\vee$$

where X is a property of connectives (has type $(t \to t \to t) \to t$). Any strategy that only allows one to paraphrase away sentences involving $\vee$ that are flanked by other sentences will not apply to the above sentence.

Luckily, the full λ-language does allow us to construct a new connective out of the connectives $\neg$ and $\wedge$ in the way envisioned, by using λ:

$$\vee := \lambda pq.\neg(\neg p \wedge \neg q)$$

Exercise 4.1. *Produce an analogous definition of* $\vee$ *in terms of* $\wedge$ *and* $\neg$ *in* $CL(\{S, K, I\}, \Lambda)$.

The second issue is more serious. What guarantees that the proposed definition is really synonymous with $\vee$? In fact, we will take issue with two synonymy claims: the claim that the connective $\vee$ is synonymyous with $\lambda pq.\neg(\neg p \wedge \neg q)$ and the weaker claim that sentences of the form $A \vee B$ are synonymous with $\neg(\neg A \wedge \neg B)$.

Let's start by considering the weaker claim. In the propositional calculus, we can prove a couple of theorems that ensure that $A \vee B$ and $\neg(\neg A \wedge \neg B)$ are always interchangeable. The first states that these two sentences are both materially equivalent, and the second states that materially equivalent sentences are intersubstitutable in all contexts. The first theorem extends to richer languages, provided the relevant propositional logic remains unchanged, but the second does not: once non-extensional contexts are introduced material equivalents are not intersubstitutable. The problem is that in higher-order logic one can define contexts that can't be proven to be extensional. For instance, from the identity

$$A \vee B =_t \neg(\neg A \wedge \neg B)$$

one can infer the higher-order identity

$$\lambda X.X(A \vee B) =_{(t\to t)\to t} \lambda X.X(\neg(\neg A \wedge \neg B))$$

That is, the property of applying to the disjunction of A and B is the same as the property of applying to the negation of the conjunction of the negations of A and B. But these higher-order properties may not even be coextensive, let alone identical. Suppose that propositions are structured: $A \vee B$ contains different fewer constituents than $\neg(\neg A \wedge \neg B)$ does. Then higher-order properties expressing constituent number can apply to one proposition but not the other.

Of course, the point can be made more simply: $A \vee B$ and $\neg(\neg A \vee \neg B)$ are distinct propositions on this structural view, and thus not synonymous. But this way of putting the point gets into side issues about what *synonymy* means, and whether it is more demanding than logical equivalence. The more substantive point here is that, once certain metaphysical views are adopted, there are truths involving $\vee$ that can't be stated using only $\wedge$ and $\neg$—such as that $\vee$ contains no proper constituents. Similar points apply to the purported definition of the connective $\vee$ in terms of the connectives $\wedge$ and $\neg$: according to some metaphysical views, $\lambda pq.\neg(\neg p \wedge \neg q)$ has more constituents than $\vee$ does.

Remark 4.1. Notice that even if we reject the structural view indicated above, and accept all propositional identities of the form $A \vee B =_t \neg(\neg A \vee \neg B)$—which we can now formalise in our higher-order signature as $\forall_t p \forall_t q.(p \vee q) =_t \neg(\neg p \wedge \neg q)$—it is not obvious that the corresponding identity between connectives is true $\vee =_{t\to t\to t} \lambda pq.\neg(\neg p \wedge \neg q)$.

To close that gap, we need something analogous to a principle of 'functionality' stating that connectives are identical if they yield identical results when applied to any pair of arguments (so called because this is how functions behave). We discuss this principle in Section 6.5. But there are also interesting metaphysical views that reject the functionality principle. For such theories, λ becomes an incredibly important device: it allows one to define things explicitly, even in contexts where equational definitions don't suffice to pin down a unique operation.

Once these points are recognized, a case could be made that our signature is too *impoverished*. It's surely just a coincidence of the history of logic that we have mostly chosen to theorize in terms of the truth-functional connectives $\top, \bot, \vee, \wedge, \neg, \to$ and $\leftrightarrow$. There are infinitely many more truth-functional connectives (for instance, there is conjunction of arity n for each n) that have just as good a claim to being logical as the connectives in this list have. But given the structural picture, none can be synonymously defined out the other connectives. So a properly expressive higher-order language ought to have primitives for all the truth-functional connectives. A similar point can be made about quantificational expressions. In first-order logic, we are used to taking one or both of the universal or existential quantifiers as primitive, from which one can define other quantifiers such as *there are at least three*, *there are exactly seven* and so on. In higher-order logic, we can provide definitions of a much wider range of quantifiers, including *most*, *many*, *there are finitely many*, *there are uncountably many*, but similar considerations may push us to take all of them as primitive. We will return to these issues in Section 9.4 where a separate set of considerations will push us to take more quantifiers as primitive.

In the other direction, there are certain metaphysical views on which, putting it roughly, logical equivalence does suffice for (higher-order) identity. On these views, we can, in fact, reduce the signature to a rather small collection of operations. Consider, for instance, the following smaller signature, Λ^-:

$$(\Lambda^-)^{t\to t\to t} = \{\to\}$$
$$(\Lambda^-)^{(\sigma\to t)\to t} = \{\forall_\sigma\}$$

It is well known that the truth-functional connectives can be defined from $\to$ and $\bot$, so to define the truth-functional connectives from Λ^- it suffices to be able to define something logically equivalent to $\bot$ from the above. One possibility is $\forall_t p.p$, for giving the quantifier a standard logic it entails A for arbitrary A. In particular, it entails $\bot$. The converse entailment is evident, so it is logically equivalent to $\bot$. Other abbreviations may be found in Table 4.1.

Table 4.1 Some common abbreviations

$\bot := \forall_t \lambda p.p$	$\neg := \lambda p.(p \to \bot)$
$\vee := \lambda pq.(\neg p \to q)$	$\wedge := \lambda pq.\neg(p \to \neg q)$
$\leftrightarrow := \lambda pq.((p \to q) \wedge (q \to p))$	$\top := \neg\bot$
$\exists_\sigma := \lambda X.\neg(\forall_\sigma y(\neg(Xy)))$	$=_\sigma := \lambda xy.\forall_{\sigma\to t} Z.(Zx \leftrightarrow Zy)$

Exercise 4.2. *Suppose that $\forall_\sigma F$ entails Fa for any terms $F : \sigma \to t$ and $a : \sigma$. Show that $\forall_{t\to t}\forall_t$ entails $\bot$, and explain why it could thus serve as an alternative (variable free) definition of $\bot$.*

Of special note is the definition of identity. Table 4.1 defines identity as higher-order indiscernibility. This relation of higher-order indiscernibility is often given a special name, in honor of Leibniz:

Definition 4.4 (Leibniz equivalence)**.** *Leibniz equivalence (at type σ) is the relation $\lambda xy.\forall_{\sigma\to t} Z.(Zx \leftrightarrow Zy) : \sigma \to \sigma \to t$*

Whether Leibniz equivalence is strictly *identical* to identity is a contentious matter for the same reason that the identity of $\vee$ and $\lambda pq.\neg(\neg p \wedge \neg q)$ was. But its logical equivalence with identity follows from some pretty uncontentious principles about identity and quantification. Let's show, informally, that first-order identity is coextensive with Leibniz equivalence at type e, using reasoning that is plausibly logical in nature. Certainly if $a =_e b$ then, by Leibniz's law an arbitrary Z applies to a if and only if it applies to b—they share the same properties—so $\forall_{e\to t} Z.(Za \leftrightarrow Zb)$ is true. Conversely, if a and b share all the same properties, and since a has the property of being identical to a (i.e. $\lambda x.(x = a)$), b is identical to a too, which we can infer by instantiating the universally quantified Z with $\lambda x.(x = a)$.

Comprehension Check 4.1. *The claim 'Leibniz equivalence is identical to identity' is obviously a sloppy formulation of a precise idea. Formalize the claim that first-order identity is identical to the appropriate notion of Leibniz equivalence in the language of higher-order logic, remembering to place any type subscripts on expressions that require them.*

As Black (1952) points out, Leibniz equivalence should *not* be confused with the relation of sharing all *qualitative* properties. A qualitative property, informally, is a property that makes no reference to particular individuals. The property of being identical to a is a paradigm example of a non-qualitative property, because it involves the individual a, and so the above argument cannot be extended to show that qualitative indiscernible are identical. The thesis that qualitatively indiscernible objects are identical may have been the thesis Leibniz subscribed to, but that is a highly contentious metaphysical posit whereas our claim that having *all* properties in common suffices for identity is a logical truth.[1]

4.2 Quantifiers and variable binding

We will be making use of higher-order quantifiers quite a bit in this book. So far we have encountered two approaches to quantification: in Chapter 1 we treated them as constants of type $(\sigma \to t) \to t$, but we have also mentioned the treatment of quantifiers found in traditional approaches to first-order logic where they are treated as primitive variable binders, like λ. In these approaches, the quantifier symbols $\forall$ and $\exists$ are 'syncategorematic' symbols: this just means they are introduced through the term-formation rules—much like the

λ-symbol—and are not expressions in their own right which can be assigned a denotation on their own. It is worth pausing for a moment to relate these different treatments.

In first-order logic the first-order quantifiers, $\forall$ and $\exists$, are *variable binders* like λ is in a λ-language. Given a formula, A, possibly containing the variable x free, one can form new formulas $\forall xA$ and $\exists xA$, in which x doesn't appear free. By contrast we have regimented first-order quantificational expressions with the constants $\forall_e$ and $\exists_e$ of type $(e \to t) \to t$. These combine with *predicates* of type $e \to t$ (as opposed to formulas, which have type t) to form formulas. Moreover, they are not variable binders since the universal quantifier combines with its argument by application: any variable free in the term $F : e \to t$ will thus also be free in $\forall_e F$ since $FV(\forall_e F) = FV(\forall_e) \cup FV(F) = FV(F)$ (see Definition 3.2).

Comprehension Check 4.2. *In this question $F, G : e \to t$, and we are adopting infix notation for the logical connectives.*

1. *Translate the following first-order sentence into a sentence of $\mathcal{L}(\Lambda)$: $\forall x.(Fx \to Gx)$.*
2. *Translate the following sentences of $\mathcal{L}(\Lambda)$ into first-order logic: $\forall_e F$, $\forall_e \lambda x.(Fx \wedge Gx)$.*

We can introduce syncategorematic quantifiers directly into a higher-order language using a set of recursive formation rules for generating terms. For clarity, we will use the symbols Π and Σ in this section for the variable binding devices instead of $\forall$ and $\exists$ to distinguish them.

Definition 4.5 (Higher-order logic with variable-binding quantifiers)**.** *Suppose that Γ is a signature that contains all the logical constants of higher-order logic except for the higher-order quantifiers. We may define the terms of type σ, for each σ, simultaneously as follows:*

- *$c : \sigma$ whenever $c \in \Gamma^\sigma$.*
- *$x : \sigma$ whenever $x \in Var^\sigma$.*
- *If $M : \sigma \to \tau$ and $N : \sigma$ then $(MN) : \tau$.*
- *If $M : \tau$ and $x \in Var^\sigma$, $(\lambda x.M) : \sigma \to \tau$.*
- *If $M : t$ and $x \in Var^\sigma$, $(\Pi_\sigma x.M) : t$, $(\Sigma_\sigma x.M) : t$.*

Given the abstraction hypothesis from chapter 3, there's very little difference between these two approaches. Let's outline a general method for translating between the variable-binding treatment of the quantifiers and the present treatment. We can map terms of the language with variable binding quantifiers to the language of higher-order logic by treating the string of symbols $\forall_\sigma x$ in the variable-binding notation as simply being short for $\forall_e \lambda x$. More precisely, we can define a mapping from terms M of the variable-binding in signature Γ to terms M' in the present higher-order language with signature $\Gamma \cup \{\forall_\sigma, \exists_\sigma : (\sigma \to t) \to t \mid \sigma \text{ a type } \}$ as follows:

1. $M' = M$ when M is a constant or a variable
2. $(MN)' = M'N'$
3. $(\lambda x.M)' = \lambda x.M'$
4. $(\Pi_\sigma x.M)' = \forall_\sigma \lambda x.M'$
5. $(\Sigma_\sigma x.M)' = \exists_\sigma \lambda x.M'$

Conversely, we can construct a translation taking a term M of our language to a term M^* of the variable binding language simply by finding something to map the constants $\forall_\sigma, \exists_\sigma : (\sigma \to t) \to t$ to. We can create such terms easily by abstracting into the position of the predicate F in sentences like $\Sigma_\sigma y.Fy$:

- $(\forall_\sigma)^* = \lambda X.\Pi_\sigma y.Xy$
- $(\exists_\sigma)^* = \lambda X.\Sigma_\sigma y.Xy$

here $y : \sigma$ and $X : \sigma \to t$. The clauses for the remaining constants, the variables, application and λ-abstraction are identical to the Clauses 1–3 above—e.g. $(MN)^* = M^*N^*$ (and there is nothing corresponding to the last Clauses, 4 and 5, since our language has no such term formation rule).

Remark 4.2. I have described these mappings between the languages as 'translations'. Whether these mappings really preserve meaning, however, depends on the interpretation of the two languages. It is especially obscure what the 'meaning' of the quantificational expressions is on the syncategorematic approach, since the quantifier symbols are not stand-alone syntactic items, and do not receive interpretations on their own. A better way to think about these mappings is simply as a recipe for understanding one of these languages for someone who already understands the other—as a way of coordinating an interpretation of one language with the other.

Comprehension Check 4.3. *The reader should verify for themselves that the terms* $(\forall_\sigma)^*$ *and* $(\exists_\sigma)^*$ *really do have the type* $(\sigma \to t) \to t$.

The quantifiers are not the only sorts of things that can be given a syncategorematic treatment. It is possible to treat the truth-functional connectives syncategorematically. One could do this by adding to the term formation clauses in Definition 4.5 clauses like the following:

When $A : t$ and $B : t$ then $(A \wedge B) : t$
When $A : t$, $\neg A : t$

In our formalism, there is a stand-alone term $\neg : t \to t$ and the expression $\neg A$ is formed by combining it with A by application. By contrast, according to the syncategorematic theory of negated propositions, we do not need a stand-alone operation of negation to form negated propositions. This treatment of the logical words was preferred by the logical atomists—including Wittgenstein, Russell and Ramsey—because it was compatible with a more modest higher-order ontology. These logical atomists did not believe that there were higher-order entities corresponding to logical words, like quantification, conjunction and negation. The symbol $\wedge$, for instance, simply indicates the way in which you should combine two propositions, just as putting terms together in parenthesis, (MN), indicates the way in which we should combine a predicate and an argument (by *application*) and is not itself a higher-order entity.

We saw above, however, that given the abstraction hypothesis a syncategorematic treatment of the quantifiers doesn't really avoid positing higher-order properties corresponding to the quantifiers, since they can be defined from sentences involving the syncategorematic quantifiers by abstraction.

Comprehension Check 4.4. *Consider the language described above, obtained by adding conjunction syncategorematically to the language in Definition 4.5. Show that we can obtain a conjunction connective term of type $t \to t \to t$ by abstracting into sentence position twice.*

In part III of this book, we will weaken the abstraction hypothesis, and we will be able to formulate the sort of view that the logical atomists were interested in.

Exercise 4.3. *Observe that these translations are not mutual inverses of each other. For instance $\forall_e F$ translates to $\lambda X.(\Pi_e y.Xy)F$, and this term translates back to $\lambda X.(\forall_e \lambda y.Xy)F$. Use the principles of η-synonymy and β-synonymy to justify the claim that $\forall_e F$ is nonetheless synonymous with $\lambda X.(\forall_e \lambda y.Xy)F$. Explain briefly what further assumptions about synonymy would be needed to establish that that $(M^*)'$ is synonymous with M for any term M of our language.*

Exercise 4.4. *Formalize the following English sentence in standard higher-order logic using λ and the quantifier constants $\forall_e$ and $\exists_e$.*

a. Someone loves himself.

b. Everybody is loved by somebody.

c. Everyone loves someone who loves them back.

d. Something is such that snow is white.

Endnote

1. See Black (1952) for countexamples to the identity of qualitative indiscernables.

5

Higher-order logics

A higher-order language is a language containing the signature of higher-order logic Λ. A higher-order logic is a set of sentences containing certain logical axioms and closed under certain logical rules.

5.1 Higher-order logics

From here on out we shall use Σ to denote a signature that is disjoint from Λ: we will call elements of Λ logical constants and elements of Σ non-logical constants. A higher-order logic is a set of formulae—i.e. a subset of $\mathcal{L}^t(\Sigma \cup \Lambda)$—that contains certain logical axioms and is closed under certain logical rules. These axioms and rules are designed to be as neutral as possible on controversial questions about the granularity of reality (that is, how fine-grained propositions, properties and relations are). The axioms and rules can be divided into roughly three camps: (i) the axioms and rules of propositional logic, (ii) the axioms and rules of first-order logic, and their obvious analogues for the higher-order quantifiers, (iii) an axiom governing the behaviour of expressions defined by abstraction based on our previous discussions of β and η.

We will begin our study of higher-order logics with the pared-down signature Λ^- discussed in Section 4.1 that only contains the material conditional $\rightarrow$: $t \rightarrow t \rightarrow t$ and for each type σ a higher-order quantifier $\forall_\sigma : (\sigma \rightarrow t) \rightarrow t$. Let's begin with the first kind of axiom and rule. Recall that in our pared down signature $\bot$ may be defined as $\forall_t p.p$, and $\neg$ as $\lambda p.(p \rightarrow \bot)$. We suggested that $\bot$ so defined will be false, so that $\neg A$ will have the opposite truth value of A. So every instance of the following schemas ought to be true since they correspond to tautologies of propositional logic:

PC1 $A \rightarrow B \rightarrow A$

PC2 $(A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$

PC3 $(\neg A \rightarrow \neg B) \rightarrow B \rightarrow A$

PC1 and PC2 are easy to remember as they correspond to the types of the K and S combinators from Chapter 3. Moreover, we have modus ponens as a rule of inference:

MP From A and $A \rightarrow B$ infer B.

It is a well-known fact that any propositional tautology may be derived from PC1-PC3 using only modus ponens.[1]

The next axiom and rule generalize an axiom and a rule from first-order logic to the higher-order quantifiers.

UI $\forall_\sigma F \rightarrow Fa$

DOI: 10.4324/9781003039181-8

Gen From $A \to B$ infer $A \to \forall_\sigma x B$ provided $x \notin FV(A)$

These axioms are schematic along two dimensions. First, there is a version of UI and Gen for every choice of type σ. And for a fixed type σ there is an instance of the axiom/rule for any terms F, a, A, B of the appropriate type, namely $F : \sigma \to t$, $a : \sigma$ and $A, B : t$, and variable $x : \sigma$.

The combination of PC1-PC3 with the obvious first-order analogues of UI and Gen form a complete axiomatization of first-order logic.

Remark 5.1 (The relation to first-order logic)**.** UI stands for universal instantiation and corresponds to the first-order axiom $\forall x.A \to A[a/x]$ where a is a term that can be substituted for x without any free variables in a getting bound by quantifiers appearing in A. Observe that our version of UI can be rewritten in this form (given β-synonymy and our conventions about suppressing λs appearing immediately after quantifiers) by letting F be $\lambda x.A$.

Our rule Gen is due to Frege, although the first-order version of it is sometimes called 'Hilbert-Ackerman Generalization'.[2] First-order logic is sometimes formulated with a simpler rule of generalization: from A infer $\forall x A$, in which case you need to add the further axioms $\forall x(A \to B) \to \forall x A \to \forall x B$ and $A \to \forall x A$ when x is not free in A. A similar reformulation of our axioms for higher-order quantification along these lines is also possible.

The final axiom governs λ and ensures that $\beta\eta$-equivalent formulae are materially equivalent:

β $A \to B$ provided A and B are immediately β-equivalent terms of type t.

η $A \to B$ provided A and B are immediately η-equivalent terms of type t.

The justification of these axioms is straightforward: given β-synonymy A and B should be synonymous when they are immediately β or η-equivalent, and thus should be materially equivalent.

Before we define the key notion of a higher-order logic, let us first introduce the notion of a higher-order *theory*. While logic concerns only the most general laws of truth—those stated in terms of logical words—a theory can concern any domain of inquiry. A physical theory might involve truths involving predicates like 'electron' and 'space-time point', a mathematical theory might involve predicates like 'is a number' and 'is a member of', and so on. Crucially, we can apply the logical axioms and rules to any collection of truths to get other truths. A theory is thus any set of sentences, of any signature, that contains the logical axioms and is closed under the logical rules.

Definition 5.1 (Higher-order theory)**.** *A higher-order theory, T, is a subset of* $\mathcal{L}^t(\Sigma \cup \Lambda^-)$ *subject to the following constraints:*

Axioms *Every instance of PC1, PC2, PC3, UI,* β *and* η *belongs to T.*

Modus Ponens *If* $A, A \to B \in T$ *then* $B \in T$.

Gen *If* $A \to B \in T$ *and* $x \notin FV(A)$, $A \to \forall_\sigma x.B \in T$.

Logic concerns only the most general sorts of truths. While one can certainly have theories that include truths about, say, Socrates' life, these are not logical theories. A logic, by contrast, may include truths involving the name 'Socrates'—e.g. 'if Socrates is wise then Socrates is wise'—but if it does so it will be the general sort of truth that would apply equally to any name. Similarly, if a logic makes a claim involving the predicate 'is an electron' it is the

sort of claim it will make about any other predicate: logics are not subject-specific. Logics are therefore theories closed under the rule of uniform substitution of non-logical constants.

Definition 5.2 (Higher-order logic)**.** *A higher-order logic* $\mathbf{L}$ *in a signature of non-logical constants* Σ *is a theory in that signature that is closed under the rule of substitution*

Rule of Substitution *If* $A \in \mathbf{L}$ *then* $A' \in \mathbf{L}$ *where* A' *is the result of uniformly replacing any list of non-logical constants or free variables in* A *with terms of the same type as the constant or variable it is replacing, provided the replacing terms do not contain any free variables that will get bound upon replacement.*

A higher-order theory or logic can involve open formulas. When a theory asserts an open formula (e.g. $p \rightarrow p$), it should be informally understood to be asserting that the formula is universally satisfied ($\forall_t p(p \rightarrow p)$).

We can immediately infer the existence of a logic:

Example 5.1 (The degenerate logic)**.** *For any signature* Σ*,* $\mathcal{L}^t(\Sigma \cup \Lambda^-)$ *is a higher-order logic (and a higher-order theory), for it trivially contains the logical axioms and is closed under the logical rules and the rule of substitution.*

The claim that there exists a non-degenerate higher-order logic—a *proper* subset of the set of formulas that contains the axioms and is closed under the rules—is a substantive claim, but one that can be substantiated through models (see Chapter 15).

As we have defined it, a higher-order logic is signature dependent. The following exercise shows we can identify logics across logical signatures by checking that they contain the same sentences in the purely logical fragment of the language, $\mathcal{L}(\Lambda^-)$.

Exercise 5.1.

a. Show that if two higher-order logics, $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *in signatures* $\Sigma_1 \cup \Lambda^-$ *and* $\Sigma_2 \cup \Lambda^-$*, agree about the purely logical sentences (so that* $\mathbf{L}_1 \cap \mathcal{L}^t(\Lambda^-) = \mathbf{L}_2 \cap \mathcal{L}^t(\Lambda^-)$*), then* $\mathbf{L}_1 \cap \mathcal{L}^t(\Sigma_2 \cup \Lambda^-) = \mathbf{L}_2 \cap \mathcal{L}^t(\Sigma_1 \cup \Lambda^-)$.

b. Show that any logic $\mathbf{L}$ *in the purely logical signature* Λ^- *extends to a unique logic* $\mathbf{L}^\Sigma$ *in a signature* $\Sigma \supseteq \Lambda^-$ *such* $\mathbf{L}^\Sigma \cap \mathcal{L}^t(\Lambda^-) = \mathbf{L}$.

We will begin by examining the smallest higher-order logic, which can be obtained by starting with the axioms and contains only sentences that can be obtained from those axioms by repeatedly applying the rules. We call this logic H.

Definition 5.3 (H)**.** *We will write* H *to denote the smallest higher-order logic.*

The system H has a good claim to being the analogue of 'classical logic' for higher-order languages. It is characterized by the rules and axioms of classical propositional logic, and the classical quantifier rules and axioms of predicate logic (extended to the higher-order quantifiers). Higher-order languages contain the λ device which does not appear in propositional or predicate logic, and so H has a pair of axioms governing it which we do not find in propositional or predicate logic.

Not all of these axioms and rules are entirely uncontested however, and there are a couple of salient ways in which people have been tempted to go 'non-classical' in the literature. One class of concerns involve the β axiom schema. A wide class of positions on propositional granularity deny the existence of a unary property that yields the proposition that *Socrates is wise* when applied to *Plato*, since applying a unary property to *Plato* should result in a proposition that 'contains', or 'is about' *Plato*. Yet given β we can infer the identity of

the propositions *Fa* and $(\lambda x.Fa)b$. The position of the present book (to be developed in part III) is this: to capture these more structured views of reality we do not reject β, rather we simply remove the offending λ-terms from the language. The term $\lambda x.Fa$, if it denotes anything, denotes a property that yields the proposition *Fa* when applied to an individual *b*. If someone thought that there is no property that yields *Fa* when applied to *b*, then it would not be perspicuous for them to employ the λ-term $\lambda x.Fa$ in their theorizing, for it would not be clear what it referred to.[3] We show how to reformulate H in different λ-languages in part III (Section 9.5) while keeping all instances of β that are well-formed in the language. Some philosophers have instead allowed more structured views of reality to make use of the problematic λ-terms but weakened β so that it no longer applies to the problematic terms—a prominent system like this is H_0, which we will describe and discuss further in Section 11.1.[4] Instead of requiring *Fa* and $(\lambda x.Fa)$ to be identical it merely requires them to be materially equivalent. However, we have not adopted this approach in the present book: the rationale is that we don't really have much of an independent grip on the λ-device outside of our stipulations about how it behaves. Given β and η the meanings of λ-terms are pinned down uniquely in a way that's analogous to the way that the classical rules for the connectives and quantifiers pin down those things (recall Proposition 3, and the analogous results of Harris (1982) for identity and the quantifiers discussed in the introduction). By contrast the weaker system H_0 only pins down the meanings of λ-terms up to their extension: it tells us that $\lambda x.Fa$ refers to a property that applies to everything if Socrates is wise, and applies to nothing if he isn't, but tells us no more than this. We discuss this further at the end of Section 11.1.

The other class of concerns involve the axiom of universal instantiation, UI. (UI is essentially the universal elimination principle stated in the introduction, which we similarly took to be integral to our understanding the $\forall$.) Two different reasons for weakening it arise in connection with puzzles we encounter in chapters on modality and structure (specifically Chapters 8 and 11). The first has to do with the derivability of certain surprising theses in quantified modal logic, such as the claim that everything necessarily exists. The latter has to do with the Russell-Myhill paradox. A time-honoured response to this paradox maintains that all higher-order quantifiers are restricted so that the logic of quantificational logic must be the logic of restricted quantification which, of course, does not include the principle of universal instantiation. There is consequently a weakening of H, Free H, which is obtained by replacing universal instantiation in H with Free UI below, and adds to the remaining axioms and rules of H the axioms Quantifier Exchange and Quantifier Normality (both of these are already theorems of the stronger system H):

Free Instantiation $\forall_\sigma x(\forall_\sigma F \to Fx)$

Quantifier Exchange $\forall_\sigma x \forall_\tau y.A \to \forall_\tau y \forall_\sigma x.A$

Quantifier Normality $\forall_\sigma x(A \to B) \to \forall_\sigma xA \to \forall_\sigma xB$

The notion of a free higher-order logic can be defined in a way entirely analogously to the way we have defined a (classical) higher-order logic above, and we may refer to Free H as the smallest free higher-order logic. We henceforth set aside these possible weakenings and continue our investigation of higher-order logics that extend H.

In Chapter 4 we suggested that just as disjunction can be characterized, up to extension, in terms of conjunction and negation, identity could be characterized up to extension by the relation of sharing all the same properties, which we called Leibniz equivalence:

$\lambda xy.\forall_{\sigma\to t}Z(Zx \leftrightarrow Zy)$. Indeed, there is a variant definition of Leibniz equivalence, $\lambda xy.\forall_{\sigma\to t}Z(Zx \to Zy)$, which can be shown to be equivalent to the original:

Exercise 5.2. *Show that* $\forall_{\sigma\to t}Z(Zx \to Zy) \to \forall_{\sigma\to t}Z(Zx \leftrightarrow Zy)$.[i]

From now on, we will adopt the above simplified formulation of Leibniz equivalence. Given the axioms of self-identity, $a =_\sigma a$, and Leibniz's law, $a =_\sigma b \to Fa \to Fb$, we can actually prove that identity is coextensive with Leibniz equivalence—see Exercise 5.7 below.

We are concerned with the study of higher-order logics, rather than some single system of 'higher-order logic', and our definition of a higher-order logic reflects this. Thus the study of higher-order logic is in this sense more analogous to the study of modal logic than first-order logic. There is only one system that people have in mind when they talk about 'first-order logic'—the study of 'first-order logics' would presumably be taken to mean the study of first-order logics that weakens classical logic in some way (such as intuitionistic predicate logic or free logic). By contrast, there are many modal logics that say interesting and different things about the behaviour of $\Box$.

Of course, there is a way of defining a (classical) first-order logic that is completely analogous to our definition of a higher-order logic: a set of first-order formulas closed under the first-order axioms and rules, and under a version of the rule of substitution.[5] However, the collection of first-order logics is not particularly interesting: For any finite list of finite cardinalities there is a first-order logic that extends first-order logic proper with the claim that the size of the universe is one of the listed cardinalities, for any (possibly infinite) list of finite cardinalities there is a first-order logic that extends first-order logic proper with the claim that the size of the universe is not one of the listed cardinalities, and these moreover exhaust all of the first-order logics.[6] So the first-order logics that strengthen plain first-order logic are pretty uninteresting and mostly ignored. The case of propositional logic is even more extreme: Post (1921) showed that there are no consistent propositional logics extending classical propositional logic.[7]

Because the logical language of higher-order logic is significantly richer than that of propositional and first-order logic, there are many pairwise inconsistent higher-order logics containing many mathematically and philosophically contentious claims. For instance, one can formulate higher-order analogues of contentious set-theoretic principles like the axiom of choice and the continuum hypothesis. And there are other higher-order logics that contain the negations of these principles. These questions are part of logic, not mathematics, in the sense that they can be formulated using only logical primitives (higher-order quantifiers $\forall_\sigma$ and the material conditional $\to$) and do not use mathematical primitives like set membership. Whether these logical principles stand or fall with the corresponding set-theoretic principles is itself open to debate, but the fact that the set-theoretic versions of these principles have a contested status suggests the higher-order versions should also be treated as substantive postulates. One can also formulate interesting philosophical claims: for instance the claim $\forall_t pq.(p \wedge q) =_t (q \wedge p)$ is stated purely using the logical vocabulary, and there are higher-order logics that contain this sentence, and others which contain its negation. But this logical principle is closely related to a contentious debate concerning the metaphysics of propositions: some structured proposition theorists reject analogous principles concerning propositions, while possible world theorists accept it. Again, our principle is a sentence of pure logic and does not mention the word 'proposition' anywhere. But still: the fact that the claim about propositions is hotly contested suggests there could also be an analogous debate at the level of logic concerning the principle $\forall_t pq.(p \wedge q) =_t (q \wedge p)$, and so we should proceed with caution.

Any sentence in the smallest higher-order logic, H, will belong to any higher-order logic so we will begin by examining H. Membership of a sentence in H can be determined by finding a *proof* of the sentence.

Definition 5.4 (Proof)**.** *A proof is a sequence of formulae in* $\mathcal{L}(\Sigma \cup \Lambda)$ *subject to the constraint that for any element A of the sequence either:*

(i) A is an instance of one of the axioms.

(ii) There is a formula C such that both C and $C \to A$ *appear earlier in the proof.*

(iii) $A = C \to \forall xB$ *where C is a formula such that* $x \notin FV(C)$ *and* $C \to B$ *appears earlier in the proof.*

A proof from a set of assumptions Γ *is a proof in the above sense, except that a line may also include any member of* Γ.

Observe that we don't have any clause for the rule of substitution. This is a special property the way our axioms and rules for H are formulated: if $A_1, A_2, \ldots, A_n$ is a proof, then we can construct another proof $A'_1, A'_2, \ldots, A'_n$, by letting $A'_1 \ldots A'_n$ be the result of uniformly replacing any constants or variables appearing in $A_1 \ldots A_n$ with terms $B_1 \ldots B_k$ of matching type. This will meet the conditions for being a proof provided the free variables in $B_1 \ldots B_k$ don't get bound when this substitution is performed. (To get around cases where this proviso doesn't hold, a slight fiddle is required; see endnote.[8]) Thus if there is a proof of some sentence A there exists a proof of any uniform substitution instance A'.

Comprehension Check 5.1. *Show that the smallest higher-order* theory *is identical to* H *(the smallest higher-order logic).*

A proof may be notated by giving a numbered list of sentences, with a note to the side indicating whether the line is an instance of an axiom (case (i)), an instance of modus ponens (case (ii)) or an instance of Gen (case (iii)). In practice, however, many steps can be omitted: this is not a book about propositional logic so any step that can be justified by propositional reasoning can simply stated as such. For example, here is a proof of the claim $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to \forall_\sigma x.B$ that omits many propositional steps

1. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to \forall_\sigma x(A \to B)$ (by propositional logic)
2. $\forall_\sigma x(A \to B) \to \lambda x.(A \to B)x$ (an instance of UI)
3. $\lambda x.(A \to B)x \to A \to B$ (an instance of β)
4. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to A \to B$ (1–3 by propositional logic: transitivity of $\to$)
5. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to \forall_\sigma xA$ (by propositional logic)
6. $\forall_\sigma xA \to (\lambda x.A)x$ (an instance of UI)
7. $(\lambda x.A)x \to A$ (an instance of β)
8. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to A$ (from 5–7 by propositional logic: transitivity of $\to$)
9. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to (A \wedge (A \to B))$ (4 and 8 by propositional logic)
10. $(A \wedge (A \to B)) \to B$ (by propositional logic)
11. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to B$ (9 and 10 by transitivity of $\to$)
12. $(\forall_\sigma x(A \to B) \wedge \forall_\sigma x.A) \to \forall xB$ (by Gen)

But even with many steps of propositional logic omitted this proof is pretty tedious. In fact, the thing we are trying to prove is a rather elementary fact about quantification which has an analogue in first-order logic. Thus, in this book, steps that are justified by routine quantificational reasoning—by which I mean steps whose first-order analogues are derivable in first-order logic—can also be omitted.

Observe that in this proof our notational convention of suppressing λ immediately after a quantifier makes it easy to miss potential instances of UI. For instance step 6 of this proof, if written out explicitly would be: $\forall_\sigma \lambda x.A \to (\lambda x.A)x$. This is an instance of UI where F is the just $\lambda x.A$, although this is not immediately apparent when the conventions are in place.

Lastly, observe that our axioms allow us to replace $\beta\eta$-equivalent terms in theorems to get theorems. For if A and B are $\beta\eta$-equivalent one can find a chain of formulas $C_1, \ldots, C_n$ such that any adjacent pair will be immediately β or immediately η-equivalent and such that $A = C_1$ and $B = C_n$. Thus $C_i \to C_{i+1}$ will be a an instance of axiom β or η for $i = 1, \ldots, n-1$. So by the transitivity of $\to$ we may infer that $A \to B$. Thus we may freely appeal to the following rule:

$A \to B$ whenever A and B are $\beta\eta$-equivalent.

So to save time in our numbered proofs, one can also write A if a $\beta\eta$-equivalent of A appears earlier in the proof.

Sometimes it is also convenient for the reader to add extra lines in a proof indicating a definition is being invoked. For instance, in the pared-down signature we might wish to have a proof of Leibniz's law: $a =_\sigma b \to Ga \to Gb$. This can be notated as follows:

1. $\forall_{\sigma\to t} Z(Za \to Zb) \to \lambda Z.(Za \to Zb)F$ (an instance of UI).
2. $\lambda Z.(Za \to Zb)F \to (Fa \to Fb)$ (an instance of β)
3. $\forall_{\sigma\to t} Z(Za \to Zb) \to (Fa \to Fb)$ (transitivity of $\to$)
4. $(\lambda xy.\forall_{\sigma\to t} Z(Zx \to Zy))ab \to (Fa \to Fb)$ (the previous line is $\beta\eta$-equivalent to this)
5. $a =_\sigma b \to (Fa \to Fb)$ by definition

In the following exercises work in the pared-down signature Λ^-, and treat other symbols such as $=_\sigma$ as abbreviations.

Exercise 5.3. *Suppose that A and B are $\beta\eta$-equivalent terms of type t. Explain how one can construct a proof of $A \to B$.*

Exercise 5.4. *Show that if there is a proof of A there is also a proof of $\forall xA$.*

Exercise 5.5. *Using the definitions of $=_\sigma$ and $\leftrightarrow$, prove $a =_\sigma a$.*

Exercise 5.6. *Using the β axiom prove $(\lambda x.M)a =_\tau M[a/x]$.*[ii] *Using η similarly prove $\lambda x.Mx =_{\sigma\to\tau} M$.*

Finally, observe that a great number of metatheorems from first-order logic carry over to higher-order logics. For instance, if there is a proof of B from a set of of assumptions $\Gamma \cup \{A\}$, then there also exists a proof of $A \to B$ from assumptions Γ. (This is shown in usual fashion, by showing that prefixing $A \to$ to each element of a proof of B from $\Gamma \cup \{A\}$ yields a sequence of sentences that can easily be seen to be provable from Γ.) Similarly if you can prove C from $\Gamma \cup \{A\}$ and also from $\Gamma \cup \{B\}$ then you can prove C from $\Gamma \cup \{A \vee B\}$.

In light of facts like these, one can use informal mathematical reasoning—such as conditional proof, proof by cases and so on—to show things are theorems of H while being safe in the knowledge that such informal reasoning can be turned into a formal proof if necessary. Thus, for instance, if you want to show a conditional $A \to B$, it is sufficient to proceed in the usual fashion by starting the argument 'Suppose A', and then proceeding to establish B using axioms and rules of H, or previously established lemmas. The situation here is similar to the situation with respect to set-theory. Officially ZFC is a first-order theory, usually presented as a list of axioms with only two rules of inference corresponding to Modus Ponens and Gen. But no practicing set-theorist actually reasons in this formal system: set-theoretic arguments are presented in mathematical English and freely use conditional proof, reasoning by cases, and so on, and are no less acceptable as a result.

To illustrate informal reasoning in higher-order logic, let us give a simple proof that there are two propositions: $\exists_t pq(p \neq_t q)$.

Example 5.2. *It suffices to find two propositions and show that they are not the same. Consider the proposition* $\bot$ *(i.e.* $\forall_t p.p$*) and* $\top$ *(which is, modulo definitions,* $\bot \to \bot$*). By Leibniz's law we have that if* $\top =_t \bot$ *then if* $(\lambda p.p)\top$ *then* $(\lambda p.p)\bot$*. By* β *we know that if* $\top =_t \bot$ *and* $\top$ *then* $\bot$*. But of course, propositional logic tells us that* $\top$ *and not* $\bot$*, and we can infer* $\top \neq_t \bot$*. Finally, by quantificational reasoning (existential generalization)* $\exists_t pq.p \neq_t q$*.*

5.2 Higher-order logics in other logical signatures

As we noted previously, the remaining truth-functional connectives cannot be defined from $\to$ and $\bot$ without special assumptions about the granularity of propositions, so there may be philosophical reasons to want to introduce further logical connectives as primitives. If we are taking $\wedge$ and $\vee$ as further primitives, we need to include further logical axioms governing them, listed below.[9] These axioms will entail, for instance, that $(A \vee B)$ and $\neg(\neg A \wedge \neg B)$ are materially equivalent, but will be neutral about the more contentious identity of $\vee$ with $\lambda pq.\neg(\neg p \wedge \neg q)$.

Conjunction Axioms

$A \to B \to (A \wedge B), (A \wedge B) \to A, (A \wedge B) \to B$

Disjunction Axioms

$(A \to C) \to (B \to C) \to ((A \vee B) \to C), A \to (A \vee B), B \to (A \vee B)$

Negation Axioms

$(A \to B) \to (A \to \neg B) \to \neg A, A \to (\neg A \to B)$

Biconditional Axioms

$(A \leftrightarrow B) \to (A \to B), (A \leftrightarrow B) \to (B \to A), (A \to B) \to (B \to A) \to (A \leftrightarrow B)$

Top and Bottom Axioms

$\bot \to A, A \to \top$

Identity Axioms

$a =_\sigma a, a =_\sigma b \to A \to A[a/b]$ provided a and b are free for all variables in A.

Exercise 5.7. *Prove* $a =_\sigma b \leftrightarrow \forall_{\sigma \to t} Z(Za \to Zb)$ *using the axioms for identity and the other logical principles.*[iii]

5.3 Inductive definitions in higher-order logic

With some higher-order logic under our belt, a number of novel logical techniques open up that were not available to us in a first-order setting. We will explore many of these in subsequent chapters, however now is a good time to highlight one particular advantage that higher-order resources afford us: we are now in a position to offer inductive definitions and reason about them using a principle of induction.

Consider the relations *x is a parent of y* and *x is an ancestor of y*. The ancestors of *y* includes *y*s parents, the parents of *y*s parents, the parents of the parents of *y*s parents, and so on. Thus it seems like one ought to be able to define ancestorhood from parenthood. But it is a well-known fact that this is not possible in first-order logic.[10] The most flatfooted approach would be to try to capture the connection using a disjunction—*x* is an ancestor of *y* iff *x* is a parent of *y*, or a parent of a parent of *y*, or a parent of a parent of a parent of *y*, or . . . —but all disjunctions are finite in first-order logic, and this disjunction would have to go on forever to be adequate.

The trick for defining notions like *ancestor* from notions like *parent of* in higher-order logic traces back to the founder of higher-order logic, Gottlob Frege. The *ancestral* of a relation, informally, is the smallest transitive relation containing that relation. Because Frege identified coextensive relations, it was safe for him to talk about *the* ancestral of a relation, but it turns out his definitions work just as well when extensionalism isn't assumed. (Extensionalism, discussed in the next chapter, is not a theorem of H.) However, the idea behind Frege's definition is very general, and key for carrying out recursive definitions in higher-order logic.

Frege's trick is this. He begins by noting that *x* is an ancestor of *y* iff every transitive relation which holds between parents and their children holds between *x* and *y*. We can reason this through as follows. Suppose *S* is a transitive relation that holds between parents and their children. Since *S* holds between parents and their children, *S* must hold between *y* and *y*s parents, between *y*s parents and *their* parents, and so on. And because *S* is transitive, it must also hold between *y* and *y*'s parents' parents, between *y* and *y*'s parents' parents' parents, and so on. So *S* must hold between *y* and any of *y*s ancestors. In the other direction, if every transitive relation that holds between parents and their children holds between *x* and *y*, then *x* is an ancestor of *y*, simply because *ancestorhood* is a transitive relation that holds between parents and their children.

This leads to the more general definition of the *ancestral* of a relation. *x* bears the ancestral of *R* to *y* iff any relation *X* that (i) applies to *z* and *w* whenever *Rzw* and (ii) is transitive applies to *xy*. We will use R^* for the ancestral of *R*, and it may be defined as follows:

$$R^* := \lambda xy.\forall X((\forall_e zw(Rzw \rightarrow Xzw) \wedge \forall zwu(Xzw \wedge Xwu \rightarrow Xzu)) \rightarrow Xxy)$$

We may show that R^* is an extensionally smallest transitive relation containing *R* as follows. (This argument will also serve to illustrate informal reasoning in H.) Suppose that *S* is an $e \rightarrow e \rightarrow t$ relation containing *R* (by extension)—i.e. $\forall_e zw(Rzw \rightarrow Szw)$. Suppose also that *S* is transitive ($\forall_e zwu(Szw \wedge Swu \rightarrow Szu)$). It suffices to show that R^* is contained in *S*: $\forall_e xy(R^*xy \rightarrow Sxy)$. Suppose that R^*xy. This means any transitive relation containing *R* (by extension) applies to *x* and *y*. But *S* is such a relation, so *S* applies to *x* and *y*.[11]

Quite often we will want to talk about a 'collection' of entities that are defined by taking some initial entities and closing them under certain operations. Frege's trick is useful here as

well. Let's illustrate with an example. In the early study of modal logic, logicians would ask how many different modalities one could define from a given modality and negation: things like $\neg\Box$, $\neg\Box\neg$, $\Box\neg\Box\neg$, and so on.[12] While their approach was essentially metalinguistic—they would ask, up to logical equivalence, how many different operator *expressions* one could define from a given modality using negation—in higher-order logic we can formulate and investigate an analogous question of *metaphysics*. Informally, the *modalities* one can define from $\Box$ and $\neg$ are simply the operators you can obtain by composing finite strings of operators consisting of either $\Box$ or $\neg$. Thus the modalities are the smallest collection of operators containing $\Box$, $\neg$, and closed under composition. Using Frege's trick, X is a modality iff every property of operators applying to $\neg$ and $\Box$ that is closed under composition applies to X:

$$\text{Modality}_{\Box,\neg} := \lambda X.\forall_{(t\to t)\to t} W(W\Box \wedge W\neg \wedge \forall_{t\to t} YZ(WY \wedge WZ \to W(Y \circ Z)) \to WX)$$

Recall that $\circ$ is our infix notation for the B' combinator, $X \circ Y := \lambda p.X(Yp)$.

Remark 5.2 (Operator granularity)**.** Using the above we can formulate the question: how many modalities are definable from $\neg$ and $\Box$? However, to make *progress* on this question one would need to work in a higher-order logic that is stronger than H. For instance, one can obtain a higher-order logic by extending H with the following structure-like principle, telling us that different compositions of negations are different

$$\forall_{t\to t} XY(\neg \circ X =_{t\to t} \neg \circ Y \to X =_{t\to t} Y)$$

a similar principle for $\Box$ would imply that different strings of $\Box$s and $\neg$s have a different composition. On the other hand, more coarse-grained theories of operators will identify certain compositions. In Section 6.1, for instance, we will discuss a theory that implies identities like $\neg =_{t\to t} (\neg \circ \neg) \circ \neg$. So the answer to the question of how many modalities there are will depend on the background logic.

Things defined inductively, like our notion of a modality, come with their own principle of induction. If you want to show that all modalities have a given property, W, show that $\neg$ and $\Box$ have this property, and that this property is preserved by composition: if X and Y have it, so does $X \circ Y$. The reason this induction principle holds is practically built into to the definition of a modality. By definition, a modality is an operator that possesses all properties that apply to $\neg$ and $\Box$ and are closed under composition. Thus every modality must possess W.

Exercise 5.8. *Consider a higher-order signature containing a connective* $\leq$: $t \to t \to t$, *intuitively to be thought of expressing propositional entailment. Say that an operator X is monotonic iff, whenever* $p \leq q$, $Xp \leq Xq$, *and anti-monotonic iff whenever* $p \leq q$, $Xq \leq Xp$. *Thus, for instance,*

$$\text{Monotonic} := \lambda X.\forall_t pq(p \leq q \to Xp \leq Xq)$$

Assuming that $\Box$ *is monotonic, and* $\neg$ *is anti-monotonic, show that every modality is either monotonic or anti-monotonic.*

Exercise 5.9. *Assume that x has blue eyes, and that any one with blue eyes has blue-eyed children.* $Bx \wedge \forall yz(By \wedge Ryz \to Bz)$. *Prove that every ancestor of x has blue eyes:* $\forall y(R^*xy \to By)$.

Let's now consider another example, this time illustrating an inductively defined *relation* between modalities. Some modalities are 'reversals' of each other: for instance, the modalities $\neg\neg\Box\neg\Box$ and $\Box\neg\Box\neg\neg$ are constructed in the same way but in opposite orders. As noted in Remark 5.2, H is neutral about operator granularity and so neutral about whether, say, $\neg\neg\Box\neg\Box =_{t\to t} \neg\Box\neg\Box$. However, we can define the notion of reversal in a way that is neutral about operator granularity. We define it inductively: $\neg$ on its own is a reversal of itself, and similarly for $\Box$, and if X is a reversal of Y, then $\neg \circ X$ is a reversal of $Y \circ \neg$, and $\Box \circ X$ is a reversal of $Y \circ \Box$. Thus the reversal is the smallest relation that holds between $\neg$ and itself, holds between $\Box$ and itself, and holds between $\neg \circ X$ and $Y \circ \neg$ and between $\Box \circ X$ and $Y \circ \Box$ whenever it holds between X and Y:

$$\begin{aligned}\text{Reversal} := \lambda UV.\forall_{(t\to t)\to(t\to t)\to t} W(W\neg\neg \wedge W\Box\Box \\ \wedge \forall_{t\to t} XY(WXY \to W(\neg \circ X)(Y \circ \neg) \wedge W(\Box \circ X)(Y \circ \Box)) \to WUV)\end{aligned}$$

Exercise 5.10. *Prove in* H *that* $(\neg \circ (\neg \circ \Box))$ *is a reversal of* $((\Box \circ \neg) \circ \neg)$ *using the definition of reversal. That is, show that an arbitrary relation satisfying the stated conditions, must hold between these two operators.*

We'll end this discussion by returning to the notion of a Church numeral, explored in Exercise 3.9. A Church numeral, recall, is an operation of type $(\sigma \to \sigma) \to \sigma \to \sigma$ which takes an operation $X : \sigma \to \sigma$ and returns the $\sigma \to \sigma$ operation you get by iterating X (i.e. composing X with itself) a particular finite number of times.[13] Thus the third Church numeral maps X to $\lambda y.X(X(Xy))$. We identify the zeroth Church numeral with the operation that maps each $\sigma \to \sigma$ operation to the identity at type σ:

$$0 := \lambda X\lambda y.y$$

That is, it takes an operation X and an argument y and applies X zero times to y, yielding y. The successor operation takes a Church numeral, $n : (\sigma \to \sigma) \to (\sigma \to \sigma)$, and yields the operation that takes an operation X, iterates it n times X^n—which is obtained by simply applying n to X—and then applies X one more time (yielding $\lambda x.X((X^n)x)$).

$$\text{Suc} := \lambda n\lambda X\lambda x.X((nX)x)$$

So the finite Church numerals are the smallest collection of entities of type $(\sigma \to \sigma) \to (\sigma \to \sigma)$ containing 0 and closed under successor.

Exercise 5.11.

a. Construct an explicit definition of the finite Church numerals, Num_σ*, in higher-order logic.*

b. State and prove the principle of induction for Church numerals (in H*).*

In the final pair of (more tricky) exercises you will prove that one can define "functions" from the Church numerals to entities of another type τ by recursion. By a function, we will mean a binary relation that relates each Church numeral to a unique entity in some other type τ. To define a function by recursion on the Church numerals you must say what that function does to 0, and you must say what it does to the successor of a numeral in terms of what it does to that numeral.

Exercise 5.12. *Let* τ *be an arbitrary type,* $a : \tau$ *and* $G : \tau \to \tau$*. Let* γ *be short for* $(\sigma \to \sigma) \to \sigma \to \sigma$*, the type of the Church numerals at type* σ*. Using Frege's trick explicitly define the smallest relation* $R : \gamma \to \tau \to t$ *that relates 0 to a, and for any Church numeral n (as defined in the previous exercise), that is related to* $y : \tau$*,* Suc *n is also related to Gy.*

Say that $S : \gamma \to \tau \to t$ is a functional relation on the Church numerals iff Sxy implies that x is a Church numeral, and each Church numeral bears S to a unique entity of type τ. In order to show that the relation R we have defined in this exercise behaves properly we need to show that R is a functional relation on the Church numerals. We would also like to show that a definition by recursion yields a unique functional relation: no other functional relation on the Church numerals that relates 0 to a and relates $\mathrm{Suc}\, n$ to Gy whenever it relation n to y, is coextensive with R.

Unfortunately, we cannot prove that R is functional in H alone. This is because H cannot prove that the Church numerals, so defined, are different from one another. For instance, H is consistent with a broadly extensionalist way of individuating propositions and operators, in which there are 2 propositions, 4 operators (of type $t \to t$) and 128 operations of type $(t \to t) \to t \to t$. This means there are only finitely many Church numerals at type t, and so some Church numerals are identified. (In fact, one can check that the result of iterating a truth function once is the same as iterating it thrice, since only negation iterates non-trivially.) Nonetheless, we can add an axiom to H stating that no two Church numerals (at a given type) have identical successors, and that 0 is not the successor of any number. These principles suffice to rule out this sort of behaviour. Because this principle is stated in purely logical terms, the result of adding it to H is another higher-order logic.

The Axiom of Infinity$^{\sigma}$ $\forall_{n\sigma} XY(\mathrm{Num}\, X \wedge \mathrm{Num}\, Y \wedge \mathrm{Suc}\, X =_{n\sigma} \mathrm{Suc}\, Y \to X =_{n\sigma} Y) \wedge \forall_{n\sigma} X(\mathrm{Num}\, X \to 0 \neq_{n\sigma} \mathrm{Suc}\, X)$

here we write $n\sigma$ as short for the type $(\sigma \to \sigma) \to \sigma \to \sigma$, the 'Church numbers for type σ'.

Exercise 5.13.

a. Using the Axiom of Infinity, show that if $R(\mathrm{Suc}\, x)y$ then for some $z : \tau$ Rxz and $Gz =_{\tau} y$.[iv]

b. Using part (a) and the Axiom of Infinity, show that R is functional.

c. Using part (b) show that there is at most one functional relation up to extension defined by recursion. Prove that if S is a functional relation on the Church numerals such that $S0a$ and, for any Church numeral n, if Sny then $S(\mathrm{Suc}\, n)(Gy)$, then S is coextensive with R.

These theorems can be used to form a basis for arithmetic in the higher-order logic H plus the Axiom of Infinity. For instance, for a given Church numeral n, one can define a unique functional relation $\mathrm{Add}\, n : \gamma \to \gamma \to t$, of 'adding n', given by the smallest relation that relates 0 to n and relates $\mathrm{Suc}\, x$ to $\mathrm{Suc}\, y$ when it relates x to y. Multiplication by n, $\mathrm{Mult}\, n : \gamma \to \gamma \to t$, is then defined as the smallest relation that relates 0 to 0, and relates $\mathrm{Suc}\, x$ to $\mathrm{Add}\, ny$ when it relates x to y. Using induction we can prove the normal laws of addition and multiplication, such as associativity and commutativity.

Endnotes

1. See, e.g., Sider (2010).
2. In Frege (1893), Section 17 it is formulated in a slightly more general form: from $A_1 \to \cdots \to A_n \to B$ infer $A_1 \to \cdots \to A_n \to \forall_{\sigma} x.B$ provided $x \notin FV(A_1) \cup \cdots \cup FV(A_n)$. See also Hilbert and Ackermann (1928).
3. This does not mean that they must reject English locutions like 'is such that Socrates is wise', which are sometimes claimed to be the English correlates of such λ-terms. That theorist must either find a property which when applied to b has a weaker connection to the proposition Fa than identity (and refrain from calling this property $\lambda x.Fa$), or tell a more complicated story about the interface between syntax and semantics. Perhaps 'is such that Socrates is wise' doesn't semantically express a unary property, but a proposition, with a special syntactic rule for disregarding syntactic arguments. Similar things can be said about other English expressions that are

sometimes thought to bear some important relation to contentious λ-terms, such as 'loves oneself' and $\lambda x.Lxx$, 'is loved by' and $\lambda xy.Lyx$. See Williamson (1985) for thoughts along these lines in the context of converses.

4. Uses of the system H_0 in the philosophical literature include Bacon and Zeng (2022) and Dorr et al. (2021). A sequent calculus formulation of it called ITL was originally given in Muskens (2007).
5. This rule of substitution requires the theory to be closed under the substitution of constants with terms, and predicate constants with open formulas.
6. Recall that for any finite k one can construct a first-order formula using only connectives, identity and the quantifiers saying that there are exactly k things.
7. A propositional logic must be closed under the rule of substitution. If L properly contains classical propositional logic, it must contain some non-tautologous sentence A. But then there is a way of substituting the propositional letters in A with contradictions or tautologies to get a classically inconsistent sentence A' (one can define the substitution in terms of the valuation that refutes A). By the rule of substitution, A' must also belong to L, and since L extends classical propositional logic, L must be inconsistent.
8. Replace the variables in $B_1 \ldots B_k$ with fresh variables that don't appear in $A_1 \ldots A_n$, and use the earlier method to find a proof of the desired formula that is correct except for the free variables it contains. We can then obtain a proof of the desired claim as follows: given a proof of $C[x_1 \ldots x_m]$ with free variables $x_1 \ldots x_m$ we can always prove $\forall x_1 \ldots x_n C[x_1 \ldots x_m]$ using Gen, and then we can infer $C[y_1 \ldots y_m]$, involving the desired free variables $y_1 \ldots y_m$, by UI.
9. Another way to state the axioms of higher-order logic in the full signature is to add, for each putative 'definition' in Table 4.1 , a corresponding axiom that merely guarantees coextensiveness of the definiens and definiendum. However the resulting axiomatization is slightly less elegant than the one presented below.
10. This follows, essentially, from the compactness theorem.
11. We can make this this last bit of reasoning a little more explicit if we want. Given R^*xy we know by definition and β that: $\forall X((\forall_e zw(Rzw \rightarrow Xzw) \wedge \forall zwu(Xzw \wedge Xwu \rightarrow Xzu)) \rightarrow Xxy)$. By UI we can instantiate X for S giving $(\forall_e zw(Rzw \rightarrow Szw) \wedge \forall zwu(Szw \wedge Swu \rightarrow Szu)) \rightarrow Sxy)$. And finally by our assumption that S contains R and is transitive we can apply modus ponens to get Sxy.
12. For instance, Parry (1939) shows there are, up to logical equivalence, fourteen potentially different modalities definable from an S4 modality.
13. The idea of identifying natural numbers with these higher-order iterating operations traces back to the Tractatus. See Wittgenstein (1922) 6.02.

Hints for exercises

[i] **Hint:** Given the antecedent, one must show that for an arbitrary X, $Xy \rightarrow Xx$. Consider instantiating Z in the antecedent with $\lambda w.\neg(Xw)$ and using β and .

[ii] **Hint:** Start with the fact that we can prove $(\lambda x.M)a =_\tau (\lambda x.M)a$.

[iii] **Hint:** To establish the right-to-left direction appeal to UI instantiating Z with $\lambda z.(z =_\sigma x)$.

[iv] **Hint:** Suppose not for contradiction. Let $R^- := \lambda zw.(Rzw \wedge \neg(z = \mathrm{Suc}\, x \wedge w = y))$. Show that R^-0a and that R^-xy implies $R^-(\mathrm{Suc}\, x)(Gy)$ and use this to conclude $\neg R(\mathrm{Suc}\, x)y$ for a contradiction.

6

Application

Higher-order theories of granularity

Higher-order logic forms a strong framework in which questions about the structure of reality can be formulated and investigated using logical methods.

Following Frege, many systems of higher-order logic have an axiom (The Fregean Axiom, below) stating that propositions are individuated by material equivalence. This axiom effectively tells us there are only two propositions. However, since the intensional turn in the 1970s, philosophers have taken seriously the idea that there are intensional operators, including modal, counterfactual and tense operators, that can distinguish between materially equivalent propositions. Whereas previous philosophers took expressions like these to be covertly making metalinguistic claims about the status of some bit of language, philosophers such as Arthur Prior and Saul Kripke have popularized the idea that these expressions correspond to genuine intensional operators that talk directly about reality, and so the Fregean Axiom is not appropriate.

While the operators discussed above draw distinctions between materially equivalent propositions, they do not seem to draw distinctions between logically equivalent propositions. We thus begin by precisifying the thesis that 'logically equivalent' propositions are the same. First, we focus on the truth-functional connectives—$\wedge, \vee, \neg, \rightarrow, \top, \bot$ etc.—and formulate a theory that states, roughly, that equivalence in propositional logic suffices for identity: that any two combinations of propositions in these connectives that correspond to a provable equivalence in propositional logic is an identity. Since this corresponds to the claim that the propositions form a Boolean algebra under the truth-functional connectives (albeit stated with higher-order rather than first-order quantifiers) we call the smallest higher-order logic containing these identities *Propositional Booleanism*. After this we will extend this idea to the remaining logical operations $\forall_\sigma, \exists_\sigma$ and $=_\sigma$ for each σ, by stating axioms that correspond, roughly, to the idea that provable equivalence using quantificational principles suffices for identity. Since this corresponds to the idea that provable equivalence in classical logic suffices for identity, we call this theory *Propositional Classicism*. More recently, there has been a hyperintensional turn in metaphysics. As with the intensional turn, the hyperintensional metaphysicians posit the existence of operators, such as the notion of propositional ground, propositional aboutness and propositional constituenthood, that draw distinctions between logically equivalent propositions. If these philosophers are correct then Booleanism and Classicism need to be weakened, and we consider ways in which to weaken them so as to be consistent with various hyperintensional connectives.

DOI: 10.4324/9781003039181-9

Everything we have said about propositions extends to properties and relations. For each theory of propositional individuation, there is a corresponding theory of individuation for properties and relations. We show how to generalize theories of propositional individuation to arbitrary relational types in Section 6.3. For properties and relations, we also consider further principles that draw on analogies between properties and relations and functions: that they are individuated by their applicative behaviour (see the principle Functionality below) and that they are plenitudinous (Functional Plenitude below). While theorems of earlier systems of higher-order logic, these principles can similarly be seen to be contentious from the perspective of intensional and hyperintensional metaphysics respectively.

The principles of higher-order logic discussed in this chapter are loosely related to a well-known framework for modelling properties and propositions, widely appealed to in philosophy and semantics: the *possible worlds framework*. It consists of two theses that can be maintained individually, but are often lumped together. The first is a thesis about propositions, namely that they are sets of possible worlds. The second is about properties and relations: that they are functions with propositions in their range. Properties are functions from individuals to propositions, relations are functions from pairs of individuals to propositions, operators are functions from propositions to propositions, and so on.

Both of these claims associated with the possible worlds formalism concern certain kinds of abstract objects: propositions, properties and relations. However, as we discussed in Sections 0.5 and 0.6, many debates about propositions, properties and relations have higher-order analogues that are not formulated by quantifying over any particular sort of abstract object, but by quantifying into sentence position, predicate position, operator position, and so on.

Not all aspects of these debates about propositions and properties have analogues in the higher-order case: there is usually a reductive element to the standard possible worlds picture—certain sorts of abstract objects (propositions, properties, relations, etc.) are being reduced to another sort of mathematical object (sets or functions). We will see that this part of the debate gets lost in the translation into higher-order language. But there are other aspects of the debate that are preserved, such the question of whether the conjunction of p and q is the same as the conjunction of q and p.

Remark 6.1. A common complaint against the possible worlds formalism is that there appears to be an embarrassment of options. There are many mathematical objects isomorphic to sets of worlds that could play the role of propositions equally well, such as functions from worlds to truth values. There are similarly many mathematical objects isomorphic to functions from individuals to sets of worlds that could play the role of properties: functions from worlds to extensions (i.e. sets of individuals), sets of pairs of worlds and individuals, functions from pairs of individuals and worlds to truth values, functions from worlds to functions from individuals to truth values, and so on. Similar variant isomorphic representations are available for relations, operators, and so on. It is hard to adjudicate between these different variations.

One nice thing about the corresponding higher-order questions we will study is that there is not a similar abundance of ways of formulating these questions. The principles we will consider are formulated using unrestricted quantification into sentence position. The question

of reducing one sort of type t entity to another simply doesn't arise as it does for propositions and sets—propositions and sets or special *sorts* of individual, whereas the corresponding higher-order claims are unrestricted—so the question 'which kind of individual is a proposition' has no higher-order analogue.

6.1 Propositional individuation: propositional Booleanism

Let's begin by considering a very strong principle about propositional individuation:

The Fregean Axiom $(A \leftrightarrow B) \rightarrow A =_t B$

Here A and B can be any terms of type t. This principle tells us that material equivalence suffices for identity at type t. There is a related view about first-order propositions, namely that there are only two of them: the true and the false. But observe that unlike this principle, which involves the word 'proposition', The Fregean Axiom is formulated in completely logical terms and is thus a thesis of logic proper.

Exercise 6.1. *'there are two' is a first-order quantifier (we see it combining with a first-order predicate 'cat' in 'there are two cats', for instance). However, there is an analogous quantifier at type t, and it can be defined from the type t quantifiers and type t identity in the same way as the first-order variant. The analogue of the claim that there are two propositions is just* $\exists_t pq(p \neq_t q \wedge \forall_t r(r =_t p \vee r =_t q))$. *Provide an informal proof of this from The Fregean Axiom.*

The early versions of higher-order logic, following Frege (1879), would often make this assumption.[1] While this assumption may be harmless for many mathematical applications of higher-order logic, it has some pretty surprising philosophical implications. It seems, for instance, to be possible for John to believe that snow is white without believing that there are infinitely many primes. Leibniz's law for the propositional identity connective, $=_t$, lets us infer that the proposition that snow is white is distinct from the proposition that there are infinitely many primes: for if they were identical then by Leibniz's law, John believes that snow is white only if he believes that there are infinitely many primes. This contradicts the Fregean Axiom since both of these propositions are true, and so identical.

Frege was certainly aware of this argument and had things to say about it that preserved his theory. But whether or not Frege's response was correct, there is a big problem with the above style of argument. One distinctive feature of attitude verbs is that this exact style of argument could be used to infer many obviously false distinctness claims. Take, for instance

Ptolemy believed that Hesperus appeared in the morning.

Ptolemy did not believe that Phosphorus appeared in the morning.

Therefore Hesperus and Phosphorus are distinct.

It is quite clear that the conclusion of this argument is false: Hesperus and Phosphorus are one and the same planet. Philosophers have considered many different treatments of attitude verbs in which this inference fails in some way or other so as to preserve this basic astronomical fact, but they all agree that you can't infer distinctness claims straightforwardly from face-value judgements about attitude reports.

This example concerns a first-order distinctness claim, but I think given the close analogy between first-order and higher-order identity we should be just as cautious about inferring higher-order distinctness claims—such as the distinctness of the proposition that there are

infinitely many primes from the proposition that snow is white—from face-value judgements about attitude reports.[2] It follows that attitudes should not play a straightforward role in our theorizing about proposition individuation.

However, the problems for The Fregean Axiom extend well beyond propositional attitudes to operators that appear to be in just as good a standing as negation. For instance, the operator expression *it is necessary that* does not express a propositional attitude, and whether something is necessary or not does not seem to depend on the way in which the world is presented to us. Indeed the operator appears to be *transparent*, in the sense that the application of Leibniz's law is licensed in its scope. This lets us build a more compelling case against The Fregean Axiom: it is necessary that there are infinitely many primes ($\Box A$), but it does not seem to be necessary that snow is white ($\neg\Box B$). Since Leibniz's law is licensed, $A =_t B \to \Box A \to \Box B$, we can infer that these propositions are distinct, $A \neq_t B$, despite being materially equivalent.

Operators that differ with regard to materially equivalent propositions are called *intensional*, or *non-extensional*. Both notions can be defined in higher-order logic:

$$\text{Extensional}_{t\to t} := \lambda X.\forall_t pq((p \leftrightarrow q) \to (Xp \leftrightarrow Xq))$$

$$\text{Intensional}_{t\to t} := \lambda X.\neg\,\text{Extensional}_{t\to t}\, X$$

It may be verified from the axioms of propositional logic that the truth-functional connectives ($\neg$, $\to$, $\wedge$ etc.) are extensional operators.

While there is no consensus on how the puzzles surrounding propositional attitudes should be solved, most philosophers recognize the existence of intensional operators. Examples include other modal operators ('possibly A', 'actually A' etc.), counterfactual connectives ('if it were the case that A then it would be the case that B'), tense operators ('always A', 'sometimes A'). Even though operator expressions like these do not permit the substitution of materially equivalent sentences *salve veritate*, they *do* seem to permit the substitution of *logically equivalent* sentences. For example, if 'it's necessary that P and Q' is true then so is 'it's necessary that Q and P'. This is also true according to many standard logics for tense operators and counterfactuals.[3]

These facts are predicted by a much more general postulate: every genuine operator applying to $P \wedge Q$ also applies to $Q \wedge P$, and similarly for other logical equivalences like this. Given the coextensiveness of propositional identity with Leibniz equivalence (i.e. the relation of sharing all operators) proved in Exercise 5.7, we can infer that $P \wedge Q$ is identical to $Q \wedge P$, and may infer many other similar identities between logical equivalents.

In order to formulate this as a principle of higher-order logic, we need to be precise about what 'logical equivalence' actually means. A very natural thing to mean here is provable equivalence in H, and we will explore this in a moment. For now, we will focus on equations between propositions formulated in terms of the truth-functional connectives, and in this case, we may simply take logical equivalence to mean equivalence in propositional logic. Thus we should accept all identities

$$A =_t B$$

where A and B are propositional formulae and $A \leftrightarrow B$ is a theorem of propositional logic.

Let us consider some special cases of this schema. Certainly biconditionals like $A \vee B \leftrightarrow B \vee A$ and $A \wedge A \leftrightarrow A$ are theorems of propositional logic, so we can infer identities such as

Table 6.1 Propositional Booleanism

$\forall_t pqr((p \vee q) \vee r) =_t (p \vee (q \vee r))$ $\forall_t pqr((p \wedge q) \wedge r) =_t (p \wedge (q \wedge r))$	Associativity^t
$\forall_t pq((p \vee q) =_t (q \vee p))$ $\forall_t pq((p \wedge q) =_t (q \wedge p))$	Commutativity^t
$\forall_t pq((p \vee (p \wedge q)) =_t p)$ $\forall_t pq((p \wedge (p \vee q)) =_t p)$	Absorption^t
$\forall_t p((p \vee \bot) =_t p)$ $\forall_t p((p \wedge \top) =_t p)$	Identity^t
$\forall_t pqr((p \vee (q \wedge r)) =_t (p \vee q) \wedge (p \vee r))$ $\forall_t pqr((p \wedge (q \vee r)) =_t (p \wedge q) \vee (p \wedge r))$	Distributivity^t
$\forall_t p((p \vee \neg p) =_t \top)$ $\forall_t p((p \wedge \neg p) =_t \bot)$	Complements^t
$\forall_t p((p \rightarrow q) =_t (\neg p \vee q))$	Conditional^t

$A \vee B =_t B \vee A$, $A \wedge A =_t A$. Since there are instances of these axioms where A and B are variables of type t, we infer the universal generalizations as well: $\forall_t pq(p \vee q =_t q \vee p)$ and $\forall_t p(p \wedge p =_t p)$. Indeed, it is possible to capture all identities that correspond to equivalences in propositional logic with a finite set of axioms, called the *Boolean identities*: see Table 6.1 (these axioms are especially symmetric, but there are also many more compact axiomatizations of the Boolean identities and axiomatizations that use fewer primitive connectives).[4] We will call this theory *Propositional Booleanism*. Observe that Propositional Booleanism specifically concerns the behaviour of the Boolean connectives $\neg, \wedge, \vee, \top$ and $\bot$, and thus says nothing about the other logical constants, $\forall_\sigma$, $\exists_\sigma$ and $=_\sigma$.

There is, of course, an analogue of Propositional Booleanism for first-order propositions: that they satisfy the axioms of a Boolean algebra—the obvious analogues of the axioms in Table 6.1 with first-order quantifiers restricted to propositions and first-order identities instead of type t quantifiers and identities, and functions on propositions representing the truth-functional operations instead of the truth-functional connectives. The first-order version of Booleanism is of special interest because it falls out of the well-known possible worlds theory of propositions. According to this theory, first-order propositions are simply subsets of a set of metaphysically possible worlds, W, and the operations of conjunction, disjunction, and negation on propositions are given by the set-theoretic operations of intersection, union and complementation with respect to W. $\top$ and $\bot$ correspond to the sets W and $\emptyset$ respectively. Note, however, that while the claim that first-order propositions form a Boolean algebra falls out of this picture, it is strictly weaker for it does not imply that propositions are individuated by metaphysically necessary equivalence—the statement that propositions form a Boolean algebra is formulated in terms of first-order identity and makes no mention of the notion of metaphysical necessity. The Boolean equations are also satisfied, for instance, by theories that identify propositions with sets of ordered pairs of worlds with something else (these could be times, or precisifications, corresponding to theories in which tense and determinacy operators can distinguish necessarily equivalent propositions).

Remark 6.2. Observe that the thesis that propositions are individuated by metaphysically necessary equivalence has a higher-order counterpart stating that propositions are so individuated. To do so we need a primitive operator $\Box_M : t \rightarrow t$ for metaphysical necessity:

$\Box_M(A \leftrightarrow B) \rightarrow A =_t B.$

The higher-order theory obtained by adding this axiom to **H** is not a higher-order *logic* since it says something specific about the non-logical operator $\Box_M$ (one cannot apply the rule of substitution to $\Box_M$ in the above axiom).

Besides the involvement of metaphysical necessity, the possible worlds theory makes several other posits that are not consequences of the Boolean equations alone. It posits the existence of 'world' propositions—consistent propositions W whose conjunction with any other proposition is either W or $\bot$. There are algebras satisfying the Boolean equations that do not posit world propositions. Since one can take arbitrary unions and intersections of sets, the possible worlds theory also posits infinitary analogues of the operations of disjunction and conjunction—a posit that similarly does not follow from the Boolean equations alone. Similarly, in the higher-order case, Propositional Booleanism is neutral about the existence of world propositions and infinitary completeness. We will return to these implications in Chapter 8.

6.2 Propositional individuation: weaker theories

Propositional Booleanism was motivated by the observation that, once you have set aside putative operators expressed by attitudes (where we cannot naïvely apply Leibniz's law), and focus on the sorts of operators we study in metaphysics—necessity, counterfactuals, etc.—logical equivalents appear to be intersubstitutable. But the claim that metaphysics does not draw distinctions between Boolean equivalents has been brought under scrutiny. Let us consider a couple of strands of thought in recent metaphysics that postulate operators that draw distinctions between Boolean equivalents.

One such challenge comes from work on grounding. Metaphysicians working in this tradition posit a special form of metaphysical explanation which can be expressed by an $n+1$-ary connective between n explanans and an explanandum: $P_1, \ldots, P_n$ fully grounds Q, read as saying that the truth of $P_1, \ldots, P_n$ fully explain (in this metaphysical sense) the truth of Q.[5] This connective, according to most theorists of grounding, does not permit the substitution of Boolean equivalents. Perhaps the easiest way to illustrate this is through the binary connective of *strict partial ground*: P is a strict partial ground of Q, written $P \prec Q$, iff P, possibly along with some other propositions, fully ground Q. Strict partial ground is supposed to be an asymmetric relation: one cannot have circular metaphysical explanations. Thus we should accept:

$P \not\prec P$

A disjunction is fully grounded by any of its true disjuncts, so we should also accept:

$P \rightarrow P \prec (Q \vee P)$

From these facts we can show that the operator of being grounded by P, $\lambda r.(P \prec r)$, for some fixed truth P, does not licence the substitution of Boolean equivalents. So by Leibniz's law at type t, Propositional Booleanism must be false. The argument refutes the law Absorption, which says $\forall_t pq.((p \wedge q) \vee p) =_t p$. By the second principle and the assumption that P was true, $P \prec (P \wedge Q) \vee P$. But by the Law of Absorption $((P \wedge Q) \vee P) =_t P$ and Leibniz's law we may immediately infer $P \prec P$, contradicting the first assumption. (The notion of

explanation appealed to here, according to its practitioners, is metaphysical not epistemic—so the application of Leibniz's law is not subject to the sorts of scrutiny we must apply when we use Leibniz's law in epistemic and other attitudinal contexts.)

There is another strand of recent metaphysics that could be taken to refute Propositional Booleanism. According to this worldview, there are a special collection of *fundamental* properties, propositions, relations etc. from which all properties, propositions, relations, etc. can be built. To theorize about this notion of one entity being buildable from another we can adopt, for any choice of types $\sigma_1, \ldots, \sigma_n, \tau$, a relation $\rhd : \sigma_1 \to \cdots \to \sigma_n \to \tau \to t$. The claim that $B : \tau$ can be built from the entities $A_1 : \sigma_1, \ldots, A_n : \sigma_n$ would be formalized $A_1 \ldots A_n \rhd B$. We might also wish to assert that an entity *requires* another, with an operator $\blacktriangleright : \sigma \to \tau \to t$, to mean that $B : \tau$ couldn't be built without $A : \sigma$. It would be subject to the axiom:[6]

$$A_1 \ldots A_n \rhd B \wedge A \blacktriangleright B \to \bigvee_{A_i:\sigma} A =_\sigma A_i$$

The most famous version of this picture is surely the structured theory of propositions. This is usually presented as a first-order theory of propositions and properties: according to it propositions and properties are structured like sentences and predicates of a language, and one entity is built from some others when the former literally contains those entities as constituents. However there are many more coarse-grained theories that can make sense of these and similar notions.[7]

Some philosophers who reject the more general notion of constituent requirement will recognize special cases of it, such as the notion of a proposition being *about* an individual: for instance *Barcan was a modal logician* is about Ruth Barcan, not Arthur Prior. This is a special case of the notion of requirement introduced above where an individual appears on the left and a proposition on the right. Certain principles governing propositional aboutness (and more generally requirement) are inconsistent with Propositional Booleanism. For instance, consider the principle that if P is about Ruth Barcan, then any truth-functional complex involving P is also about Ruth Barcan:

$$a \blacktriangleright P \to a \blacktriangleright (P \wedge Q)$$

$$a \blacktriangleright P \to a \blacktriangleright (P \vee Q)$$

$$a \blacktriangleright P \to a \blacktriangleright \neg P$$

On the other hand, propositional aboutness is not a vacuous notion: there is at least one proposition, e.g. that Prior was a logician, which is not about Ruth Barcan. But now we can obtain a contradiction as follows: by assumption *Barcan was a logician* is about Ruth Barcan, but then *(Barcan was a logician and Prior was a logician) or Prior was a logician* is also about Ruth Barcan, and finally by the Law of Absorption *Prior was a logician* is about Ruth Barcan, contradicting our other assumption. (Formally: $a \blacktriangleright P$. Then by the first principle we may infer $a \blacktriangleright (P \wedge Q)$, by the second principle $a \blacktriangleright ((P \wedge Q) \vee Q)$. But finally, by the Law of Absorption, $((P \wedge Q) \vee Q) =_t Q$ so $a \blacktriangleright Q$, contradicting the assumption that Q is not about a for an arbitrary proposition Q.)

It is possible to theorize in terms of notions like $\rhd$ and $\blacktriangleright$ without rejecting Propositional Booleanism (see for instance Bacon (2020)) but doing so requires one to reject the idea that aboutness is preserved under truth-functional operations. What alternatives to

Propositional Booleanism might be consistent with operators like $\prec$ and $\blacktriangleright$ behaving in the ways described above. If one was being guided by the structured theory of propositions mentioned earlier, one might be motivated to reject all of the principles listed in Table 6.1 . For instance, on that view whenever $P \neq_t Q$, $P \wedge Q \neq_t Q \wedge P$ because the order in which P and Q are conjoined contributes to the structure of a proposition. The structured picture can similarly be seen to be at odds with the other equations in Table 6.1.

Note, however, that it was the principle of Absorption in both cases that caused the problems. So we might look for fragments of Propositional Booleanism that do not imply the principle of Absorption that might be more congenial to hyperintensional metaphysics.

Remark 6.3. The reader may be wondering why we have decided to restrict attention to equations in the connectives that form a mere fragment of Propositional Booleanism, thus excluding from consideration collections of equations that are stronger, or neither stronger or weaker than Propositional Booleanism.

The reason is that any universally quantified set of equations in the truth-functional connectives that is not weaker or equivalent to Propositional Boolean theory is inconsistent in H. Suppose that $\forall_t p_1 \ldots p_n.A =_t B$ is a universally quantified equation where A and B involve only the truth-functional connectives and the variables $p_1, \ldots, p_n : t$. Suppose moreover that this equation is not already a consequence of the Boolean equations. By previously mentioned facts, this means that the propositional formula $A \leftrightarrow B$ is not a theorem of the propositional calculus, and so by the completeness theorem for the propositional calculus there is a valuation v on the variables $p_1, \ldots, p_n$ that makes $A \leftrightarrow B$ false. Suppose, without loss of generality, that $v(A) = 1$ and $v(B) = 0$. If A' and B' are the result of replacing, in A and B, the letters that are true in v with $\top$, and the letters that are false with $\bot$, A' will be a tautology, B' a contradiction, and $A' \leftrightarrow B'$ a contradiction. Now we can see that the principle $\forall_t p_1 \ldots p_n.A =_t B$ must be inconsistent: we can infer by repeated applications of universal instantiation that $A' =_t B'$ (instantiating each variable for $\top$ or $\bot$ accordingly). But then we can infer from this (by Leibniz's law and $A' \leftrightarrow A'$) the truth-functional contradiction $A' \leftrightarrow B'$.

One way find to weakenings of Propositional Booleanism is to look at consequences of Propositional Booleanism that could instead be taken as axioms. Some useful consequences of Propositional Booleanism are listed in Table 6.2. I will focus on a weakening that have been proposed in the context of propositional aboutness; similar weakenings have been also been proposed for theorizing about propositional grounding.

The first we will call Propositional Agglommerativism, and it states that propositions satisfy the axioms of an 'agglommerative algebra' in the sense of Goodman (2018a). Goodman has proposed this weakening of the Boolean equations with the purpose of modelling the notion of aboutness discussed earlier. It can be informally motivated by the thought that logically equivalent propositions that are about the same individuals are the same. Thus, for instance, unlike with Booleanism, there may be many tautologous propositions that differ about what individuals they are about, but there is only one qualitative tautologous proposition. (A *qualitative* entity is one that is not about any individuals at all.)

In this formalism, then, $\top$ and $\bot$ must be taken as primitive, and informally understood to mean, respectively, the qualitative tautologous proposition and the qualitative contradictory proposition.[8] The Propositional Aggolemerativist then may take the following as axioms:

Table 6.2 Some consequences of Propositional Booleanism

$\forall_t p(p =_t (p \vee p))$ $\forall_t p(p =_t (p \wedge p))$	Idempotence^t
$\forall_t p(p =_t \neg\neg p)$	Involution^t
$\forall_t pq(\neg(p \vee q) =_t (\neg p \wedge \neg q))$ $\forall_t pq(\neg(p \wedge q) =_t (\neg p \vee \neg q))$	DeMorgan^t
$\forall_t p((p \wedge \bot) =_t (p \wedge \neg p))$ $\forall_t p((p \vee \top) =_t (p \vee \neg p))$	Contradiction^t Tautology^t
$\forall_t pq((p \wedge \neg p) \vee (q \wedge \neg q) =_t (p \wedge \neg p) \wedge (q \wedge \neg q))$ $\forall_t pq((p \vee \neg p) \wedge (q \vee \neg q) =_t (p \vee \neg p) \vee (q \vee \neg q))$	Flattening^t
$\forall_t pqr(p \equiv_L q \rightarrow q \equiv_L r \rightarrow p \equiv_L r)$, where $p \equiv_L q$ means $p \wedge q =_t p \vee q$	Transitivity^t

Propositional Agglomerativism The principles of Associativity^t, Commutativity^t, Distribution^t and Identity^t from Table 6.1 and the principles Involution^t, Idempotency^t, deMorgan^t, Contradiction^t and Tautology^t, Flattening^t and Transitivity^t from Table 6.2 .

Observe how the same propositional variables always appear on both sides of the equations that are taken as axioms of Propositional Agglomerativism. The identity Absorption, $p =_t (p \wedge q) \vee p$ for instance, introduces the new variable q on the right-hand-side, yet q might involve further individuals that p is not about, and so this equation is not acceptable to someone who thinks that if q is about an individual then any truth-functional complex involving q is also about that individual.

This is just one example of a weakening of Propositional Booleanism. Other theories of propositional granularity can be formulated in a completely parallel way. Kit Fine, in various works, has suggested weakenings of Propositional Booleanism that are more friendly to a notion of ground, but which nonetheless have a strong positive theory of propositional identity that includes laws like DeMorgan^t, Involvution^t, Commutativity^t, Associativity^t, and in some versions Idempotence^t.[9]

6.3 Individuating properties and relations: Booleanism and weakenings

In this section, we extend our discussion of individuation from propositions to properties and relations. Our focus will thus be entities with a relational type—a type that ends in a t such as $e \rightarrow t$, $t \rightarrow t$, $e \rightarrow e \rightarrow t$, etc (see Definition 1.2). We will see that all of the theories governing propositional individuation considered in the previous section have analogues for properties, operators, relations, etc.

Let's begin with The Fregean Axiom. It is an instance of a much more general principle stating that entities with any relational type are individuated by coextensiveness. Thus, given relations $R, S : \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow t$ (we will use τ to denote this type), if R and S apply to all the same entities $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, then $R =_{\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow t} S$. We'll call this principle Extensionalism:

Extensionalism $\forall_{\sigma_1} x_1 \ldots \forall_{\sigma_n} x_n (Rx_1 \ldots x_n \leftrightarrow Sx_1 \ldots x_n) \rightarrow R =_\tau S$

where $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, and $\tau = \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow t$. Observe that The Fregean Axiom is a special case of this where $n = 0$, for in that case R and S have type t and the antecedent simply states their material equivalence.

Table 6.3 The Boolean relational operations ($X, Y : \overline{\sigma} \to t$ and $\overline{z} : \overline{\sigma}$)

$\wedge_{\overline{\sigma}\to t}$	:=	$\lambda XY\lambda\overline{z}.(X\overline{z} \wedge Y\overline{z})$
$\vee_{\overline{\sigma}\to t}$	:=	$\lambda XY\lambda\overline{z}.(X\overline{z} \vee Y\overline{z})$
$\neg_{\overline{\sigma}\to t}$	:=	$\lambda X\lambda\overline{z}.\neg(X\overline{z})$
$\top_{\overline{\sigma}\to t}$	:=	$\lambda\overline{z}.\top$
$\bot_{\overline{\sigma}\to t}$	:=	$\lambda\overline{z}.\bot$

Like The Fregean Axiom, the instances of Extensionalism concerning properties and relations (i.e. where $\tau \neq t$) are highly contentious. The property of being a 20 kg ant is coextensive with the property of being a round square, but these properties have many different higher-order properties—for instance, it is presumably possible that there are 20 kg ants but not possible that there are round squares so one but not the other has the higher-order property of being possibly instantiated ($\lambda X.\Diamond\exists_e y.Xy$).

All of the theories of propositional granularity that we considered in the previous section have analogues at the level of properties and relations. Let's begin by generalizing Propositional Booleanism to properties of type $e \to t$. Propositional Booleanism states that propositions form a Boolean algebra with respect to the connectives $\neg, \vee, \wedge, \top$ and $\bot$. But recall in Chapter 3 we observed that there was an analogue of conjunction for properties of type $e \to t$: it takes, for instance, the property of being wise (i.e. $W : e \to t$) and the property of being old ($O : e \to t$) to the property of being wise and old ($\lambda z.(Wz \wedge Oz) : e \to t$). It was defined as $\lambda X\lambda Y\lambda z(Xz \wedge Yz)$. Since it is the analogue of conjunction for $e \to t$ properties we shall write $\wedge_{e\to t}$. Obviously there are versions of disjunction ($\lambda XYz.(Xz \vee Yz)$), negation ($\lambda Xy.\neg(Xy)$), $\top_{e\to t}$ ($\lambda x.\top$) and $\bot_{e\to t}$ ($\lambda x.\bot$) for properties. So the analogue of Propositional Booleanism for $e \to t$ properties states that properties satisfy the equations of a Boolean algebra with respect to the operations $\wedge_{e\to t}, \vee_{\to t}, \neg_{e\to t}, \top_{e\to t}$ and $\bot_{e\to t}$. So, for instance, the commutativity of property conjunction becomes:

$$\forall_{e\to t}XY.(X \wedge_{e\to t} Y) =_{e\to t} (Y \wedge_{e\to t} X)$$

Let us attempt to generalize this to arbitrary relational types. Let us first introduce some conventions:

Convention 6.1. *We will write $\overline{\sigma}$ to stand for a sequence of types $\sigma_1, \ldots, \sigma_n$. We may write $\overline{\sigma} \to \tau$ to stand for the type $\sigma_1 \to \cdots \to \sigma_n \to \tau$.*

We will similarly write $\overline{x}$ to stand for a sequence of variables $x_1, \ldots, x_n$, and $\overline{x} : \overline{\sigma}$ to mean that $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ (and more generally $\overline{M} : \overline{\sigma}$ for a sequence of terms $M_1 : \sigma_1, \ldots, M_n : \sigma_n$). Finally, if $X : \overline{\sigma} \to \tau$ and $\overline{y} : \overline{\sigma}$ we will write $X\overline{y}$ for $Xy_1, \ldots, y_n$, and $\lambda\overline{y}.M$ for $\lambda y_1 \ldots \lambda y_n.M$.

For any sequence of types $\overline{\sigma}$ we can introduce the relational analogues of the Boolean connectives for relations of type $\overline{\sigma} \to t$, as in Table 6.3 . In this table $X, Y : \overline{\sigma} \to t$ and $\overline{z} : \overline{\sigma}$.

With these operations defined, we can now formulate the thesis that relations of type $\overline{\sigma} \to t$ satisfy the Boolean equations with respect to the Boolean operations for that type. Our first axiomatization of this idea will be direct, through an infinite set of universally quantified equations—twelve universally quantified equations for each relational type—saying that the relations at that type form a Boolean algebra under the Boolean operations for that type.

Convention 6.2. *For each principle of Propositional Booleanism in Table 6.1 , and relational type ρ, we shall use the name of that principle with the superscript ρ to mean the principle that*

results from replacing the t subscripts on the quantifiers and identity with ρ and appending ρ as a subscript on the connectives. (Observe that when $\rho = t$, $\wedge_\rho$ is just $\wedge$ by definition, and so the names Commutativity^t, Absorption^t etc. are consistent with our previous usage.)

For instance, Commutativity$^\rho$ denotes the principle $\forall_\rho XY.(X \wedge_\rho Y) =_\rho (Y \wedge_\rho X)$.

Booleanism for ρ relations is then the principles of Identity$^\rho$, Commutativity$^\rho$, Absorption$^\rho$ etc.

Strictly speaking, Booleanism is the result of adding these axioms to H and closing under the rule of Modus Ponens and Gen, but we can summarize it informally as follows:

Booleanism For every relational type ρ, the relations of type ρ form a Boolean algebra under the Boolean operations for ρ.

It turns out, however, that the claim that the Boolean identities hold at each relational type follows from a much simpler finite set of axioms. We can illustrate the idea with the different versions of the principle of Commutativity, stating that for any pair of propositions, properties or relations, conjoining them yields the same proposition, property or relation as conjoining them in the opposite order. Each of these principles follows from a single principle stating that conjunction is symmetric, in the sense that it's identical to its own converse.

$$\wedge =_{t\to t\to t} \lambda pq.(q \wedge p)$$

We can simplify this somewhat using the C combinator: $\wedge =_{t\to t\to t} C^{ttt}\wedge$.

Let's see how this principle implies Propositional Commutativity, and then property Commutativity (and then, by extrapolation, all forms of commutativity). In this argument it will be more perspicuous to use prefix notation. Propositional Commutativity then says $\forall_t pq.(\wedge pq) =_t (\wedge qp)$. First note that we can prove, using the reflexivity of identity, $\forall_t pq.(\wedge pq) =_t (\wedge pq)$. Since $\wedge =_{t\to t\to t} \lambda rs.(s \wedge r)$ we can apply Leibniz's law and replace the second $\wedge$ in this claim to get $\forall_t pq.(\wedge pq) =_t ((\lambda rs.(s \wedge r))pq)$. But finally, by β, we can infer $\forall pq.(\wedge pq) =_t (\wedge qp)$.

Completely parallel reasoning can be used to establish the principle of property Commutativity.

Exercise 6.2. *Given the assumption that $\wedge$ is symmetric ($\wedge =_{t\to t\to t} \lambda pq.(q \wedge p)$) prove the principle of property Commutativity: $\forall_{e\to t} XY.(\lambda z.(Xz \wedge Yz) =_{e\to t} \lambda z.(Yz \wedge Xz)$.*

We have focused on Commutativity, but for each of the identities of Propositional Booleanism there is a corresponding identity between connectives. The recipe is this: take your universally quantified propositional identity, say $\forall_t p.(p \wedge \top) =_t p$, remove the outer quantifier, and prefix a string of λs onto the terms on either side of the identity so that all of the free propositional variables on both sides become bound, and replace the propositional identity with a connective identity of the suitable type. So in this case the result is $\lambda p.(p \wedge \top) =_{t\to t} \lambda p.p$, stating that the operation of conjunction with $\top$ is the identity operator. Applying this to recipe to Commutativity^t we get (in infix notation) $\lambda pq.((\wedge p)q) =_{t\to t\to t} \lambda pq.((\wedge q)p)$. But the left-hand-side is η-equivalent to $\lambda q(\wedge q)$ and this is η-equivalent to $\wedge$, so Commutativity$^\lambda$ is equivalent to our earlier statement $\wedge = \lambda pq.(q \wedge p)$.

Proposition 6.1. *Booleanism may be axiomatized by the equations in Table 6.4*

By extending the earlier reasoning we can prove from Booleanism all of the equations of Propositional Booleanism, and indeed Relational Booleanism for each relation of type $\bar{\sigma} \to t$.

Table 6.4 The axioms of Booleanism

$\lambda pqr.((p \vee q) \vee r) =_{t\to t\to t\to t} \lambda pqr.(p \vee (q \vee r))$ $\lambda pqr.((p \wedge q) \wedge r) =_{t\to t\to t\to t} \lambda pqr.(p \wedge (q \wedge r))$	Associativity$^{\lambda}$
$\lambda pq.(p \vee q) =_{t\to t\to t} \lambda pq.(q \vee p)$ $\lambda pq.(p \wedge q) =_{t\to t\to t} \lambda pq.(q \wedge p)$	Commutativity$^{\lambda}$
$\lambda pq.(p \vee (p \wedge q)) =_{t\to t\to t} \lambda pq.p$ $\lambda pq.(p \wedge (p \vee q)) =_{t\to t\to t} \lambda pq.p$	Absorption$^{\lambda}$
$\lambda p.(p \vee \bot) =_{t\to t} \lambda p.p$ $\lambda p.(p \wedge \top) =_{t\to t} \lambda p.p$	Identity$^{\lambda}$
$\lambda pqr.(p \vee (q \wedge r)) =_{t\to t\to t\to t} \lambda pqr.((p \vee q) \wedge (p \vee r))$ $\lambda pqr.(p \wedge (q \vee r)) =_{t\to t\to t\to t} \lambda pqr.((p \wedge q) \vee (p \wedge r))$	Distributivity$^{\lambda}$
$\lambda p.(p \vee \neg p) =_{t\to t\to t} \lambda p.\top$ $\lambda p.(p \wedge \neg p) =_{t\to t\to t} \lambda p.\bot$	Complements$^{\lambda}$
$\lambda pq.(p \to q) =_{t\to t\to t} \lambda pq.(\neg p \vee q)$	Conditionals$^{\lambda}$

Exercise 6.3. *Use Absorption$^{\lambda}$ (i.e. $\lambda pq.(p \vee (p \wedge q)) = \lambda pq.p$) to prove the principle of property Absorption. Absorption$^{e\to t}$ is $\forall_{e\to t}XY.(X \vee_{e\to t} (X \wedge_{e\to t} Y)) =_{e\to t} X$.*

Observe that while Booleanism entails Propositional Booleanism we cannot reverse the entailment. It does not seem to be possible to start with the universally quantified equations of Propositions Booleanism and derive the non-quantified identities between the λ-terms of Booleanism (Table 6.4). The reason is that the universally quantified equations of Propositional Booleanism impose constraints on the granularity of the propositions, but does not say anything directly about the Boolean operation of $\wedge$ itself. For instance, for all that Propositional Booleanism says, propositional conjunction and its converse, $\wedge$ and $\lambda pq.q \wedge p$, could be completely different connectives: Propositional Booleanism only requires that they output the same proposition whenever they are applied to the same pair of propositions. Similarly, $e \to t$ property Booleanism does not say that property conjunction is its own converse, it merely says that these two ways of combining properties to make a new property—property conjunction and its converse—yield the same output on the same input. Given these observations, one might begin to suspect that our second axiomatization of Booleanism, involving connective identities between closed λ-terms, as opposed to the universally quantified equations of our first axiomatization, is *strictly* stronger than the first axiomatization. This turns out not to be the case, but the argument is slightly non-obvious. It turns out that the claim that the ternary connectives form a Boolean algebra with respect to the Boolean operations for ternary connectives (i.e. the universally quantified equations of Table 6.1 with type subscript $t \to t \to t \to t$) imply the finite set of equations between λ-terms in Table 6.4 , and thus all of Booleanism.

Let's begin by assuming our first axiomatization of Booleanism, and attempt to derive Commutativity$^{\lambda}$—the principle that conjunction is its own converse $\lambda pq.(p \wedge q) =_{t\to t\to t} \lambda pq.(q \wedge p)$. The first axiomatization includes the claim that conjunction for binary connectives is commutative:

$$\forall_{t\to t\to t}XY.(X \wedge_{t\to t\to t} Y) =_{t\to t\to t} (Y \wedge_{t\to t\to t} X)$$

The trick to deriving Commutativity$^{\lambda}$ is to consider the connectives $\lambda pq.p$ and $\lambda pq.q$. By instantiating these connectives for X and Y respectively we get:

$$((\lambda pq.p) \wedge_{t\to t\to t} (\lambda pq.q)) =_{t\to t\to t} ((\lambda pq.q) \wedge_{t\to t\to t} (\lambda pq.p))$$

Now recall that $\wedge_{t\to t\to t}$ is an abbreviation of $\lambda XYpq(Xpq \wedge Ypq)$, so we have:

$$\lambda XYpq(Xpq \wedge Ypq)((\lambda pq.p)((\lambda pq.q) =_{t\to t\to t} \lambda XYpq(Xpq \wedge Ypq)((\lambda pq.q)((\lambda pq.p)$$

By β the left-hand-side reduces to $\lambda pq.(((\lambda pq.p)pq) \wedge (\lambda pq.q)pq)$ which in turn reduces to $\lambda pq.(p \wedge q)$. Similarly, the right-hand-side reduces to $\lambda pq.(((\lambda pq.q)pq) \wedge (\lambda pq.p)pq)$ and then to $\lambda pq.(q \wedge p)$. Thus we have derived Commutativity$^{\lambda}$ from Table 6.4 . And as we previously noted, this is η-equivalent to $\wedge = \lambda pq.(q \wedge p)$.

Exercise 6.4.

a. Use Identity$^{t\to t}$ (the claim that the unary operators form a Boolean algebra under the Boolean operations for unary operators $\wedge_{t\to t}$, $\vee_{t\to t}$ etc.) to prove Identity$^{\lambda}$ from Table 6.4 .

b. Use Absorption$^{t\to t\to t}$ (Absorption for binary connectives) to prove Absorption$^{\lambda}$.

c. Use Distribution$^{t\to t\to t\to t}$ (Distribution for ternary connectives) to prove Distribution$^{\lambda}$.

Using the ideas in the preceding discussion and exercises we can see that the universally quantified Boolean equations for unary, binary and ternary connectives imply the identities between the closed λ-terms in Table 6.4 . We can simplify this a little more:

Exercise 6.5. *Show that Absorption$^{t\to t\to t\to t}$ implies Absorption$^{t\to t\to t}$ and Absorption$^{t\to t}$. Briefly explain why Booleanism for ternary connectives implies Booleanism for binary and unary connectives.*

We may summarize this with the following proposition

Proposition 6.2. *The following three axiomatizations of Booleanism over H are equivalent*

1. *Booleanism at all relational types: The infinite set of universally quantified equations stating that the relations from each relational type form a Boolean algebra under the Boolean operations for that type (Associativity$^{\rho}$, Commutativity$^{\rho}$, Absorption$^{\rho}$, Identity$^{\rho}$, Distributivity$^{\rho}$, Complements$^{\rho}$ for every relational type ρ).*
2. *Booleanism for ternary connectives: The twelve universally quantified equations stating that the ternary connectives form a Boolean algebra under the Boolean operations for ternary connectives (Associativity$^{t\to t\to t\to t}$, Commutativity$^{t\to t\to t\to t}$, Absorption$^{t\to t\to t\to t}$, Identity$^{t\to t\to t\to t}$, Distributivity$^{t\to t\to t\to t}$, Complements$^{t\to t\to t\to t}$).*
3. *The twelve equations between closed λ-terms for connectives in Table 6.4 (Associativity$^{\lambda}$, Commutativity$^{\lambda}$, Absorption$^{\lambda}$, Identity$^{\lambda}$, Distributivity$^{\lambda}$, Complements$^{\lambda}$).*

Just as Property Booleanism does not follow from Propositional Booleanism, one might have thought that there is a generalization of the axioms in Table 6.4 stating identities not between propositional connectives but between property and relational operations. For instance, just as the claim that propositional conjunction is its own converse $\wedge =_{t\to t} (C^{tt}\wedge)$—Commutativity$^{\lambda}$—strengthens Commutativity^{t} ($\forall_t pq.(p \wedge q) =_t (q \wedge p)$), there is the principle that property conjunction $\wedge_{e\to t}$ is its own converse: $\wedge_{e\to t} =_{(e\to t)\to(e\to t)\to e\to t} C^{(e\to t)(e\to t)}\wedge_{e\to t}$. It turns out this principle is not really a strengthening, as the following

exercise illustrates: the identities between λ-terms in Table 6.4 are sufficient to derive these generalizations

Exercise 6.6. *Using the principle of Booleanism that conjunction is its own converse,* $\wedge =_{t\to t} (C^{tt}\wedge)$*, prove the corresponding identity for property conjunction. That is, prove that property conjunction is its own converse:* $\wedge_{e\to t} =_{(e\to t)\to(e\to t)\to e\to t} C^{(e\to t)(e\to t)}\wedge_{e\to t}$

For any equational theory of propositions there is an analogous strengthening of that theory stating that relations of any relational type satisfy the same equations at that type. Just as we were able to axiomatize the claim that all propositions, properties and relations form a Boolean algebra at each relational type under the relevant operations with a finite set of closed equations between λ-terms denoting connectives, we can apply an analogous trick to any other equational theory.

It is worth singling out the generalization of Propositional Agglomerativism to properties and relations for special attention here, since Propositional Agglomerativism is not an equational theory, in the sense that it cannot be axiomatized by a collection of universally quantified equations. To axiomatize Propositional Agglomerativism, for instance, one also needs the principle of Transitivity, which states a closure condition on the true propositional equations—all the equations derivable from Propositional Agglomerativism can hold without the closure condition also obtaining. A set of closed equations between λ-terms corresponding to the axioms of Propositional Agglomerativism will guarantee that all the equations of agglomerative algebras hold at each relational type, but will not obviously guarantee that the equations at any given type are closed under transitivity. So to formulate a sufficiently strong version of Agglomerativism that is akin to our formulation of Booleanism one needs to state a Transitivity axiom for each type. Let $\bar{\sigma}\to t$ be a relational type, and let $X \equiv_L^{\bar{\sigma}\to t} Y$ be short for $(X\wedge_{\bar{\sigma}\to t} Y) = (X\vee_{\bar{\sigma}\to t} Y)$

$$\forall_{\bar{\sigma}\to t}XYZ(X \equiv_L^{\bar{\sigma}\to t} Y \to Y \equiv_L^{\bar{\sigma}\to t} Z \to X \equiv_L^{\bar{\sigma}\to t} Z)$$

It seems, then, that we cannot obtain an axiomatization of Agglomerativism stated in terms of equations alone: one also needs a version of the Transitivity axiom for each relational type.

6.4 Individuating properties and relations: Classicism

Booleanism is a thesis about the truth-functional connectives: the axioms of Booleanism are equations between terms constructed from the truth-functional connectives and variables of type t. Booleanism is therefore neutral about identities involving the other logical vocabulary. For instance the identity $\exists_e x.(Fx\vee Gx) =_t (\exists_e x.Fx\vee\exists_e x.Gx)$ seems to fall out of the informal thesis that logical equivalence suffices for identity, but is not a theorem of Booleanism.

It is natural to seek a theory that stands to all of classical higher-order logic as Booleanism stands to propositional logic. In addition to the truth-functional connectives higher-order logic has the operations of quantification, $\exists_\sigma$ and $\forall_\sigma$, and identity, $=_\sigma$, at each type σ. So we should be looking for an equational theory that extends Booleanism that involves these operations as well as the truth-functional connectives. Because the principles we are about to consider say things about the granularity of properties, relations and other entities of relational type, but does not say anything about granularity at other types, it makes sense to restrict ourselves to a version of higher-order logic which only has relational types. (We will

Table 6.5 The classical identities

$\lambda Xy.(Xy \vee \forall_\sigma X) =_{(\sigma\to t)\to\sigma\to t} \lambda Xy.Xy$ $\lambda Xy.(Xy \wedge \exists_\sigma X) =_{(\sigma\to t)\to\sigma\to t} \lambda Xy.Xy$	Quantifier Absorption
$\lambda Xp.(p \vee \forall_\sigma X) =_{(\sigma\to t)\to t\to t} \lambda Xp.\forall y(p \vee Xy)$ $\lambda Xp.(p \wedge \exists_\sigma X) =_{(\sigma\to t)\to t\to t} \lambda Xp.\exists y(p \wedge Xy)$	Quantifier Distributivity
$\lambda yz.(y =_\sigma z) =_{\sigma\to\sigma\to t} \lambda yz.\forall_{\sigma\to t} X(Xy \leftrightarrow Xz)$	The Identity Identity

later give a more general formulation of this theory in the full type system, via the principle "Modalized Functionality" from Section 8.1 of Chapter 8.)

The brute force way to do this would be to add to H every identity between relational terms that corresponds to a biconditional theorem of H. Whenever every $\overline{\sigma}$ is a sequence of types, and R and S express a relation between entities of those types, i.e. $R, S : \overline{\sigma} \to t$, which are provably equivalent in H, we should have an identity between those terms.

Logical Equivalence $R =_{\overline{\sigma}\to t} S$ when $R\overline{x} \leftrightarrow S\overline{x} \in \mathsf{H}$ and $x \notin FV(R) \cup FV(S)$.

Here $\overline{x}$ is a sequence of variables with types in the corresponding sequence of types $\overline{\sigma}$—or simply, $\overline{x} : \overline{\sigma}$ using our earlier conventions. Adding every identity $R =_{\overline{\sigma}\to t} S$ where $R\overline{x} \leftrightarrow S\overline{x} \in \mathsf{H}$ to the axioms of H, and closing under the rules of Modus Ponens and Gen gives us another higher-order logic, which we shall call *Classicism*, because its identities correspond to the equivalences of classical higher-order logic. We will often abbreviate this theory as C. (Note that the resulting set of sentences is automatically closed under the rule of substitution.)

Unlike Booleanism, Logical Equivalence includes identities which involve the quantifiers or identity, such as $(=_t=_{t\to t\to t}=_t) =_t \top$.

Exercise 6.7. *Check that* $(=_t=_{t\to t\to t}=_t) =_t \top$ *(in prefix notation:* $=_t (=_{t\to t\to t}=_t=_t)\top$*) is well-typed, and show it is a consequence of Logical Equivalence.*

Exercise 6.8. *Find an example of a consequence of Logical Equivalence that is a non-trivial identity that:*

a. Involves a quantifier (of some type) on at least one side of the identity.

b. Involves the identity symbol (at some type) on at least one side of the identity and is different from $(=_t=_{t\to t\to t}=_t) =_t \top$.

It would be nice to have a more compact set of equations that capture the provable equivalences of higher-order logic, just as the Boolean equations in Table 6.4 compactly capture the equivalences of propositional logic. Since classical higher-order logic extends classical propositional logic, we should expect such an axiomatization to extend the equations of Booleanism with further equations involving the quantifiers and identity. An axiomatization of Classicism along these lines is given by adding the axioms in Table 6.5 to the axioms of Booleanism (Table 6.4). We will call these the "Classical Identities".

Once we have added identities corresponding to logical equivalences in classical higher-order logic (i.e. H) to classical higher-order logic, either by adding them all by brute force or finding a smaller set of identities that suffice to derive them all, we obtain the stronger higher-order logic we have called C (Classicism).[10] Since C is stronger than H, more things are logically equivalent, so we could in principle imagine a seemingly stronger theory where we add all identities corresponding to logical equivalences in C. If the result were indeed

stronger than Classicism, we could continue this process, adding identities to the next theory that are provably equivalent in the previous one. But in fact this series of strengthenings never gets off the ground: whenever C proves an equivalence it already proves the corresponding identity. That is to say, C is closed under the *Rule of Equivalence*

The Rule of Equivalence If $\vdash R\bar{x} \leftrightarrow S\bar{x}$ then $\vdash R =_{\bar{\sigma}\to t} S$ provided $x_1, \ldots, x_n \notin FV(R) \cup FV(S)$.

Closing H under the Rule of Equivalence gives us a third way of axiomatizing C. In fact, this will prove to be the most convenient system for reasoning in. The other axiomatizations, by contrast, wear their justifications on their sleeve but are unweildy. Thus, for the rest of the book, we will take this way of characterizing the theorems of C to be our official one for the purposes of reasoning in it:

Definition 6.1 (Classicism)**.** *Classicism, or C, is the smallest higher-order theory closed under the Rule of Equivalence.*

As with our discussion of H, one can demonstrate that a formula in a theorem of C by finding a derivation in the sense of Definition 5.4, except we may now use the Rule of Equivalence in our derivations.

We must show that these three systems are in fact equivalent.

Theorem 6.1. *The following higher-order logics are the same:*

1. *The result of closing H under the Rule of Equivalence*
2. *H + all instances of Logical Equivalence.*
3. *H + the Classical Identities.*

Proof. Every one of the Classical Identities is β-equivalent to an instance of Logical Equivalence. Take, for instance, the classical theorem $Xy \wedge \forall_\sigma X \leftrightarrow \forall_\sigma X$: the left-to-right direction just follows from conjunction elimination, and the right-to-left direction is a straightforward consequence of UI. Using β we learn that the biconditional $(\lambda Xy(Xy \wedge \forall_\sigma X)Xy \leftrightarrow (\lambda Xy.\forall_\sigma X)Xy \in \mathsf{H}$, and so by Logical Equivalence $\lambda Xy(Xy \wedge \forall_\sigma X) =_{(\sigma\to t)\to\sigma\to t} \lambda Xy.\forall_\sigma X$. So 3 is included in 2.

Similarly 2 is included in 1, since every instance of Logical Equivalence follows from an application of the Rule of Equivalence. Thus it remains only to show that every theorem of 1 belongs to 3.

First we show that if we can derive a sentence A from the Classical Identities, we can also derive $\lambda\bar{x}A = \lambda\bar{x}\top$, for any sequence of variables $\bar{x}$. This is proved by induction on the length of a proof: we show for each axiom A of H+the Classical Identities, $\lambda\bar{x}A = \lambda\bar{x}\top$ is derivable, and that the rules also preserve this property.

If A is an instance of PC1, PC2 or PC3 then it is a propositional tautology, so $\lambda\bar{x}A = \lambda\bar{x}\top$ is a consequence of the Classical Identities, since it follows from the Boolean identities. If it is an instance of β or η, $A \to A'$, then $\lambda\bar{x}(A \to A) = \lambda\bar{x}\top \to \lambda\bar{x}(A \to A') = \lambda\bar{x}\top$ is an instance of β or η respectively, and the antecedent follows from Booleanism. Suppose it is an instance of UI: $\forall_\sigma F \to Fa$. Then $\lambda\bar{x}.(\forall_\sigma F \to Fa) = \lambda\bar{x}.(\forall_\sigma F \to (\lambda Xy.Xy)Fa)$ by β, $= (\lambda\bar{x}.\forall_\sigma F \to (\lambda Xy(Xy \vee \forall_\sigma X))Fa)$ by Quantifier Absorption, $= (\lambda\bar{x}.\forall_\sigma F \to (Fa \vee \forall_\sigma F))$ by β, $= (\lambda v\bar{x}.\top)$ by Booleanism. If A is any of the Classical Identities, $a = b$, we must show that $\lambda\bar{x}(a = b) = \lambda\bar{x}\top$ also follows from the Classical Identities. Leibniz's law tells us that $a = b \to \lambda\bar{x}(a = a) = \lambda\bar{x}\top \to \lambda\bar{x}(a = b) = \lambda\bar{x}\top$. $a = b$ is a Classical Identity,

and so it suffices to show that $\lambda\bar{x}(a = a) = \lambda\bar{x}\top$. By the Identity Identity and β, $\lambda\bar{x}(a = a) = \lambda\bar{x}\forall X(Xa \leftrightarrow Xa)$. By Booleanism, $\lambda\bar{x}\forall X(Xa \leftrightarrow Xa) = \lambda\bar{x}\forall X\top$. Finally, Quantifier Distributivity let's us infer that $\forall X\top = \top$ which completes the proof. This final identity is derived as follows: $\forall X\top = \forall X(\top \vee ZX)$ (by Booleanism) $= \lambda Zp.\forall X(\top \vee ZX))Z\top$ by β, $= (\lambda Zp.(p \vee \forall Z))Z\top$ by Quantifier Distributivity, $= \top \vee \forall Z = \top$.

To complete the proof we must show that the rules of proof of H preserve "being $\top$". For Modus Ponens, suppose that $\lambda\bar{x}A = \lambda\bar{x}\top$ and $\lambda\bar{x}(A \rightarrow B) = \lambda\bar{x}\top$ are theorems of 3. Then $\lambda\bar{x}B = \lambda\bar{x}(\top \rightarrow B)$ by Booleanism, $= \lambda\bar{x}(A \rightarrow B)$ using β and $\lambda\bar{x}A = \lambda\bar{x}\top$, and finaly this $= \lambda\bar{x}\top$ using the final assumption.

For Gen, one must suppose that $\lambda\bar{x}(A \rightarrow B) = \lambda\bar{x}\top$ is a theorem of 3, and show that $\lambda\bar{x}(A \rightarrow \forall yB) = \lambda\bar{x}\top$ is too, provided $y \notin FV(A)$. I leave this as an instructive exercise applying the Quantifier Distributivity axiom and the Booleanism in conjunction.

Now we can see that if $R\bar{x} \leftrightarrow S\bar{x}$ is derivable from the Classical Identities, then so is $R =_{\bar{\sigma}\rightarrow t} S$. For by our lemma, $\lambda\bar{x}(R\bar{x} \leftrightarrow S\bar{x}) = \lambda\bar{x}\top$ is a consequence of the Classical Identities. By Booleanism, $\lambda\bar{x}R\bar{x} = \lambda\bar{x}(\top \rightarrow R\bar{x})$. Using our assumption and β, $\lambda\bar{x}(\top \rightarrow R\bar{x}) = \lambda\bar{x}((R\bar{x} \leftrightarrow S\bar{x}) \rightarrow R\bar{x})$, which y Booleanism, $= \lambda\bar{x}((R\bar{x} \leftrightarrow S\bar{x}) \rightarrow S\bar{x})$, which by our assumption $= \lambda\bar{x}(\top \rightarrow S\bar{x}$, and again by Booleanism, $= \lambda\bar{x}S\bar{x}$. Thus $\lambda\bar{x}R\bar{x} = \lambda\bar{x}S\bar{x}$, and so $R = S$ by η. □

6.5 Functionality principles

In the previous sections, we have formulated principles governing the granularity propositions, properties and relations. In general, we have focused on views that treat propositions, properties and relations uniformly: there is something unnatural about the view that propositions form a Boolean algebra under the Boolean operations, but that properties only form agglomerative algebra on the analogous operations.

It turns out some mismatched combinations of granularity at different relational types are actually impossible in H. For instance, one cannot be a Relational Booleanist at some relational type without being a propositional Booleanist. (This is just a generalization of the argument in Exercise 6.5 that Booleanism about ternary connectives implies Booleanism about binary and unary connectives.) In general, a theory of granularity at high types implies that theory at lower types. In the following exercise, you will prove that Idempotence at a "higher" type implies it at a "lower" type.

Exercise 6.9. *Suppose that τ is a relational type. Prove Idempotence$^{\tau}$ from Idempotence$^{\sigma\rightarrow\tau}$.*[i]

On the other hand, we can't directly infer a theory of granularity at a high type from a lower one: there is no way to take there is no straightforward way to derive Property Booleanism from Propositional Booleanism, for instance. Given two properties, F and G, one can derive using Propositional Booleanism the universally quantified equation:

$$\forall_e x.(Fx \wedge Gx) =_t (Gx \wedge Fx)$$

and thus, by β and the definition of property conjunction:

$$\forall_e x.((F \wedge_{e\rightarrow t} G)x =_t (G \wedge_{e\rightarrow t} F)x)$$

So, for every individual x, applying the conjunctive property $F \wedge_{e\rightarrow t} G$ to x yields the same proposition as applying the conjunctive property $G \wedge_{e\rightarrow t} F$ to x. $F \wedge_{e\rightarrow t} G$ and $G \wedge_{e\rightarrow t} F$ have the same *applicative behaviour*. More generally:

Definition 6.2 (Cofunctionality)**.** *Two relations* $F, G : \sigma \to \tau$ have the same applicative behaviour*, or* are cofunctional*, if and only if for any entity* $x : \sigma$*,* $Fx =_\tau Gx$*:*

$$\forall_\sigma x.Fx =_\tau Gx$$

But H does not prove that properties with the same applicative behaviour are identical. So we could add this principle as a further axiom schema:

Functionality $\forall_\sigma x(Fx =_\tau Gx) \to F =_{\sigma\to\tau} G$

This principle is schematic in both the types σ and τ, and the terms F and G: there is an instance of this schema for every pair of types σ and τ and pair of terms $F, G : \sigma \to \tau$.

It is worth emphasizing how the principle of Functionality differs from the principle of extensionality. Extensionality implies that coextensive properties, for instance *having a heart* and *having a kidney*, are identical. But while these two properties are plausibly coextensive, they are not plausibly cofunctional: the proposition that *Ellen has a heart* and the proposition that *Ellen has a kidney* are plausibly different since it's possible that they have different truth values, so the two properties output different (albeit materially equivalent) propositions when applied to Ellen. Finally, the principle of Functionality is simpler to state since, unlike material equivalence, there is a version of identity at each type. (Of course, for any *relational* type, ρ, one can introduce an analogue of material equivalence, $\leftrightarrow_\rho$, defined as in Table 6.3 , so there is a simple version of extensionality that looks like our statement of Functionality when σ and τ are relational types: $\forall_\sigma x(Rx \leftrightarrow_\tau Sx) \to R \leftrightarrow_{\sigma\to\tau} S$.)

Exercise 6.10. *The reader should convince themselves that the principle Functionality implies the analogue of Extensionalism for cofunctional relations,* $R, S : \sigma_1 \to \cdots \to \sigma_n \to t$*:*

$$\forall_{\sigma_1} x_1 \ldots \forall_{\sigma_n} x_n(Rx_1 \ldots x_n =_t Sx_1 \ldots x_n) \to R =_{\sigma_1\to\cdots\to\sigma_n\to t} S$$

In the presence of the Functionality principle, a theory of granularity for propositions will imply the analogous theory of granularity for arbitrary relational types. For instance, given the principle of Functionality, Propositional Booleanism implies Booleanism at all types and the Fregean axiom implies Extensionalism. The reader is encouraged to try the following exercises to get a feel for how these implications work.

Exercise 6.11. *Prove* $\text{Absorption}^{e\to t}$ *from* Absorption^t *and Functionality.*

Exercise 6.12. *Prove* $\text{Absorption}^\lambda$ *from* Absorption^t *and Functionality. Explain how our second axiomatization of Booleanism (Table 6.4) follows from Propositional Booleanism and Functionality.*

Exercise 6.13. *Prove Extensionalism from The Fregean Axiom and Functionality.*[ii]

The next exercise is about three increasingly strong senses in which a relation could be symmetric: it could be coextensive with its converse, it could be cofunctional with its converse, or it could be identical to its converse. These three senses collapse in the presence of the principle of extensionality, and the final two collapse in the presence of Functionality.

Definition 6.3 (Symmetric relation)**.** *A relation* $R : \sigma \to \tau \to t$ *is:*

1. Extensionally symmetric *when* $\forall_\sigma x \forall_\tau y.(Rxy \leftrightarrow Ryx)$.
2. Functionally symmetric *when* $\forall_\sigma x \forall_\tau y.(Rxy =_t Ryx)$.
3. Symmetric *when* $R =_{\sigma\to\tau\to t} C^{\sigma\tau} R$.

Exercise 6.14.

a. Show that a relation is symmetric only if it is functionally symmetric, and functionally symmetric only if it is extensionally symmetric.

b. Using the principle of extensionality prove that every extensionally symmetric relation is symmetric (and thus functionally symmetric by part a).

c. Using the principle of Functionality show that every functionally symmetric relation is symmetric.

Just as we connected Booleanism with the possible worlds treatment of propositions, there is also a connection between the principle Functionality and the treatment of properties as functions from individuals to sets of worlds in the possible worlds formalism. If a property is simply a set-theoretic function mapping individuals to sets of worlds, then an analogue of the Functionality principle holds according to the possible worlds account of properties: if two functions yield the same output on every input they must be the very same function. Indeed, when we later come to constructing models of higher-order logic, models where the domain of properties is given by a set of functions provide a flexible class of models validating the Functionality principle.

According to the Functionality principle, properties are like functions in the respect that they are individuated by their applicative behaviour. We might also consider other respects in which properties could be analogous to functions. Functions are also plenitudinous in the sense that whenever you can define a relation between two sets X and Y that relate each element of X to a unique element of Y, then there is some function corresponding to that relation. This means there are all sorts of hodge-podge functions, beyond what you can define from the basic functions of mathematics (such as successor, addition etc) using operations like function composition and the other operations on functions we can introduce using the $\mapsto$ notation.

So using this as our guide we might postulate a similar principle for properties in higher-order logic. For any binary relation $X : e \to t \to t$ such that for each individual y, there is a unique proposition p such that Xyp, there exists a property $Z : e \to t$ that maps each individual y to the unique proposition p such that Xyp—i.e. for all individuals y, $Xy(Zy)$. Let us use the abbreviation $\exists_e!x.A$ for the claim that there is a unique thing that is A: $\exists_e z \forall_e x(x = z \leftrightarrow A)$. Then we may define the notion of being a functional binary relation as follows:

$$\text{Functional}_{et} := \lambda X.\forall_e y \exists_t! z(Xyz)$$

Then our principle of plenitude for properties becomes:

$$\forall_{e\to t\to t} X(\text{Functional}\, X \to \exists_{e\to t} Z \forall_e y.Xy(Zy)$$

Everything we have said so far applies equally to relations. In the possible worlds formalism, these can be treated as n-ary functions, or equivalently, curried unary higher-order functions. First, we need to generalize the notion of a functional relation to more arguments: a relation R of type $\overline{\sigma} \to \tau \to t$ is functional when given inputs $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ there is a unique $y : \tau$ such that $Rx_1 \ldots x_n y$.

$$\text{Functional}_{\overline{\sigma}\tau} := \lambda X.\forall_{\overline{\sigma}} \overline{y} \exists_\tau! z(X\overline{y}z)$$

So we can formulate the general principle of Functional Plenitude as saying that for any relation relating a sequence $x_1, \ldots, x_n$ of types $\sigma_1, \ldots, \sigma_n$ of inputs to a unique output, y,

of type τ there is an operation of type $\sigma_1 \to \cdots \to \sigma_n \to \tau$ that yields the unique output when applied to $x_1 \ldots x_n$. (In formulating this we adopt Convention 6.1 where $\overline{x}$ stands for a sequence of variable $x_1 \ldots x_n$.)

Functional Plenitude $\forall_{\overline{\sigma} \to \tau \to t} X(\text{Functional } X \to \exists_{\overline{\sigma} \to \tau} Y. \forall \overline{y} X \overline{y} (Y \overline{y}))$

The connection between functions and our principle of Functional Plenitude will also make an appearance in the model theory of these principles (Chapter 15): one way to secure Functional Plenitude in a model of higher-order logic is to require that the domain of entities representing $\sigma \to \tau$ operations to be modelled by the set of *all* functions from the set of things representing σ entities in the model to the set of things representing τ entities in the model.

A common method of defining a function is by cases, by specifying what the function does to each argument. Here we'll see that Functional Plenitude allows this sort of definition.

Exercise 6.15 (Definition by cases)**.** *Let p and q be propositions, and $Y : e \to t$ a property of individuals. Show, given Functional Plenitude, that there is a property X such that for any individual z, $Xz = p$ if Yz and $Xz = q$ otherwise.*

Both the principle of Functionality and Functional Plenitude we motivated by analogy with the formalism that identifies properties and relations with functions. While this is a convenient identification in technical contexts, we should examine the metaphysical commitments of these principles.

Let us begin with the principle of Functionality. Let us focus on the instance of Functionality for $e \to t$ properties; similar considerations apply to other types. One source of counterexample comes from the possibility of individuals that don't actually exist: for instance Wittgenstein didn't in fact have any children, although it is commonly thought that he could have done. Functionality says that if F and G output the same proposition when applied to any actual individual then they are the same, but what if F and G only come apart when applied to a merely possible individual such as one of Wittgenstein's merely possible children? For concreteness let us assume, following the possible worlds theory, that propositions are individuated by (metaphysically) necessary equivalence and, following Kripke on the necessity of origins, that anyone who is not a child of Wittgenstein is necessarily so. It follows that for each actual individual, x, the proposition that x is a child of Wittgenstein (which we'll formalize Cx) is necessarily equivalent to $\bot$, and so identical to it. That is to say, $\forall_e x.(Cx =_t (\lambda y.\bot)x)$. So by the principle of Functionality $C =_{e \to t} \lambda x.\bot$—the property of being a child of Wittgenstein is identical to the inconsistent property. On the other hand, these properties cannot be identical since, by assumption, the former is possibly instantiated, albeit not by any actually existing individual, whereas the latter is not.

Although we relied above on a particular theory relating propositional individuation to metaphysical necessity, and a thesis about the necessity of origins, there is a deeper connection between the Functionality principle and the possibility of new individuals. In C, the Barcan formula–a principle often glossed as saying there can't be any "new" individuals—is equivalent to the Functionality principle. See Exercise 8.6 in Section 8.2.

Remark 6.4 (Modalized functionality)**.** There is a weakening of Functionality which states that *necessarily* cofunctional properties, relations, etc., are identical (see Modalized Functionality discussed in Section 8.2). This weakening of Functionality is not subject to the above sort of counterexample, and it also can fill the role of closing the gap between theories of granularity at different types. For instance, from the necessity of Propositional

Booleanism, we can prove that Boolean equivalent properties are necessarily cofunctional, allowing us to infer Property Booleanism.

Functional Plenitude is also incompatible with certain metaphysical views. Consider, for instance, the idea that propositions can be *about* individuals, or that they are built out of those individuals. Just as we suggested that the proposition that *Barcan is a logician* is about Barcan, we might more generally posit that for any property X, the proposition that Barcan is X is about Barcan:

$\forall_{e\to t}X\forall_e y.y \blacktriangleright Xy$

But according to Functional Plenitude, there is an $e \to t$ property, X, that maps Barcan to the proposition that Prior likes blue cheese, and every other individual to the proposition that snow is white. Intuitively the proposition that Prior likes blue cheese is about Prior, not Barcan. However, the proposition that Prior likes blue cheese just is the proposition that Barcan is X, and so by our principle this proposition is about Barcan. This threatens to trivialize the notion of aboutness. There was nothing special about this proposition and individual—parallel reasoning using Functional Plenitude can be used to show that any proposition is about any individual.

Actually, it turns out the naïve principle we stated above has this consequence already in H, without the assumption of Functional Plenitude. The reason is that we have been working in the full λ-language that permits *vacuous* λ-abstraction: for any proposition p there is the vacuous property $\lambda x.p$, or $K^{te}p$, of *being such that p*.

Exercise 6.16. *Let p be any proposition, and x any individual. Prove* $x \blacktriangleright p$ *from* $\forall_{e\to t}X.\ a \blacktriangleright Xa$.

Philosophers interested in such notions often work in a λ-language in which vacuous λ-abstraction is banned (explored in part III of the book).[11] However, we can formulate a version of our aboutness principle that does not trivialize in the background theory of H by simply restricting to non-vacuous properties

$\forall_{e\to t}X\forall_e y(\forall_t p(X \neq_{e\to t} \lambda x.p) \to y \blacktriangleright Xy)$

Exercise 6.17. *Prove* $\forall_t p\forall_e y.y \blacktriangleright p$ *from our revised principle of aboutness and Functional Plenitude and the assumption that there are two individuals.*[iii]

A final class of functionality principles to consider are inspired by the axiom of choice in set-theory. This principle was originally formulated by Zermelo as a principle governing sets. In his original formulation it states that for every set of pairwise disjoint non-empty sets, T, there is another set which contains exactly one element in common with each element of T.[12] The axiom of choice can be reformulated in terms of functions. Consider a relation r—now identified with a set of ordered pairs—between two sets A and B—so $r \subseteq A \times B$. If for each element $a \in A$ there is at least one element $b \in B$ such that $(a, b) \in r$–that is, r is a *serial* relation–then we can find a *functional* subrelation of r that maps A to B: a subset f of r such that for each $a \in A$ there is a unique $b \in B$ such that $(a, b) \in f$.[13]

So we might posit an analogous functionality principle governing properties and relations. A binary relation $R : e \to e \to t$ is serial when $\forall_e x\exists_e y.Rxy$. That is, we just relax the uniqueness requirement from our definition of a functional relation. Like our notion Functional there is an analogue for higher-arity relations:

$\text{Serial}_{\overline{\sigma}\tau} := \lambda X.\forall_{\overline{\sigma}}\overline{y}\exists_\tau z(X\overline{y}z)$

One choice-inspired principle is a straightforward strengthening of Functional Plenitude:

Functional Choice $\forall_{\overline{\sigma}\to\tau\to t}X(\text{Serial}\,X \to \exists_{\overline{\sigma}\to\tau}Y.\forall\bar{y}X\bar{y}(Y\bar{y}))$

Functional Choice is stronger than Functional Plenitude, since we have simply enlarged the sorts of binary relations that correspond to unary properties from functional to merely serial relations.

Comprehension Check 6.1. *Prove Functional Plenitude from Functional Choice.*

It follows that Functional Choice has the same sorts of substantial metaphysical implications as Functional Plenitude. It follows, for instance, that there's a property which, when applied to John, yields the proposition that the moon is made of blue cheese, and when applied to Mary yields the proposition that the Principia Mathematica was coauthored.

It might be tempting to think that Functional Choice (and thus also Functional Plenitude) enjoys a degree of mathematical orthodoxy, since the set-theoretic axiom of choice is almost universally taken for granted among mathematicians. But this sort analogizing is highly suspect. In Section 0.5 we were careful to distinguish higher-order generalizations from first-order quantification over abstract objects, likes sets and functions. We have no good reason to assume that they behave in the same way. More importantly, there are generally many different and non-equivalent ways to take a principle of set-theory and turn it into a principle of higher-order logic that is 'inspired' by it. Actually, the most flatfooted analogy of the claim that every serial relation has a functional subrelation is the following:

Relational Choice $\forall_{\overline{\sigma}\to\tau\to t}X(\text{Serial}\,X \to \exists_{\overline{\sigma}\to\tau\to t}Y(\text{Functional}_{\overline{\sigma}\to\tau\to t}\,Y \wedge \forall\bar{x}y.(Y\bar{x}y \to X\bar{x}y)))$

This is the form that choice principles took in early work on logical foundations conducted in higher-order logic.[14]

Thus, for instance, given a binary serial relation $R : e \to e \to t$, one can find another binary functional relation $S : e \to e \to t$ that is a subrelation of R. The existence of this other relation does not imply the existence of any of the spurious $e \to t$ *properties* discussed earlier, unless we also assume Functional Plenitude. In fact, Functional Choice can be more transparently analysed as the conjunction of Relational Choice and Functional Plenitude.

Comprehension Check 6.2. *Prove that Functional Choice is equivalent to the conjunction of Relational Choice and Functional Plenitude.*

Relational Choice has a more direct connection to mathematical practice. For instance, one only needs Relational Choice to prove the set-theoretic axiom of choice, avoiding the need to posit any of the spurious properties posited by Functional Plenitude/Choice. Let's see how one can get set-theoretic choice from Relational Choice. This argument will require a little familiarity with set-theory—the reader who wishes to skip ahead may do so. One can formulate the axiom of set-theoretic choice in the signature of higher-order logic with a binary predicate $\in$: $e \to e \to t$. The background theory of sets assumed by most mathematicians, ZF, can similarly be formulated in this signature and in this setting we can prove the set-theoretic axiom of choice from Relational Choice.[15] The reader doesn't really need to know much set-theory to follow this proof. The crucial thing they need to know is that, for any predicate in this language, F, and any set x, ZF posits a subset y of x that contains all and only the F members of x: $\forall_e x\exists_e y\forall_e z(z \in y \leftrightarrow z \in x \wedge Fz)$. This is called the axiom schema of *separation*.

Proposition 6.3. *The set-theoretic axiom of choice is derivable from the higher-order Relational Choice and the axioms of ZF.*

Suppose that T is a set of non-empty pairwise disjoint sets. Let $R := \lambda xy.(x \in T \wedge y \in x) \vee x \notin T$. Because each element of T is non-empty, R is serial. So, by Relational Choice, there is a functional subrelation, S, relating each element x of T to a unique element of x, y. Now by the axiom of separation there exists a set of these ys: $\{y \in \bigcup T \mid \exists_e x.Sxy\}$.[16] By the functionality of S, this set has exactly one element in common with any member of T.

The implication in the other direction does not hold: one cannot prove our higher-order Relational Choice principle from the set-theoretic choice principle. The reason is that some relations of type $e \to e \to t$ do not correspond to a set of ordered pairs: there cannot be a set of ordered pairs corresponding to every relation for reasons relating to Russell's paradox.

Comprehension Check 6.3. *Use Relational Choice to show there is a relation that relates each non-empty set to a unique member of that set.*

Another disanalogy between set-theory and higher-order logic is that sets, unlike properties and relations, are individuated extensionally. Many early-to-mid twentieth century mathematicians who made use of higher-order logic tended to follow Frege in assuming extensionality in their higher-order theorizing as well, at least as a simplifying assumption: mathematics is generally concerned with necessarily true or false propositions of which there might only be two. (It was common to take extensionality to be true as a matter of understanding the higher-order quantifiers as restricted to in some way to exclude intensionality—one such strategy is outlined in Whitehead and Russell, (1910), Introduction III.2; we discuss a related more general proposal for doing this in Remark 16.5.[17]) Under the assumption of Extensionalism there isn't an important difference between Relational and Functional Choice, and so the higher-order mathematicians working under the assumption of Extensionalism had no need to distinguish them. It would be an error to mistake an endorsement of Functional Choice, under the simplifying assumption of Extensionalism, as an endorsement of its implications outside of that simplifying assumption.

Exercise 6.18. *Assume Extensionalism and Relational Choice and prove the $e \to t$ version of Functional Choice: for every serial relation $R : e \to t \to t$, there is an $e \to t$ property F such that $Rx(Fx)$ for all individuals x.*

The reader should convince themselves that sort of argument extends to other relational types.

Remark 6.5 (Well-ordering principles)**.** The axiom of choice was originally introduced by Zermelo as an axiom from which to prove Cantor's well-ordering principle: that for every set there is a relation (considered as a set of ordered pairs) that totally orders the set in such a way that every non-empty subset of the set has a minimal element under the ordering. The set-theoretic axiom of choice is in fact equivalent to the well-ordering principle. Relational Choice is similarly equivalent to a higher-order analogue of the well-ordering principle: that a given type, σ, can be well-ordered by a $\sigma \to \sigma \to t$ relation. However, there doesn't appear to be any 'functional' analogue of the well-ordering principle, standing to the well-ordering principle as Functional Choice stands to Relational Choice.

Endnotes

1. See, for instance, Tarski (1936a), Henkin (1950).
2. The interaction of higher-order identities and attitude reports are discussed in an explicitly higher-order context in Bacon and Russell (2019), Caie et al. (2020) and Yli-Vakkuri and Hawthorne (2021).

3. Some have disputed the intersubstitivity of logical equivalents in the antecedent position of a counterfactual in relation to counterlogicals ('if the law of non-contradiction had failed then . . . ' and other puzzles involving disjunctive antecedents; see Fine (2012a,b)). But these reasons typically do not extend to the consequent position of a counterfactual.
4. The fact we are appealing to here is this: where C and D are propositional formulas, an equation $C = D$ is a theorem of the theory of Boolean algebras iff $C \leftrightarrow D$ is a theorem of propositional logic. The proof of this would take us too far afield, but here is a sketch of the (harder) completeness direction for readers with some background on the theory of Boolean algebras (the reader may consult Givant and Halmos (2010) for an introduction to Boolean algebras). Suppose that an identity $C = D$ is not derivable from these equations, so there is an interpretation of C and D in a Boolean algebra $\mathbf{B}$ (the Lindenbaum-Tarski algebra) in which $C \neq D$, and thus $(C \wedge \neg D) \vee (\neg C \wedge D) \neq \bot$. It follows, by the ultrafilter extension lemma, that there is a homomorphism $h : \mathbf{B} \to 2$ such that $h((C \wedge \neg D) \vee (\neg C \wedge D)) = 1$. h determines a propositional valuation, v such that $v((C \wedge \neg D) \vee (\neg C \wedge D)) = 1$, or equivalently, $v(C \leftrightarrow D) = 0$, establishing that $C \leftrightarrow D$ is not a theorem of propositional logic.
5. See Fine (2012c). Some grounding theorists do not use a connective, but a relation between first-order propositions, or sometimes other first-order entities as well. The connective formalism will be our focus since it directly implies things about granularity at type t.
6. Here $\bigvee_{A_i:\sigma} A =_\sigma A_i$ is a disjunction with a disjunct for each term A_i from the sequence $A_1, \ldots, A_n$ that has type σ, and is $\bot$ if there is no term of that type in the sequence.
7. Dorr (2016), Bacon (2020).
8. One either is taken as primitive, the other can be defined from it, since the identities $\top =_t \neg\bot$ and $\bot =_t \neg\top$ are theorems of the theory.
9. See, for example, Fine (2012c).
10. Establishing that C is indeed stronger than H requires some model theory. It is not too hard to find a model of H which, say, the law of idempotence is false using the model theory in the final part of the book.
11. See for instance Hodes (2015), Dorr (2016), Goodman (2018b).
12. See, e.g. axiom VI in Zermelo (1908).
13. Let T be $\{r_a \mid a \in A\}$ where $r_a = \{(a, b) \mid (a, b) \in r\}$: clearly the r_a are pairwise disjoint, and the seriality of r ensures they're non-empty. By the axiom of choice, there is a set, f, which contains exactly one element of r_a for each $a \in A$, and it's easily verified that f is a functional subrelation of r. In the other direction, if T is a set of non-empty disjoint sets, the relation that holds between each set belonging to T and its members is serial and so the range of the functional subrelation gives us the required choice set.
14. It is stated in essentially this form in Hilbert and Ackermann's textbook on logic, an early record of the higher-order thinking among mathematicians. They explicitly introduce it by analogy with (but distinguish it from) the choice principle from set-theory. See Hilbert and Ackermann (1928), p. 130, p. 156.
15. See, for instance Levy (2012)) Section 1.5. Here we understand the schemas of separation and replacement to include any formula of the higher-order language in this signature.
16. We here appeal to the union of T, $\bigcup T$, containing all and only elements of elements of T. The axioms of ZF ensure that this set exists.
17. See, for example, the justification of extensionality in the context of propositional logic in Hilbert and Ackermann (1928), p. 5.

Hints for exercises

[i] **Hint:** You should use the K combinator, and appeal to the fact that $\exists_\sigma x.x =_\sigma x$ is a theorem of H.

[ii] **Hint:** It's an induction on type complexity.

[iii] **Hint:** For any given p and y, consider the functional relation $\lambda x \lambda q.((q = p \wedge x = y) \vee (q = \neg p \wedge x \neq y)$ and apply Functional Plenitude.

7

Application

Modal logicism

This chapter is devoted to an application of higher-order logic in metaphysics. In this chapter, we explore issues in the philosophy of modality from the higher-order perspective. The philosophy of modality concerns the study of modal operators corresponding to expressions like 'it is possible that' and 'it is necessary that'. When one is theorizing about a particular modality, such as metaphysical necessity, or physical necessity, a typed language containing operator constants corresponding to the modality in question—such as the signature of propositional modal logic (Example 1.2)—is typically sufficient. However, it is quite often the case that we want to compare modalities, and quantify over them: consider, for instance, the assertion that metaphysical necessity is the *broadest* kind of necessity.[1] This appears to be a comparative claim—that metaphysical modality is broader than any other kind of modality—and requires us to quantify into operator position. So higher-order logic is an extremely natural framework for formulating these questions. This chapter examines some central questions in modal metaphysics from the perspective of higher-order logic. We will begin by analysing key notions from the philosophy of modality within the higher-order logic C (Classicism) introduced in Section 6.4, and will explore the thesis of 'Modal Logicism': the idea that important notions in the philosophy of modality—including the notions of being a necessity operator, of one operator being broader than another, of necessity 'in the highest degree', of an intensional operator, of a possible world, and so on—can be defined reductively in logical terms. Like the mathematical logicists, we take logic to include all of higher-order logic.

The approach of this chapter is to take a particular topic in the philosophy of higher-order modal logic and work through it in detail. However, this is a rich and fast growing area. There are many other issues in the philosophy of modality on which the framework of higher-order logic bears. A brief survey of other strands of research in higher-order modal logic may be found at the end of the next chapter in Section 8.4 .

Unless otherwise stated, Classicism will be assumed throughout this chapter. As explained in Section 6.4, it will be convenient to assume that we are working in a language with only relational types, i.e. e, t, and types of the form $\sigma \to \tau$ where both σ and τ are relational types but $\tau \neq e$. We will give a different formulation of Classicism for the full type system in Section 8.1.

DOI: 10.4324/9781003039181-10

7.1 Modal logicism

Let us begin by reviewing some of the key notions and questions of modal metaphysics. Our interest is in the study of *modality*: a modality is an operator that states that something about the world is necessary or possible. A modality will have two forms—necessity and possibility—which are duals of one another (the dual of $X : t \to t$ is $\lambda p.\neg(X(\neg p))$).[2] But for convenience we will generally pick out modalities by their necessity form. Here are some examples:

Aletheic necessity: what is true.

Practical necessity: what has to be true given the relevant practical constraints.

Legal necessity: what is required by the relevant laws.

Physically necessity: what is necessary given the laws of physics.

Metaphysical necessity: (contentious, but we will return to this below).

The philosophy of modality includes both the study of particular modalities, as well as the general relationships between them. The most basic relationship between two modalities is the relation of one modality being broader than another. Roughly speaking, a modality is broader than another when it counts more things as being possible. Here are some examples:

It's not practically possible for me to outrun a bicycle, although it is physically possible. It is not physically possible to run faster than the speed of light, although it is metaphysically possible, etc.

Here we say that the physical modalities are broader than the practical, and the metaphysical modalities are broader than the physical. So another key notion to the metaphysics of modality is this relationship of one modality being broader than another, or conversely, one modality being a restriction of the other.

Metaphysical necessity, of course, plays a particularly central role in modal metaphysics. The notion, as it appears in contemporary philosophy, traces back to Kripke's lectures 'Naming and Necessity'. Kripke is often thought to have introduced the notion by way of example. For instance, in Kripke's sense of *could* I could not have had parents other than the ones I in fact have, a table could not have been made from substantially different material than the material it is in fact made of, and there could not have been unicorns.[3] These distinguish Kripke's sense of could, for instance, from various idealized notions of epistemic possibility; sometimes people do not know who their real parents are, or what a given table is made out of.

Kripke also states that what is metaphysically necessary is what is 'necessary in the highest degree—whatever that means'.[4] This latter claim, recast in our terminology, is that metaphysical modality is the *broadest kind of modality*. This means that the epistemic possibilities alluded to above are not genuine possibilities for Kripke. Epistemic modalities are presumably subject to the same sorts of problems that arise when you naïvely treat attitude verbs as propositional operators, as discussed in Chapter 6.

There is a terminological problem here. Some theorists, unlike Kripke, recognize epistemic modalities as genuine propositional modalities so that the broadest kind of modality, whatever that might be, must at least count the epistemic possibilities where I have different

parents as broadly possible. Other theorists who accept propositional notions of logical necessity, tense operators, or propositional determinacy operators for theorizing about vagueness, will similarly posit modalities that outstrip what is possible according to Kripke's worldview. In this case, should we take 'metaphysical necessity' to refer to the broadest kind of necessity, and take Kripke to be making some substantive (and false) assertions about it? Or should we take 'metaphysical necessity' to be introduced by the examples so that it refers to something satisfying particular theses about the essentiality of parenthood and origins, but which is narrower than other modalities out there. Some philosophers working in the tradition of higher-order logic have maintained that there are many senses of 'could' which validate Kripke's examples, and that his remarks are too thin to single out any particular notion uniquely. According to these theorists, if we are to mean anything determinate by 'metaphysical necessity' we are to mean necessary in the broadest sense: Peter Fritz, for instance, writes that 'it is one of the most central theoretical roles of metaphysical necessity that it is the strongest of the relevant kinds of necessity.'[5] But others have resisted this skepticism. After all, there are many notions of philosophical importance—including notions such as 'epistemic justification' and 'moral obligation'—about which there is very little we can say to single it out uniquely, yet we seem to succeed in introducing these notions to newcomers through examples and through their role in philosophical theorizing.[6]

We may, at least for now, sidestep this issue. Whatever we take 'metaphysical necessity' to refer to, the notion of a broadest necessity is unambiguous, under certain conditions, and clearly an interesting notion in its own right. It is identical to the central theoretical modality in Kripke's worldview, but it is arguably just as central for those who acknowledge wider kinds of possibility than Kripke does. So we may focus our efforts on elucidating the notion of *broadest necessity* or *necessity in the highest degree*. Some crucial questions are, then:

Is there a broadest kind of modality?

If there is, is it unique?

What is the logic of the broadest necessity?

A negative answer to the first question could arise under a couple of conditions. For instance, there might exist, for each modality, a strictly broader modality precluding the existence of a maximally broad modality. Or, due to the existence of incomparabilities between modalities under the broadness ordering, there might be multiple maximally broad necessities, none of which are as broad as any other, thus precluding a broad*est* necessity. And even if there is a broadest necessity, the second question could be negative if there were several broadest necessities all as broad as each other.

Contemporary discussions of metaphysical necessity are intimately bound up with the notion of a *possible world*. The notion traces back to Leibniz, and plays a critical role in philosophical logic, metaphysics and semantics. Formally, a possible world provides us with a model theory for modal logic—it can be used to characterize the valid inferences of a given logic. But many philosophers have taken the possible worlds formalism more seriously, accepting the hypothesis that possible worlds are real and that modal reality itself has the structure of some particular Kripke model.[7] Indeed, there are few substantive metaphysical commitments that are this widely held. But many questions about the nature of possible worlds are highly contentious, and it is somewhat unclear what the thesis that 'possible worlds are real' means that is independent of the question of their nature.

What are possible worlds? Some have identified them with concrete objects, others with abstract individuals. Some have denied that they are individuals at all, and replace first-order world talk with quantification in sentence position.

Is it possible to give a reductive analysis of modal notions in non-modal terms?

What does it mean to take possible worlds 'metaphysically seriously'? What would it mean for modal reality to have the structure of a Kripke frame, and what would it mean for it to fail to?

The contention of this chapter is that higher-order logic is a useful tool for the modal metaphysician. Many of the central notions and key questions identified above may be formalized in higher-order language, and may be derived or tested for consistency using model-theoretic methods. The most starkest form in which higher-order logic could bear on modal metaphysics would be if it subsumed it: if the central notions of modal metaphysics could literally be reduced to logic, and the important questions could be settled by purely logical principles. We will call this thesis:

Modal Logicism The central concepts in modal metaphysics—including (i) the notion of being a necessity, (ii) the relation of one necessity being broader than another, (iii) the notion of 'necessity in the highest degree', and (iv) the notion of a possible world—can be reductively defined in purely logical terms from the higher-order quantifiers and truth-functional connectives.

This is the thesis I will investigate in this chapter, against the background of a particular higher-order logic: C. Of course, Modal Logicism is a highly contentious claim, even granting the background logic of Classicism. I will not attempt to anticipate or adjudicate objections to the analyses of modal notions provided—I trust the reader to apply their own judgement here—but I hope that the exercise will both make the thesis of modal logicism seem more plausible than it might have initially, and to inspire the reader to investigate alternative frameworks. Some of these are discussed in the further reading section.

One preliminary question that we must confront before we start concerns which operator expressions in philosophical English express genuine notions of necessity and which do not. Appearances can sometimes be misleading—consider the following list of operator expressions:

it's logically necessary that, *it's determinate that*, *it's a priori knowable that*, *it's always the case that*.

While the first two expressions appear to be operator expressions, they are often used in philosophy as a shorthand way of talking about a property of sentences, not propositions. The operator of logical necessity is sometimes used as a stand-in for the linguistic property of being a logical truth, and according to semantic theories of vagueness, the determinacy operator is a stand-in for a special property of sentences. When used this way, it would be more perspicuous to abandon the operator expression, and instead to theorize in terms of an $e \to t$ predicate of sentences, using explicit quotation marks instead of that-clauses. The status of *a priori* knowability as a necessity is similarly contentious due to its sensitivity to modes of presentation; some authors also take it to be a property of sentences, but even those that treat it as an operator expression allow it to be context sensitive in a way that allows it to depend on modes of presentation. The final expression is treated as a kind of necessity

by some authors, most prominently Arthur Prior. But some philosophers and linguists have argued that it is not a propositional operator at all: sentences contain a hidden time variable, and what might at first appear to be a propositional operator is in fact quantificational. The moral to take away from this is that the classification of various expressions as necessities is subtle, and we should be careful not to conflate propositional necessities with linguistic notions or quantificational expressions.

7.2 Necessity

The notion of necessity (and thus also of possibility) that we will attempt to analyse in this chapter is a fairly liberal one. Logically complex necessities can count as necessities by our lights. So, for instance, the following are necessities in our sense:

> *it is legally necessary that it is practically necessary that*, *it is legally and practically necessary that*

These are obtained by composing and conjoining (using the operations B^{ttt} and $\wedge_{t\to t}$ respectively) two operators that we already take to be necessities. Generally it is sufficient that an operator be logically well-behaved, in a sense to be spelled out, for it to count as a necessity by our lights. There are some philosophers who will no doubt posit, in addition to this liberal notion of necessity, a more demanding one according to which the above do not count, in virtue of being 'gruesome'.[8] I am myself skeptical that there is a single clear target for analysis here. But at any rate, we can all agree that there is the more liberal notion, and it plays an important role in modal metaphysics; it is at least a starting point from which one could begin to articulate a more demanding notion, if you were so inclined.

Modal logic is the principal formalism used in the study of necessity, under even the most liberal conceptions of what a necessity is. The language of propositional modal logic is the typed language described in Example 1.2, which contains in addition to the truth-functional connectives, infinitely many constants of type t—the 'propositional letters'—and a constant $\Box$ of type $t \to t$. The modal logic $\mathsf{K}^{\Box}$ is the smallest set of sentences of this signature that contains (i) all propositional tautologies, (ii) contains $\Box(A \to B) \to \Box A \to \Box B$ for every sentence A and B (the K schema), (iii) contains B whenever it contains sentences A and $A \to B$ (is closed under modus ponens), (iv) contains $\Box A$ whenever it contains the sentence A (is closed under necessitation). A logic, $\mathbf{L}$, in this signature, is normal when it contains $\mathsf{K}^{\Box}$.

At a first pass, we might say that an interpreted operator expression, '$\Box$', expresses a kind of necessity if it is normal: all the sentences of the theory $\mathsf{K}^{\Box}$ are true. Some questions we will be concerned with in this section are:

> Is there a propositional analogue of a normal modal logic?
>
> Is there an analogous definition of a necessity for operators?

We will see that there are notions answering these descriptions, and they can be reductively defined in purely logical terms. (Again, it should be emphasized here that our target is not a very demanding notion of necessity; it corresponds merely to a kind of logical well-behavedness. We leave open the question of whether there are more demanding notions of necessity, and the question of their analysis if so.)

Given a candidate necessity operator $X : t \to t$, we can define a propositional property of 'belonging to the smallest normal modal logic for X', by analogy with the property of sentences identified above. We define a property of propositions $\mathrm{SmallestKTheory}\, X : t \to t$ (the propositional analogue of $\mathsf{K}^{\Box}$) as the smallest collection of propositions that contains (i) the propositional tautologies, (ii) contains $\forall_t pq(X(p \to q) \to Xp \to Xq)$, (iii) contains the proposition q whenever it contains p and $p \to q$, and (iv) contains Xp whenever it contains p. Then we may define a weak necessity as one such that the propositions in $\mathrm{SmallestKTheory}\, X$ are all true. (Note that talk of 'collections' of propositions, going forward, will be simulated by quantification over operators.)

One might have thought that (i) is too weak in the richer signature of higher-order logic, since a *real* necessity should not only apply to tautologies but also to logical truths involving higher-order quantifiers and identity. However, C ensures that all things expressed by a theorem of C are identical—propositional tautologies and distinctively higher-order theorems alike. So condition (i) may be formulated in terms of a single tautology $\top$.

Remark 7.1. It is at this point that the present account of necessity does not obviously extend to other theories of propositional granularity in which distinct theorems of logic can express different propositions. In fact, it is possible to characterize 'being a tautological proposition' much as we characterize the propositional analogue of 'being a normal modal logic' below—by talking about the smallest collection of propositions containing all instances of the axioms and closed under modus ponens. However, unlike propositional logic and propositional modal logic, higher-order logic is not finitely axiomatizable due to the infinity of primitive quantifiers, so a parallel characterization of the propositions corresponding to theorems of higher-order logic is not possible.[9]

Remark 7.2. Note that (ii) says that the universally quantified statement $\forall_t pq(X(p \to q) \to Xp \to Xq)$ is in our propositional analogue of the modal logic $\mathsf{K}^{\Box}$, whereas the strict analogue of the K principle would simply require the modal logic to contain each instance of the K principle. Gor any propositions p and q, the modal logic should contain the proposition $X(p \to q) \to Xp \to Xq$, but needn't contain the universal generalization. Following this second way of cashing out (ii) would give us an apparently weaker notion of a modal logic. Suppose there could have been (in some sense of *could have been*) propositions which do not actually exist. If Z is a property of propositions satisfying the stronger notion of modal logic, then it necessarily must apply to all instances of K, including those involving the new propositions. If Z only met the weaker condition it need only contain instances of K that actually exist.[10]

Our goal is to define the propositional analogue of the modal logic $\mathsf{K}^{\Box}$ for the operator X. We will do this by introducing a relation between operators, $\mathrm{KTheory}\, XZ$ which says that Zs extension is a collection of propositions that is closed under the four closure conditions mentioned above: it contains tautologies, the claim that X is closed under modus ponens, and Z itself is closed under modus ponens and the rule of necessitation for X. To that end, let us introduce the following definitions:

$$\mathrm{PC} := \lambda Z.Z\top$$

$\mathrm{PC}\, Z$ means that Z contains all propositional tautologies (here we are appealing to our above observation that they are all identical in C).

$$\mathrm{MP} := \lambda Z.\forall_t pq(Zp \wedge Z(p \to q) \to Zq)$$

MP Z states that Z is a collection of propositions closed under modus ponens:

$$\mathrm{N} := \lambda X.\lambda Z.\forall_t p(Zp \to Z(Xp))$$

N XZ states that Z is a collection of propositions closed under the rule of necessitation for X.

$$\mathrm{K} := \lambda X.\lambda Z.Z(\mathrm{MP}\ X)$$

K XZ states that Z is a collection of propositions that contains the proposition $\forall_t pq(X(p \to q) \to Xp \to Xq)$.

Definition 7.1 (KTheory)**.** *We define the propositional analogue of a theory containing the logic* $\mathsf{K}^{\Box}$ *as follows:*

$$\mathrm{KTheory} := \lambda XZ.(\mathrm{PC}\ Z \wedge \mathrm{MP}\ Z \wedge \mathrm{N}\ XZ \wedge \mathrm{K}\ XZ)$$

The modal logic $\mathsf{K}^{\Box}$ *is the smallest theory closed under the rules we have specified, so we can define the analogue of* $\mathsf{K}^{\Box}$ *for the operator X as the propositions that belong to every* KTheory.

$$\mathrm{SmallestKTheory} := \lambda X.\lambda p.\forall_{t \to t} Z(\mathrm{KTheory}\ XZ \to Zp)$$

The reader should remember that, strictly speaking, theories are sets of sentences. The name KTheory is chosen simply to make the analogy salient.

This brings us to our first definition of a *weak* necessity.

Definition 7.2 (Weak Necessity)**.** *An operator X is a weak necessity if and only if every proposition in the propositional analogue of* $\mathsf{K}^{\Box}$ *for X is true. That is to say, if p is in every* KTheory *for X then p is true:*

$$\mathrm{WNec} := \lambda X.\forall_t p(\mathrm{SmallestKTheory}\ Xp \to p)$$

Exercise 7.1. *Prove that if X is a weak necessity, then* $\forall_t pq(X(p \to q) \to Xp \to Xq)$ *and* $X\top$.

The notion of a weak necessity has a special relationship to the study of normal modal logics. When you consider an interpreted propositional modal language with the modal operation interpreted by a weak necessity, all the theorems of the weakest normal modal logic K will be true.

While the notion of a weak necessity is general enough to account for the full variety of applications of modal logic in philosophy, including applications in epistemic logic and the natural language semantics, I take it that it is too weak to properly capture the metaphysical notion of *necessity* that we are trying to capture. To see why it is too weak, suppose that Simon is a logically perfect agent, and that he knows this, knows that he knows this, knows that he knows that he knows this, and so on. It may be shown (Exercise 7.3) that the operator *Simon knows that* is a weak necessity. However, Simon is only contingently logically perfect: it is physically possible (say) that he knows that it's raining and that if it's raining he'll get wet, and that he never realises that he'll get wet. It is physically possible that *Simon knows that* fails to 'satisfy the K axiom', and so it is only contingently logically well-behaved.

The problem is that a weak necessity is necessarily logically well-behaved by its *own* lights, but not by the lights of *other* necessity operators. A necessity, in the sense relevant to metaphysicians, must be one that it is *necessarily* a weak necessity, in every weak sense of *necessarily*.

Definition 7.3 (Necessity)**.** *An operator X is a necessity when, for every weak necessity, it is necessarily a weak necessity.*

$$\mathrm{Nec} := \lambda X.\forall_{t\to t} Y(\mathrm{WNec}\ Y \to\ Y(\mathrm{WNec}\ X))$$

How do we go about proving that an operator, X, is a weak necessity or a necessity? Let's start with weak necessities. We need to show that if p belongs to every $\mathrm{KTheory}$ for X then p is true. The general strategy, then, is to find a $\mathrm{KTheory}$, Z, that only applies to truths. The property of being true itself, $\lambda p.p$, seems like it would be the natural choice, except truth does not generally preserve necessitation (i.e. $\neg((\mathrm{N}\ X)\lambda p.p)$), and so is not itself a $\mathrm{KTheory}$: while *snow is white* is true, *it's metaphysically necessary that snow is white* is not.

Exercise 7.2. *While the truths are not closed under X-necessitation for arbitrary necessities, in the special case where the necessity X is truth then the truths* are *closed under truth-necessitation. We can use this fact to prove that truth is a weak necessity.*

a. Show that truth, $\lambda p.p$ is a $\mathrm{KTheory}$ *for $\lambda p.p$: it has the four closure conditions* $\mathrm{PC}, \mathrm{K}(\lambda p.p), \mathrm{MP}$ *and* $\mathrm{N}(\lambda p.p)$

b. Conclude that $\lambda p.p$ is a weak necessity.

c. Using the above prove that every necessity is a weak necessity: $\mathrm{Nec}\ X \to \mathrm{WNec}\ X$.

The example we considered above involving knowledge gives us a hint as to what a more appropriate property would be: the property of being X-necessary 'at all orders'—being true, being X-necessary, X-necessarily X-necessary, X-necessarily X-necessarily X-necessary, and so on. The claim that p is X-necessary at all orders can be stated by saying that if q belongs to every collection of propositions containing p and closed under prefixing X to a proposition already in the collection, then q is true (the reader may wish to revisit Section 5.3 on inductive definitions):

$$X^* := \lambda p.\forall_t q(\forall_{t\to t} Y(\,Yp \wedge \forall_t r(\,Yr \to\ Y(Xr)) \to\ Yq) \to q)$$

Proposition 7.1. *X is a weak necessity if and only if $X^*\top$ and $X^*\forall pq(X(p \to q) \to Xp \to Xq)$.*

Proof. We prove the left-to-right direction and leave the right-to-left as an exercise. Suppose that X is a weak necessity. We want to show that $X^*\top$ and $X^*\forall pq.(X(p \to q) \to Xp \to Xq)$.

To show $X^*\top$ we must establish the truth of every proposition q belonging to every 'collection' of propositions containing $\top$ and closed under applying X. Suppose that q is in every collection $Y : t \to t$ applying to $\top$ and closed under the operation X. But the smallest K theory for X contains $\top$ and is closed under X. And since X is a weak necessity, every proposition in its smallest K theory is true. So q is true. Thus $X^*\top$. A completely parallel argument establishes that $X^*\forall pq.(X(p \to q) \to Xp \to Xq)$. □

The following exercise is somewhat more tricky and can be skipped by the reader wishing to proceed with the discussion of necessities.

Exercise 7.3. *In this exercise, we will prove the right-to-left direction of Proposition 7.1. Suppose that $X^*\top$ and $X^*\forall pq.(X(p \to q) \to Xp \to Xq)$ (i.e.* $\mathrm{PC}\ X^*$ *and* $\mathrm{K}\ XX^*$*).*

a. Prove that X^* is closed under modus ponens; i.e. show that $\mathrm{MP}\ X^*$.

b. Prove that X^* is closed under necessitation for X, i.e show that $\mathrm{N}\ XX^*$.

c. Show that X^* only applies to truths; i.e. show that $\forall_t p(X^*p \to p)$.

d. Conclude that X^* is a K theory that only applies to truths. Explain why this means that X is a weak necessity.

In many cases—indeed in any case where the operator X is not just a weak necessity but a necessity—the property of being $\top$, $\lambda p.(p =_t \top)$, also is a KTheory for X. So this will be our primary way of showing that something is a weak necessity and then, ultimately, showing it is a necessity. Since $\lambda p.(p =_t \top)$ will crop up a lot, we will give it a special name:

$\Box_\top := \lambda p.(p =_t \top)$

It is possible to prove (see the exercise below) that $\mathrm{MP}\,\Box_\top$. $\mathrm{PC}\,\Box_\top$ is trivial, for it is β-equivalent to $\top =_t \top$.

Exercise 7.4. *Prove that* $\forall_t pq(\Box_\top(p \to q) \to \Box_\top p \to \Box_\top q)$.[i]

So all we need to do to prove that some operator X is a weak necessity is prove to $\mathrm{K}\,X\Box_\top$ and $\mathrm{N}\,X\Box_\top$. Not only can this strategy be used to show more operators are weak necessities, it gives us an alternative strategy for proving that truth is a necessity.

Exercise 7.5. *In this exercise we will show that the operators $\lambda r.\top$, $(\lambda r.r)$ and $\Box_\top$ are weak necessities:*

a. Prove that $\Box_\top \forall_t pq((\lambda r.\top)(p \to q) \to (\lambda r.\top)p \to (\lambda r.\top)q)$ and $\forall_t p(\Box_\top p \to \Box_\top((\lambda r.\top)p)$ (i.e. $\mathrm{K}(\lambda r.\top)\Box_\top$ and $\mathrm{N}(\lambda r.\top)\Box_\top$).

b. Prove that $\Box_\top \forall_t pq(\Box_\top(p \to q) \to \Box_\top p \to \Box_\top q)$ and $\forall_t p(\Box_\top p \to \Box_\top \Box_\top p)$ (i.e. $\mathrm{K}\,\Box_\top\Box_\top$ and $\mathrm{N}\,\Box_\top\Box_\top$).

c. Prove that $\Box_\top \forall_t pq((\lambda r.r)(p \to q) \to (\lambda r.r)p \to (\lambda r.r)q)$ and $\forall_t p(\Box_\top p \to \Box_\top(\lambda r.r)p)$ (i.e. $\mathrm{K}(\lambda r.r)\Box_\top$ and $\mathrm{N}(\lambda r.r)\Box_\top$).

d. Conclude that $\lambda r.\top$, $\lambda p.p$ and $\Box_\top$ are weak necessities.

Let us turn to the problem of showing that an operator is a necessity. Recall that C is closed under the Rule of Equivalence, the type t instance of which states that if it proves $A \leftrightarrow B$ it also proves $A =_t B$. This ensures that C is also closed under the rule of necessitation for $\Box_\top$:

If A is a theorem of C, so is $\Box_\top A$.

For if A is a theorem of C, so is $A \leftrightarrow \top$ by propositional logic. So, by the Rule of Equivalence, $A =_t \top$ is a theorem of C, which is β-equivalent to $\Box_\top A$.

It follows that if we have been able to prove that something is a weak necessity in that system, we can prove that the claim that it is a weak necessity is $\top$. Since every weak necessity applies to $\top$, it follows that every weak necessity applies to the claim that our operator is a weak necessity.

Exercise 7.6.

a. Drawing on the previous exercise, explain why we can infer that $(\mathrm{WNec}\,\lambda r.\top) = \top$, $(\mathrm{WNec}\,\lambda r.r) = \top$ *and* $(\mathrm{WNec}\,\Box_\top) = \top$.

b. Prove $\mathrm{Nec}\,\lambda r.\top$, $\mathrm{Nec}\,\lambda r.r$ *and* $\mathrm{Nec}\,\Box_\top$.

Putting these together we can obtain necessary and sufficient conditions for an operator to be a necessity analogous to the necessary and sufficient conditions we obtained from Proposition 7.1.

Proposition 7.2. *X is a necessity if and only if $\Box_\top(X\top)$ and $\Box_\top\forall_{\iota}pq(X(p \to q) \to Xp \to Xq)$*

Proof. Left-to-right: Suppose X is a necessity. We will show only $\Box_\top(X\top)$, since $\Box_\top\forall_{\iota}pq(X(p \to q) \to Xp \to Xq)$ is proven by a completely parallel argument. From the definition of WNec, one can derive $(\mathrm{WNec}\, X \to X\top) \leftrightarrow \top$ in C, so by the Rule of Equivalence $\Box_\top(\mathrm{WNec}\, X \to X\top)$. Since we have already established that $\Box_\top$ distributes over conditionals we can infer that $\Box_\top(\mathrm{WNec}\, X) \to \Box_\top(X\top)$. Finally if X is a necessity, then it is necessarily a weak necessity for every weak necessity. Since we have shown that $\Box_\top$ is a weak necessity, $\mathrm{Nec}\, X \to \Box_\top\, \mathrm{WNec}\, X$. So, $\mathrm{Nec}\, X \to \Box_\top(X\top)$.

Right-to-left: We want to show that X is a necessity, so we must show that if p belongs to every KTheory for X then p is true. It suffices to show that $\Box_\top$ is a KTheory, since it would then follow that $\Box_\top p$ and thus that p is true (since it is identical to $\top$, which is true). $\mathrm{PC}\,\Box_\top$ and $\mathrm{MP}\,\Box_\top$ have be shown previously, and the remaining two conditions, $\mathrm{K}\, X\Box_\top$ and $\mathrm{N}\, X\Box_\top$ follow from the assumptions. □

Exercise 7.7. *In this exercise, you will use proposition 7.2 to show that the composition of two necessities is a necessity. Suppose that* $\mathrm{Nec}\, X$ *and* $\mathrm{Nec}\, Y$ *and write* $X \circ Y$ *for* $(\lambda r.X(Yr))$.

a. Show that $\Box_\top((X \circ Y)\top)$

b. Show that $\Box_\top\forall_{\iota}pq((X \circ Y)(p \to q) \to (X \circ Y)p \to (X \circ Y)q)$.[ii]

c. Conclude that $(X \circ Y)$ *is a necessity.*

Exercise 7.8. *Show that the conjunction,* $X \wedge_{\iota\to\iota} Y$ *or* $\lambda p.(Xp \wedge Yp)$*, of two necessities, X and Y, is a necessity.*

7.3 Entailment

The concept of propositional entailment—that is, of one proposition being a necessary consequence of another—is another central notion that calls out for reductive definition. It is common to explain the idea of one proposition entailing another, as we have just done, in terms of strict implication: necessarily, if p then q. But what sort of necessity should be plugged into this definition? Some philosophers assume from the outset that there is a 'broadest' kind of necessity and simply use that notion. However, our goal is to reduce the notion of a broadest necessity to purely logical notions and to prove the existence of such a necessity from assumptions of a purely logical nature. And since the idea of one necessity being broader than another, and consequently the notion of a *broadest* necessity, is most naturally explained in terms of entailment we should not presuppose such a notion in the definition of entailment.

However, we don't actually need to make a choice about what sort of strict implication to use in defining entailment, since in higher-order logic we can quantify into operator position, and thus over all necessities at once. I put forward that if p entails q then q is a necessary consequence of p, for *every* notion of *necessary consequence*—it is a "universally strict implication". This claim can be motivated through examples. Paradigm examples of entailments include:

$p \wedge q$ entails p

$\forall_e xFx$ entails Fa

$a = b$ entails $Fa \to Fb$

In each case, we may prove that the corresponding material conditional is necessary for every necessity; for instance for the first point, we must prove $\forall_{t\to t}X(\mathrm{Nec}\, X \to X(p \wedge q \to p))$. I leave this as an exercise left to the reader.

Conversely, I put forward that if q is a necessary consequence of p for every sense of *necessary consequence*, then p entails q. This implication can be argued for directly, for if we are happy with the binary connective of propositional entailment, then we should be happy with the unary connective corresponding to zero-premise entailment (since it can be defined from the binary notion as *being entailed by* $\top$). This has as good a claim as any notion to satisfying our formal conditions for being a necessity operator, and so if $p \to q$ is necessary for every necessity, then $p \to q$ is entailed by $\top$. Now, given a standard law of entailment—that some premise entails $p \to q$ if and only if that premise along with p entails q, we can reason that since $\top$ entails $p \to q$, $\top \wedge p$ entails q. Given Booleanism this is just to say that p entails q, as required.

Exercise 7.9. *Letting $p \leq q$ represent propositional entailment prove that entailment is equivalent to universally strict implication—that is $(p \leq q) \leftrightarrow \forall_{t\to t}X(\mathrm{Nec}\, X \to X(p \to q))$—from the following assumptions.*

$$\mathrm{Nec}\, \lambda p.(\top \leq p)$$
$$p \wedge q \leq r \leftrightarrow p \leq (q \to r)$$
$$\forall X(\mathrm{Nec}\, X \to \forall_t pq(p \leq q \to X(p \to q)))$$

So here is a hypothesis:

Entailment is universally strict implication For p to entail q *just is* for q to be a necessary consequence of q for every necessity.

This identification explains why standard examples of entailments are also strict implications—entailment just is strict implication under every kind of necessity—as well as explaining the more general coincidence between entailment and strict implication we have just argued for. Writing $\leq_t$ for propositional entailment, this gives us the following definition $\leq_t := \lambda pq.\forall_{t\to t}X(\mathrm{Nec}\, X \to X(p \to q))$. Since Nec was defined in purely logical terms, $\leq_t$ is defined in purely logical terms.

Exercise 7.10. *There are systematic connections between the notion of propositional entailment the notion of necessity. For example, it is usually assumed that a necessity is closed under entailment:*

$$\forall_t p_1 \ldots p_n q \forall_{t\to t}X(\mathrm{Nec}\, X \wedge Xp_1 \wedge \ldots \wedge Xp_n \wedge (p_1 \wedge \ldots \wedge p_n \leq_t q) \to Xq))$$

Prove that this follows from our definitions of entailment and necessity.

The notion of propositional entailment discussed above is often thought to have analogues for properties and relations. For instance, given properties $F, G : e \to t$, we may say that F entails G when, necessarily every F is G, for every necessity. More generally:

Definition 7.4 (Entailment)**.** *Given a sequence of types $\overline{\sigma} = \sigma_1 \ldots \sigma_n$, and two relations $Y, Z : \overline{\sigma} \to t$ we say that Y entails Z iff for every Necessity X, it's X-necessary that for every $z_1 : \sigma_1, \ldots, z_n : \sigma_n$, if $Yz_1 \ldots z_n$ then $Zz_1 \ldots z_n$:*

$$\leq_{\overline{\sigma}\to t} := \lambda YZ.\forall_{t\to t}X.(\mathrm{Nec}\, X \to X(\forall_{\overline{\sigma}}\overline{z}(Y\overline{z} \to Z\overline{z})))$$

We earlier introduced the idea of one necessity being as broad as another. We can now analyse this in terms of our newly introduced notion of operator entailment

Definition 7.5 (Broadness)**.** *Given two necessities, X and Y of type $t \to t$, we say that X is as broad as Y iff X entails Y; in the present notation $X \leq_{t \to t} Y$.*

Exercise 7.11. *Show that the relation of being* as broad as *is a transitive and reflexive relation.*

Exercise 7.12. *Show that $\lambda p.\top$ is the narrowest necessity: it is entailed by every other necessity. (You have already shown it is a necessity.)*

We informally explained the notion of one necessity being broader than another in terms of it recognizing more possibilities. This is substantiated by our definition too:

Exercise 7.13. *Show that if X and Y are necessities and $X \leq_{t \to t} Y$, then for any p if $\neg Y \neg p$ then $\neg X \neg p$.*

As we are using the term, a modality is broader than another when it considers more things to be possible (and fewer things to be necessary): it acknowledges a broader class of possibilities (and thus a narrower class of necessities).[11]

Exercise 7.14. *Show, within* C*, that if X and Y are necessities, then X is* as broad as *Y if and only if Ys dual is* as broad as *Xs dual (the dual of X is $\lambda p.\neg(X(\neg p))$, and similarly for Y).*

We have considered a philosophical definition of entailment in terms of strict implication. But there is another notion of propositional entailment available. In the theory of Boolean algebras, it is quite common to introduce a partial order on an algebra by saying that for any two elements of the algebra a and b, $a \leq b$ iff $a \wedge b = a$. This idea of defining notions like containment and entailment in terms of identity traces back to Boole (1847). Since, given C, propositions (and indeed any relational type $\overline{\sigma} \to t$) form a Boolean algebra under the Boolean operations at that type ($\wedge_{\overline{\sigma} \to t}$, $\vee_{\overline{\sigma} \to t}$, etc.) we have another potential way of defining entailment:[12]

Definition 7.6 (Entailment (Alternate Definition))**.** *Given relations X and Y of type $\overline{\sigma} \to t$, we say that X entails Y iff the conjunction of X and Y is X:*

$$\leq_{\overline{\sigma} \to t} := \lambda XY.(X \wedge_{\overline{\sigma} \to t} Y =_{\overline{\sigma} \to t} X)$$

We will ultimately be able to show that the alternate relation of entailment is identical to the original. But to do that we will need to prove some more facts first.

The notion of entailment—both the official and alternate definition—is binary, relating a single premise to a conclusion. Yet it is often useful to have a notion that relates a *number* premises to a conclusion. When the premises are finite, it is possible to express such an entailment using the binary notion of entailment by conjoining the premises: $p_1, \ldots, p_n$ entail q iff $(p_1 \wedge \ldots \wedge p_n) \leq_t q$. But absent any principles guaranteeing the existence of infinite conjunctions, this strategy does not extend infinite premise sets. In this case, we can instead express the entailment by saying that every proposition entailing all of the premises entails the conclusion. Given principles ensuring the existence of infinite conjunctions, it may be shown that our definition coincides with the definition in terms of conjunctions. However, this definition makes sense even if the premises do not have a conjunction, so this will be our official definition of multipremise propositional entailment: it is a relation between an operator $X : t \to t$, whose extension represents a collection of propositions, and a conclusion

proposition. We can extend this notion of multipremise entailment to properties and relations in an obvious manner. We will use the symbol $\leq_{\sigma s}$ to express multipremise entailment between type σ entities, adding an 's' to the type subscript to indicate it relates a plurality of things of type σ.

Definition 7.7 (Multi-premise entailment)**.** *Let $X : t \to t$ and $p : t$. Multi-premise propositional entailment is defined as follows:*

$$\leq_{ts} := \lambda Xp.\forall_t r((\forall_t q(Xq \to r \leq_t q) \to r \leq p))$$

This definition extends to any relational type σ in an obvious way, by replacing X and p with variables of type $\sigma \to t$ and t, and the t subscripts in our definition with σ.

We can now express a distinction between necessities we were unable to express before. Recall from Exercise 7.10 that necessities are closed under finite entailments: if $p_1, \ldots, p_n$ entails q and $p_1, \ldots, p_n$ are necessary so is q. With the general notion of entailment in hand, we can ask whether a necessity is closed under arbitrary entailments. Our notion of a necessity, like the theory of normal modal operators on which it is based, does not require necessities to be closed under arbitrary entailments. Some putative examples of necessities do not appear to be closed under arbitrary entailments. One might think that having a chance of 1 is a necessity, but a number of chance 1 propositions can entail a proposition that does not have chance 1 (for instance, it could be that for each point on a dart board, the proposition stating that a dart did not land on that point has chance 1, but these together entail that the dart didn't land anywhere on the dartboard, which may have a chance less than 1). Or perhaps deontic necessity is like this. Suppose it is within your ability to save any finite number of lives, but not an infinite number of lives. So for each n, you ought to save more than n lives. But then the things you ought to do jointly entail you ought to save an infinite number of lives, which is not something you ought to do, simply because you cannot do it. On the other hand, many other notions—such as metaphysical necessity, tense operators, and so on—do appear to be closed under infinite consequence. We can now express this distinction as follows.

Definition 7.8 (Infinitely closed necessity)**.** *A necessity is infinitely closed if and only if it applies to any proposition that is entailed by the necessary propositions:*

$$\mathrm{Nec}_\infty := \lambda X(\mathrm{Nec}\, X \wedge \forall p(X \leq_{ts} q \to Xp))$$

7.4 Necessity in the highest degree

We now have the resources to formulate one of the central questions of modal metaphysics: is there a *broadest* necessity? And if there is, is it unique? The answer to the first question is not obvious: for all we presently know, it could be the case that for every necessity there is a strictly broader one. Or perhaps there are two (or more) necessities that are maximally broad, in the sense that there are no necessities broader than either of them, but neither is as broad as the other.

In fact in C we are in a position to answer the existence claim in the affirmative. The proof is constructive: we provide a definition, in purely logical terms, of broad necessity. A proposition is broadly necessary if and only if it is necessary for every necessity.

Definition 7.9 (Broad Necessity)**.** *To be* broadly necessary*, or* necessary in the highest degree*, is to be necessary for every necessity:*

$$\Box := \lambda p.\forall_{t\to t} X(\mathrm{Nec}\, X \to Xp)$$

Recall that we defined the predicate Nec in purely logical terms so that $\Box$ can ultimately be defined from purely logical expressions: the truth-functional connectives and the (higher-order) quantifiers.

Exercise 7.15. *In this exercise, you will show that broad necessity is a necessity.*

a. Prove, in C*, that broad necessity applies to* $\top$ *and is closed under modus ponens.*

b. Show that $\lambda p(p = \top)$ *is a* KTheory $\Box$ *and conclude that broad necessity is a weak necessity.*

c. Using the Rule of Equivalence, show that $(\mathrm{WNec}\,\Box) =_t \top$*, and infer that* $\Box$ *is a necessity.*

Not only is broad necessity a necessity, but it is an infinitely closed necessity. You will establish this in the next exercise

Exercise 7.16. *In this exercise, you will show that* $\mathrm{Nec}_\infty\,\Box$*. Suppose that* $\Box \leq_{ts} p$*—i.e. any proposition r that entails every broadly necessary proposition, entails p—and show that* $\Box p$*.*[iii]

Exercise 7.17. *Show that broad necessity,* $\Box$*, is as broad as any other necessity:* $\forall_{t\to t} X(\mathrm{Nec}\, X \to \Box \leq X)$.

Being the broadest necessity is actually a weaker condition than we might like. For it merely states that $\Box$ is as broad as any other actually existing necessity. But what if there could have been, in some sense of 'could', a necessity which was strictly broader than $\Box$? Luckily this situation cannot arise: $\Box$ is necessarily the broadest necessity in every sense of 'necessarily'.

Proposition 7.3. $\Box$ *is* necessarily *the broadest necessity, in all senses of* 'necessarily'*. In* C *we can prove:* $\Box\forall_{t\to t} X(\mathrm{Nec}\, X \to \Box \leq_{t\to t} X)$

Since we established $\forall_{t\to t} X(\mathrm{Nec}\, X \to \Box \leq_{t\to t} X)$ within Classicism (Exercise 7.17), this claim must be $\top$. Since $\top$ is necessary for every necessity, the claim that $\Box$ is the broadest necessity must be necessary for every necessity.

Note that *now* we have introduced a broadest necessity, we could alternatively define propositional entailment by a broadly strict implication: $\lambda pq.\Box(p \to q)$. The reader should check for themselves that this is equivalent, given definitions, to our original definition of entailment.

We have thus established that there exists a broadest necessity. But is it unique? The answer to this follows from a much more general thesis that is a consequence of C, which we will call *Intensionalism*. This is the thesis that higher-order entities with relational type are individuated by broadly necessary coextensiveness. Thus, for instance, we have:

Propositional Intensionalism $\Box(p \leftrightarrow q) \to p =_t q$

Property Intensionalism $\Box\forall_\sigma x(Fx \leftrightarrow Gx) \to F =_{\sigma\to t} G$

Relation Intensionalism $\Box\forall_\sigma x\forall_\tau y(Rxy \leftrightarrow Sxy) \to R =_{\sigma\to\tau\to t} S$

We will prove Intensionalism in a minute. Assuming it, we can show that there cannot be more than one broadest necessity:

Exercise 7.18.

a. Show, assuming Intensionalism, that if necessities X and Y are as broad as one another, they are identical.

b. Show that the broadest necessity is unique.

Exercise 7.19. *In this exercise, you will establish Propositional Intensionalism.*

a. Prove that if $(p \leftrightarrow q) =_t \top$ *then* $p =_t q$ *in* C.[iv]

b. Show that if $\Box(p \leftrightarrow q)$ *then* $(p \leftrightarrow q) = \top$.[v]

Theorem 7.1 (Intensionalism)**.** *In Classicism relations are individuated by broadly necessary coextensiveness:*

Intensionalism $\Box\forall_{\overline{\sigma}}\overline{z}(R\overline{z} \leftrightarrow S\overline{z}) \rightarrow R =_{\overline{\sigma}\rightarrow t} S$

Proof. The proof is a generalization of the argument in Exercise 7.19 to arbitrary relations. Firstly we observe that:

$$\lambda\overline{y}.(\forall_{\overline{\sigma}}\overline{z}(R\overline{z} \leftrightarrow S\overline{z}) \rightarrow R\overline{y}) = \lambda\overline{y}.(\forall_{\overline{\sigma}}\overline{z}(R\overline{z} \leftrightarrow S\overline{z}) \rightarrow S\overline{y})$$

is a theorem of C. This follows from the Rule of Equivalence, because the corresponding biconditional, obtained by removing $\lambda\overline{y}$ and replacing $=$ with $\leftrightarrow$ in the above, is a theorem of H. Now if $\Box\forall_{\overline{\sigma}}\overline{z}(R\overline{z} \leftrightarrow S\overline{z})$—i.e. $(R\overline{z} \leftrightarrow S\overline{z})$ is necessary in all senses of *necessary*—then, since $\Box_\top$ is a necessity, $\forall_{\overline{\sigma}}\overline{z}(R\overline{z} \leftrightarrow S\overline{z}) =_t \top$. So making that substitution in the above we get

$$\lambda\overline{y}.(\top \rightarrow R\overline{y}) = \lambda\overline{y}.(\top \rightarrow S\overline{y})$$

which of course yields:

$$\lambda\overline{y}.R\overline{y} = \lambda\overline{y}.S\overline{y}$$

and finally $R = S$ by η (we can stipulate at the beginning that $\overline{y}$ are free in neither R nor S). □

Exercise 7.20. *Use Intensionalism to show that the relation of identity between relations* just is *the relation of necessary coextensiveness. For any relational type* $\overline{\sigma} \rightarrow t$, $=_{\overline{\sigma}\rightarrow t} = =_{(\overline{\sigma}\rightarrow t)\rightarrow(\overline{\sigma}\rightarrow t)\rightarrow t} \lambda XY.\Box\forall_{\overline{\sigma}}\overline{z}(X\overline{z} \leftrightarrow Y\overline{z})$.

Exercise 7.21. *Use Theorem 7.1 to show that the alternate definition of entailment (namely* $\lambda YZ.(Y \wedge_{\overline{\sigma}\rightarrow t} Z =_{\overline{\sigma}\rightarrow t} Y)$*) is equivalent to the official definition.*

Exercise 7.22. *Using Intensionalism show that* $\Box =_{t\rightarrow t} \Box_\top$. *(You may assume that Exercises 7.1 and 7.6 have been established.)*

The previous exercise establishes that $\Box =_{t\rightarrow t} \Box_\top$, which is quite useful: sometimes it is much easier to prove something of $\Box_\top$ than of $\Box$.

Remark 7.3. The reader will likely have noticed that $\Box$ has a comparatively complicated definition, whereas $\Box_\top$ does not. Given that they are identical, one might wonder why we have not simply adopted $\Box_\top$ as the definition of broad necessity from the start? The project of giving a successful philosophical analysis is inherently tied to whether the analysis presents the analysandum under the right mode of presentation.[13] This seems widely accepted for mathematical analyses—the seminal $\epsilon\delta$ definition of continuity, for instance, is equivalent

to being a function that satisfies the intermediate value theorem. But the latter would not suffice as an analysis, since there is a substantive bit of reasoning bridging the intuitive concept of continuity and the intermediate value theorem. Similarly, even though broad necessity is identical to being a particular tautology, this is an informative identity that was discovered through some substantive logical reasoning.

If we had, prior to investigation, known that $\Box_\top$ was the broadest necessity, we could have used a simpler hierarchy of definitions. For instance $\lambda X.(\Box_\top \mathrm{MP}\, X \wedge \Box_\top(X\top)))$ turns out to be equivalent to our original definition of a necessity. But but it is not obvious that this is a correct definition of a necessity without first knowing that $\Box_\top$ was itself a kind of necessity, and that there are no necessities stronger than it.

7.5 Possible worlds

Arguably no formal tool has had more influence on the study of modal logic than possible worlds model theory. But while its success as a model theory is suggestive, nothing compels us to regard possible worlds as providing an *analysis* of the modal operators—that to be possible just is to be true in some world, and to be necessary is to be true in all worlds. One could take the necessity and possibility operators as primitive, or take them to be reducible to other notions—the logical primitives as in this chapter, or otherwise—while still taking the possible worlds model theory to be adequate with regards to its account of metalinguistic notions, such as the notion of validity.

Nonetheless, possible worlds talk is pervasive in philosophy and is often applied in a metaphysically serious way that elevates it beyond a mere tool for checking for validity and invalidity. Philosophers will move back and forth, without ceremony, between the claim that P is possible and the existential claim that there is a possible world at which P is true.

Moreover, even when taken as a mere tool for assessing validity and invalidity possible worlds model theory can have substantive commitments. They are not so visible in the language of propositional modal logic, but when used to interpret languages with quantification into sentence position, such as higher-order languages with modal operators, one obtains some very distinctive validities. The possible worlds model theory identifies propositions with sets of possible worlds, and so posits special propositions corresponding to the worlds themselves: sets containing a single world. Thus when quantification into sentence position is interpreted, in a model, in terms of quantification over arbitrary sets of worlds, sentences asserting the existence of these world propositions become valid.

So even if we do not accept the claim that being necessary in the highest degree is *analysed* as being true in all possible worlds, we can still ask whether the more substantive applications of possible world talk are legitimate. Is modal reality as the possible worlds metaphysician says it is? And if so, an analysis in the other direction seems to be in order. Is it possible to analyse world talk in the terms you do take to be basic—in our case the purely logical notions?

The first question we will defer until Section 8.2. One reason the move from 'P is possible' to 'P is true in some possible world' is substantive is because it tells us that the possibility of P being true implies the possibility of P being true, while every other proposition's truth value is settled one way or the other. Claims like this are not theorems of **C** alone, and one can construct models of **C** that refute them. We shall consider a strengthening of **C** which corroborates the possible world metaphysician's view of modal reality in Section 8.2. But we can offer analyses of possible worlds without taking a stance on the metaphysical question of whether things meeting our definition exist.

According to orthodox possible world accounts of propositions and modality, á la Lewis (1986) and Stalnaker (1976), for every world there corresponds a proposition that is true at only that world and which entails all and only the propositions that are true at that world. These 'world propositions' are thus possible, and entail any other proposition or its negation. Someone who does not take possible worlds to be primitive in their theorizing, but can make sense of propositional entailment using their own primitives can use this as a basis of an analysis of possible world talk in their own primitives. This suggests we might also be able to find definitions in higher-order logic that play something like the role of a possible world in a first-order theory of propositions.

In fact, in the framework of Classicism, there are actually two notions of world that fit the bill. The weaker notion, that of a *weak world*, is a broadly possible proposition that settles every question in the sense of entailing any proposition or its negation. A *strong world*, by contrast, is a broadly possible proposition that *necessarily* settles all the questions—informally, something that settles all the questions there in fact are, and also settles all the questions there would have been, had there been any further questions.

Definition 7.10 (Weak world proposition)**.** *A proposition $w : t$ is a weak world proposition if it is possible and entails any other proposition or its negation:*

$$\mathrm{WWorld} := \lambda w(\Diamond w \wedge \forall_t p(w \leq p \vee w \leq \neg p)$$

Definition 7.11 (Strong world proposition)**.** *A proposition $w : t$ is a strong world proposition if it is possible and necessarily entails any other proposition or its negation:*

$$\mathrm{SWorld} := \lambda w(\Diamond w \wedge \Box\forall_t p(w \leq p \vee w \leq \neg p)$$

One can bring out the difference by considering a possibility in which the physical laws are very different to our own, and involve new fundamental properties that don't in fact exist. Consequently, there would be new questions involving these fundamental properties. A strong world would settle these new questions, had the world been like this, while a weak world need not.

Observe that both of these definitions are formulated in purely logical terms, since $\Diamond$ and $\Box$ are defined from logical terms. Unlike Lewis, Stalnaker and others, we have not taken it for granted that the modality in question is metaphysical modality, so we shall occasionally call these 'broadly possible worlds' (weak or strong) to emphasize this.

7.6 Reducing the intensional to the extensional

One prominent question in the philosophy of modality concerns whether it is possible to reduce the modal to the non-modal. David Lewis famously attempted to do this by positing an infinity of concrete possible worlds spatiotemporally disconnected from our own, explaining modal operators in terms of ordinary first-order quantification over these worlds.

However the words 'modal' and 'non-modal' are notoriously slippery. Given a putative reduction of modality to something else, it is often tempting to extend our conception of what is modal to the notion doing the analysing or to shrink it to exclude the modal expressions being analysed. Lewis argued that competing analyses of *possibly* in terms of the notion of a *world* employ an irreducibly modal notion of world. And on Lewis's own view expressions that appear to be modal in reality just express quantificational claims—there are no intensional operators to be analysed in Lewis's extensional metalanguage.[14]

This kind of back and forth seems unproductive. It is hard to see our way past these dialectical stalemates without a precise conception of 'modal' to be arguing about. Related to the distinction between 'modal' and 'non-modal' is the more precise distinction between extensional and intensional entities. An operator (property, relation, etc.) is extensional, roughly, if it is insensitive to differences between coextensive entities.

Definition 7.12 (Coextensiveness)**.** *The relation of being* coextensive*,* $\sim_\rho$*:* $\rho \to \rho \to t$*, between entities of type* ρ *is inductively defined as follows:*

- $\sim_t := \leftrightarrow$
- $\sim_e := =_e$
- $\sim_{\sigma\to\tau} := \lambda XY.\forall_\sigma x(Xx \sim_\tau Yx)$

We will always apply infix conventions when using the $\sim$ *symbol.*

Comprehension Check 7.1. *The reader should convince themselves that two binary relations, R and S, are coextensive when* $\forall_e xy(Rxy \leftrightarrow Sxy)$.

Definition 7.13 (Extensionality and Intensionality)**.** *Suppose* $\overline{\sigma} \to t$ *is the type of a relation and* $R : \overline{\sigma} \to t$*. Then R is extensional iff it is insensitive to differences between coextensive entities. i.e.*

$$\mathrm{Ext}_{\overline{\sigma}} := \lambda X \forall_{\sigma_1} x_1 y_1 \ldots \forall_{\sigma_n} x_n y_n (x_1 \sim_{\sigma_1} y_1 \wedge \ldots \wedge x_n \sim_{\sigma_n} y_n \to (Xx_1 \ldots x_n \sim_\rho Xy_1 \ldots y_n))$$

where ρ *is e or t. A relation is intensional if and only if it is not extensional:* $\mathrm{Int}_{\overline{\sigma}} := \lambda X.\neg(\mathrm{Ext}_{\overline{\sigma}} X)$

The paradigm example of an extensional operator would be negation: if p and q are coextensive (i.e. materially equivalent), then so are $\neg p$ and $\neg q$. Metaphysical necessity, $\Box_M$, is a paradigm example of an intensional operator: p and q may be materially equivalent even though $\Box_M p$ and $\Box_M q$ are not—consider, for instance, the case where p is the proposition that there are eight planets, and q the proposition that eight is an even number.

Exercise 7.23.

a. Show that $\wedge$ *is extensional.*

b. Show that $\forall_\sigma$ *is extensional.*

c. Show that $\neg_{e\to t}$ *is extensional.*

Exercise 7.24. *In this exercise we consider an alternative definition of extensionality. Propositions are always extensional. Suppose that we have defined extensionality for relations of arity n. Then a relation* $R : \sigma_1 \to \ldots \to \sigma_{n+1} \to t$ *is extensional iff Rx is extensional for every* $x : \sigma_1$*, and whenever* $x \sim_{\sigma_1} y$*,* $Rx \sim_{\sigma_2\to\ldots\to\sigma_{n+1}\to t} Ry$*. Show that this definition of extensionality is equivalent to our original definition.*

With these notions in place, one can pose the following more precise question:

Can intensional notions be defined from extensional ones?

Perhaps the surprisingly, the answer to this question is yes. Indeed, our key notion of broad necessity is one example: by chasing the definitions backwards we see that it is defined from the higher-order quantifiers $\forall_\sigma$ and the truth-functional connectives, and we established that

these were extensional above. To complete the case we need to show that $\Box$ is intensional. We cannot establish this in **C** alone, since **C** is consistent with the Fregean axiom, according to which there are only two propositions, and *all* operators are extensional. However, if we assume there are more than two propositions then we may prove that $\Box$ is intensional. For let p be a proposition distinct from $\top$ and from $\bot$. Then $\neg p$ is also distinct from $\top$ and $\bot$, and one of p and $\neg p$ must have the same truth value as $\top$ (i.e. must be true). Without loss of generality suppose p is true: then we have p and $\top$ are materially equivalent, while $\Box p$ is false (since p is distinct from $\top$, some necessity fails to apply to p, namely $\Box_\top$) and $\Box\top$ is true.

Assuming we are working in a minimal signature for pure higher-order logic—say, one that takes the material conditional $\rightarrow$, and the higher-order quantifiers $\forall_\sigma$ as primitive—then every notion definable in the pure higher-order signature is defined from extensional primitives. This provides us with a variety of examples of intensionality arising from extensional notions. We explore this in the following exercise.

Exercise 7.25. *Assume that there are more than two propositions*

a. Show propositional identity, $=_t$, defined by $\lambda pq.\forall_{t\rightarrow t}X(Xp \leftrightarrow Xq)$, is intensional.

b. Show that the operation of operator application, $\lambda X.\lambda p.Xp$, is intensional.

The second part of this exercise illustrates how intensionality can arise in the pure λ-language without the need of *any* extensional primitives. One might be tempted to then raise the question of whether λ itself is intensional. This question doesn't quite make sense as it stands, since λ is a syncategorematic operation, and so is not the sort of thing that can take the position of an argument to our extensionality predicate, $\mathrm{Ext}_{\overline{\sigma}}$. However, there are alternative frameworks that achieve the job λ fills through other means, such as in a combinatory language. Since in a combinatory language the only primitives are combinators many of which are relations (have types ending in t) and thus may be assessed for extensionality. Thus we might ask whether the combinators in the standard S, K, I basis are extensional, and more generally, whether there could be an extensional basis for a combinatory language.

Exercise 7.26. *Show that the K^{tt} combinator is extensional and that I^t is extensional. Assuming there are more than two propositions, show that S^{ttt} is not extensional, and find types σ, τ such that I^σ and $K^{\sigma\tau}$ are intensional.*

Exercise 7.27.

a. Prove by induction that if $M : \sigma \rightarrow \tau$ and $N : \sigma$ are extensional, then so is MN.

b. Assuming there are more than two propositions, show that no basis of combinators can contain only extensional elements.

The emergence of intensionality from extensional primitives is a striking phenomenon. We end by making some remarks about the features of the present system of higher-order logic permit it.

The appearance of intensionality from extensional primitives seems only to occur when we allow λ-abstraction of variables appearing outside of argument position: for instance the third occurrence of X in $\forall_{t\rightarrow t}X(\mathrm{Nec}\, X \rightarrow Xp)$, or the second occurrence of X in $\lambda Xy.Xy$ are in predicating position, and were two examples of the emergence of intensionality. The latter case is η-equivalent to $\lambda X.X$ where the second X is not appearing in argument or predicating position. The reader may check that the other examples we have discussed similarly involve

the binding of variables that do not appear in argument position. In Chapters 9 and 10 we will consider λ-languages that do not permit variables appearing outside of the argument position to be bound. It turns out that in systems like this intensionality cannot emerge from extensional primitives—this will be explored later in Exercise 10.15. Observe also that the appearance of intensionality also essentially involves quantification or λ-abstraction of variables appearing in operator position. The emergence of intensionality from extensional primitive does not appear to arise in second-order logic where one only has variables and quantifiers for first-order relations of type $e \to \ldots \to e \to t$, even though these variables can appear in predicating position. Languages with higher-order quantifiers or λ containing variables with operator type or types of higher complexity appear to be required for this phenomenon to arise.

Endnotes

1. This is often attributed to Kripke, see Kripke (1980), p. 99.
2. This definition is adequate given **C**. However, if one were to assume a more fine-grained theory of operators the idea that possibility and necessity are duals in this sense might be too strong, since it might seem to imply that one is more logically complex than the other (since one has negation as a constituent). A fine-grained metaphysician might prefer a weaker relationship to hold, where necessity is merely necessarily coextensive with the dual of possibility; the necessity appealed to in specifying 'necessary coextensiveness' would then have to be spelled out.
3. See, respectively, p. 111, pp. 113–114, p. 24.
4. Kripke (1980) p. 99.
5. Fritz (2017b). Similar claims are also made in Williamson (2016), Dorr and Goodman (2020).
6. For better or for worse, this assumption is made at least implicitly in the majority of work on metaphysical necessity following Kripke.
7. A few representative works defending this line of thought are Lewis (1986), Stalnaker (1976), Plantinga (1974), Forbes (1985). Few post-Quinean authors have explicitly rejected the possible worlds picture; but see Chihara (1998) for one example.
8. Of course, it is very unlikely that the notion of legal necessity or practical necessity is fundamental or perfectly natural, so there is a question as to whether we can really rule out the complex operators above solely on the grounds of being too 'gruesome'.
9. Some conditions under which reductive analyses of necessity and similar notions are possible in theories not extending **C** can be found in Bacon and Zeng (2022), Section 6.2.
10. At later points we employ LogicK to define other notions, such as entailment and broad necessity. It turns out not to matter much to these notions whether we use the stated definition or the weaker variant that is more closely modelled on a sentential modal logic.
11. Some authors, for instance Fine (2002b), use the terminology in the opposite order: stronger necessities are narrower because they consider fewer things necessary.
12. This is notion adopted in Bacon and Dorr (forthcoming), for instance.
13. See Williamson (forthcoming).
14. See Williamson (2013), Sections 7.4 and 8.4 for relevant discussion pertaining to Lewis's project.

Hints for exercises

[i] **Hint:** Use the fact that $(\top \to q) =_t q$.
[ii] **Hint:** Establish the following lemmas first: $\mathrm{MP}\ X$ and $X(\mathrm{MP}\ Y)$.
[iii] **Hint:** Show first that $\top$ entails every broadly necessary proposition.
[iv] **Hint:** First show $((p \leftrightarrow q) \to q) =_t ((p \leftrightarrow q) \to p)$ in **C**.
[v] **Hint:** Recall that you have already established that $\Box_\top$ is a necessity.

8

Application

Consequences and strengthenings of Classicism

Given that broad necessity can be defined from logical notions, it follows that the *modal* logic of broad necessity may be reduced to the logic of the more straightforwardly logical notions—the truth-functional connectives and the quantifiers. In this chapter, we will take the logic of the quantifiers and connectives to be **C**. Section 8.1 takes a look at which modal principles are already derivable in Classicism: the principles of **S4** (but not **S5**), the necessity of identity (but not of distinctness), the converse Barcan formula (but not the Barcan formula), Modalized Functionality (but not Functionality). In Section 8.2 we will look at extensions of Classicism that strengthen the modal logic of $\Box$, such as the parenthetical principles just listed, giving us ways to motivate new logical principles—principles formulated purely in terms of the quantifiers and truth-functional connectives—from modal intuitions. Conversely, principles from Section 6.5, like Functionality and Functional Plenitude, are formulated and motivated more directly in terms of the logical notions, but have many substantive modal consequences (such as the Barcan formula and the principles of **S5** respectively) thus giving us a route to justifying modal principles from logical principles. In Section 8.3 we investigate the notion of logical necessity in the framework of Classicism. The final section provides the reader with some further reading on higher-order modal logic.

8.1 The modal logic of broad necessity

Here is a (non-exhaustive) list of well-known modal principles:

K $\Box(A \to B) \to \Box A \to \Box B$

T $\Box A \to A$

4 $\Box A \to \Box\Box A$

5 $\Diamond A \to \Box\Diamond A$

B $A \to \Box\Diamond A$

M $\Box\Diamond A \to \Diamond\Box A$

G $\Diamond\Box A \to \Box\Diamond A$

By adding all instances of these schemas besides K to the minimal normal modal logic $\mathsf{K}^{\Box}$ and closing under implication in propositional logic and the rule of necessitation one can obtain further normal modal logics. (Recall that a logic is closed under the rule of necessitation iff, whenever it contains A it also contains $\Box A$.)

DOI: 10.4324/9781003039181-11

One might wonder which of these further axiom schemas can be proven, in C, when $\Box$ is read as broad necessity. (We will employ the conventions of Chapter 7, writing $\Box$ as short for $\lambda p.\forall_{t\to t} X(\mathrm{Nec}\, X \to Xp)$.) Indeed, just as we identified the propositional analogue of a normal modal logic (i.e. a certain class of propositions containing certain axiom propositions and closed under certain rules), we might seek propositional analogues of these stronger modal logics. To ease readability, we have written $\langle X\rangle$ as short for $\lambda p.\neg(X(\neg p))$, and have omitted evident brackets.

Definition 8.1. *We introduce the following predicates:*

$\mathsf{K} := \lambda X.\Box\forall_t pq(X(p \to q) \to Xp \to Xq)$

$\mathsf{T} := \lambda X.\Box\forall_t p(Xp \to p)$

$\mathsf{4} := \lambda X.\Box\forall_t p(Xp \to XXp)$

$\mathsf{5} := \lambda X.\Box\forall_t p(\langle X\rangle p \to X\langle X\rangle p)$

$\mathsf{B} := \lambda X.\Box\forall_t p(p \to X\langle X\rangle p)$

$\mathsf{M} := \lambda X.\Box\forall_t p(X\langle X\rangle p \to \langle X\rangle Xp)$

$\mathsf{G} := \lambda X.\Box\forall_t p(\langle X\rangle Xp \to X\langle X\rangle p)$

Clearly any principle of propositional modal logic can be turned into a predicate of necessities in this way. For any sentence of propositional modal logic, A, there is a predicate of necessities $\mathsf{A} : (t \to t) \to t$ defined by the term $\lambda X.\Box\forall_t \overline{p} A[X/\Box, \overline{p}/\overline{P}]$ obtained by replacing $\Box$ with X, and replacing the propositional constants $\overline{P}$ in A with propositional variables.

Observe that for a necessity to be factive—i.e. to satisfy the predicate T—it must not only apply to truths, it must necessarily do so, in every sense of necessity. This ensures that if X is a necessity and satisfies one of these conditions then it's X-necessary that it does so, it's X-necessary that it's X-necessary that it does so, and so on. This accords with the modal logics corresponding to the principles listed above: in the modal logic KT for instance (obtained by adding T to $\mathsf{K}^{\Box}$), the T principle is not only true, but necessarily true, necessarily necessarily true, and so on. Indeed, one can show that for any theorem A of a modal logic obtained by augmenting K with a finite list of further modal principles, there is a corresponding truth about a necessity satisfying the predicates corresponding to those modal principles. Specifically, it will also satisfy the sentence corresponding to that theorem, $\mathsf{A}X$.[1]

Proposition 8.1. *Let* $\mathbf{L}$ *be a finitely axiomatizable normal modal logic obtained by adding modal schemas* $A_1, \ldots, A_n$ *to* $\mathsf{K}^{\Box}$*, and let* A *be a theorem of* $\mathbf{L}$*. Then we may prove*

$\forall_{t\to t} X(\mathrm{Nec}\, X \wedge \mathsf{A}_1 X \wedge \ldots \wedge \mathsf{A}_n X \to \mathsf{A}X)$

Proof. Proof sketch: this is a routine induction on the length of the derivation of A. The property of being a A such that

$\forall_{t\to t} X.(\mathrm{Nec}\, X \wedge \mathsf{A}_1 X \wedge \ldots \wedge \mathsf{A}_n X \to \mathsf{A}X)$

is provable in C holds whenever A is axiom, and is preserved by the rules of modus ponens and necessitation. $\Box$

Remark 8.1. Our notion of a T-necessity, a 4-necessity, and so on, are appropriate for theorizing about necessities but not weak necessities. In order for X to be a T-necessity, for instance, it must broadly necessary that it satisfies the T axiom (i.e. $\Box\forall_t p(Xp \rightarrow p)$), reflecting the fact that necessities have their basic logical behaviour of broad necessity (like being closed under modus ponens). By contrast, a weak necessity, like the beliefs of a contingently logically perfect and introspective agent, will not have their basic logical properties of broad necessity, they will just be internally necessary. One can introduce corresponding weak notions of the listed modal properties above by replacing $\Box$ with X^*; e.g. $\mathsf{T}^{wk} := \lambda X(X^*\forall_t p(Xp \rightarrow p))$.

Theorem 8.1. *Broad necessity has a logic of* S4. *For any theorem A of* S4, $\mathsf{A}\Box$ *is a theorem of* C.

Given Proposition 8.1 it suffices to show that K$\Box$, T$\Box$, and 4$\Box$. This is an instructive exercise: K follows from the fact that every necessity is closed under modus ponens. T follows from the previously established fact that $\lambda p.p$ (aletheic necessity) is a necessity. And 4 from the fact that $\Box \circ \Box$ is a necessity; we have previous established that $\Box$ is a necessity and necessities are closed under composition.

Exercise 8.1. K$\Box$, T$\Box$, *and* 4$\Box$.

Here's a useful distinction between higher-order logics that extend C:

Definition 8.2 (Normal extension of Classicism)**.** *Let* **L** *be a higher-order logic extending* C *(i.e.* $\mathbf{L} \subseteq \mathsf{C}$*).* **L** *is* a normal extension *if and only if, for any formula A, if $A \in \mathbf{L}$ then $\Box A \in \mathbf{L}$.*

Not every logic extending C is a normal extension. A simple example would be the result of adding the unnecessitated version of Brouwer's axiom, $\forall_t p(p \rightarrow \Box\Diamond p)$, to C and closing under the logical rules does not contain the necessitation of Brouwer's axiom $\Box\forall_t p(p \rightarrow \Box\Diamond p)$. (In order to prove this, however, we will need to wait until we have some more model-theoretic tools.)

In turns out that Classicism itself is closed under necessitation. (Of course, we have just shown that C contains a subset, namely the theorems of the propositional modal logic S4, that is closed under necessitation, but an additional argument is needed to show that C itself is closed under necessitation.)

Proposition 8.2. *C is normal.*

This follows straightforwardly given the identity of $\Box$ with $\Box_\top$, for if A is a theorem of C, so is $A \leftrightarrow \top$, and since C is closed under the rule of equivalence $A =_t \top$ is a theorem too.

Exercise 8.2. *We stated that C plus $\forall_t p(p \rightarrow \Box\Diamond p)$ is not closed under necessitation. Show that C plus $\Box\forall_t p(p \rightarrow \Box\Diamond p)$, by contrast,* is *closed under necessitation.*

What other principles of modal logic can we prove about the broadest necessity? The most salient strengthening of the logic S4 is S5, which can be obtained by adding the B or 5 principles to S4. However, there is no obvious way to prove either of these principles within Classicism. One can raise similar questions about other modal logics that extend S4. In Chapter 18 we will develop some model-theoretic tools that will let us treat this question in more generality. The gist is this: It is a well-known fact that every sentence that is not a theorem of S4 can be refuted in a 'transitive reflexive Kripke frame'. In Chapter 18 we will show how to construct, for every Kripke model over a transitive reflexive Kripke frame, a corresponding higher-order model of C with the same modal truths. What this means is

that *no* sentence of propositional modal logic not already in S4 can be proven within C (see Theorem 18.2).

Remark 8.2. The fact that strengthenings of S4, like the B principle, cannot be proven within C does not mean, necessarily, that they are false. In the subsequent sections, we will explore strengthenings of C that entail the B principle, and other strengthenings that refute it.

So far we have only discussed the propositional modal logic of broad necessity. Next we look at some well-known modal principles that involve the quantifiers and identity. We'll begin with the necessity of identity:

NI$^{\sigma}$ $\forall_{\sigma} xy(x =_{\sigma} y \rightarrow \Box x =_{\sigma} y)$

The first known derivation of this fact (in the case where $\sigma = e$) is due to Barcan, whose proof was later simplified by Quine.[2] Here, in essence, is Quine's argument. Suppose $x =_{\sigma} y$. In C we may infer that $\Box x =_{\sigma} x$, using the rule of necessitation so that x has the property $\lambda z.\Box x =_{\sigma} z$. Since by Leibniz's law x and y have the same properties, y has $\lambda z.\Box x =_{\sigma} z$, so $\Box x =_{\sigma} y$ by β.

Exercise 8.3.

a. Prove that all necessities satisfy analogues of the necessity of identity: $\forall_{t \rightarrow t} X(\mathrm{Nec}\, X \rightarrow \forall_{\sigma} xy(x =_{\sigma} y \rightarrow X(x =_{\sigma} y))$.

b. Using part a prove that $\Box_{\top}$ *satisfies the* S4 *principle:*

$$p =_{t} \top \rightarrow (p =_{t} \top) =_{t} \top$$

You may appeal to the fact that $\Box_{\top}$ *is a necessity, but not to Theorem 8.1 or the fact that* $\Box =_{t \rightarrow t} \Box_{\top}$.

Observe that, since Hesperus is Phosphorus, we can infer from the necessity of identity that it's necessary, in every sense of necessity, that Hesperus is Phosphorus. This might initially strike one as puzzling, since idealized epistemic and doxastic attitudes appear to have a normal modal logic and so plausibly express operators satisfying our conditions for being a necessity. Questions relating to propositional attitudes are indeed puzzling: recall, as we noted in Chapter 6, that we simply cannot combine face value judgements about propositional attitudes with Leibniz's law. The usual attitude towards this conflict is that, as a principle of metaphysics, Leibniz's law is valid, and had one been theorizing in a metaphysically perspicuous language that reveals the true logical structure of belief reports, there would be no failures of Leibniz's law. Whatever features are responsible for the apparent failures of Leibniz's law in opaque languages, such as English, they are not present in our higher-order language.

Another important modal principle, this time involving the interaction of the quantifiers with the modal operators, is the converse Barcan formula:

CBF$^{\sigma}$ $\forall_{\sigma \rightarrow t} X(\Box \forall_{\sigma} y\, Xy \rightarrow \forall_{\sigma} y \Box Xy)$

This too is a theorem of C. $\forall_{\sigma} y\, Xy \rightarrow Xy$ is an instance of UI. By necessitation we can infer $\Box(\forall_{\sigma} y\, Xy \rightarrow Xy)$, and by K we can infer $\Box \forall_{\sigma} y\, Xy \rightarrow \Box Xy$. Using Gen we get $\Box \forall_{\sigma} y\, Xy \rightarrow \forall_{\sigma} y \Box Xy$, and CBF$^{\sigma}$ follows using propositional logic and Gen again. As with NI$^{\sigma}$, CBF$^{\sigma}$ has analogues for every necessity operator. That is, we may prove in a completely parallel fashion:

$$\forall_{t \rightarrow t} Z(\mathrm{Nec}\, Z \rightarrow \forall_{\sigma \rightarrow t} X(Z \forall_{\sigma} y\, Xy \rightarrow \forall_{\sigma} y\, Z(Xy)))$$

CBF^e has received a lot of attention due to the fact that it is both a theorem of the result of combining classical first-order logic with the principles of a normal modal logic, and that it has some highly non-obvious consequences. For instance, letting X be the property of being identical to something: $\lambda y.\exists_e x(x =_e y)$. By necessitating the theorem of **H**, $\forall_e y \exists_e x(x =_e y)$ (clearly everything is identical to something, namely itself), we get that it is broadly necessary that everything is identical to something. So CBF^e let's us infer that everything necessarily is identical to something:

Broad Necessitism $\forall_e y \Box \exists_e x(x =_e y)$

But of course it seems as though there are many things that could have failed to be anything—in many senses of *could* and thus in the broadest sense of *could*. Say a proposition is *counterfactually possible* when it doesn't counterfactually imply a contradiction. Had my parents never met, for instance, I wouldn't have been anything, or so it seems. But it is not true that had my parents never met contradictions would have been true, so by standard principles of counterfactual logic we can conclude that it is counterfactually possible that I do not exist: $\neg(\neg\exists_e x(x =_e a) \boxdot\!\!\rightarrow \bot)$.

There is a vast literature on this consequence of combining classical with modal logic so we must make do with some cursory remarks. Some philosophers have simply accepted Broad Necessitism.[3] Since the English quantifiers are highly context-sensitive, and the predicate 'exists' seems to have its own life, there are plenty of true readings of sentences like 'I could have failed to be anything' and 'I could have failed to exist' with which to interpret our intuitive judgements, leaving room for CBF^e and Broad Necessitism to be true on the more theoretical (and less accessible) unrestricted readings of the quantifiers. However, even the so-called 'actualists' who reject Broad Necessitism on what they take to be the unrestricted reading of the quantifiers, will nonetheless often posit 'possibilist' or 'outer' quantifiers, either as primitive or by definition in terms of modal notions and regular quantification.[4] Under these readings of the quantifiers CBF^e and Broad Necessitism come out true, and so even this sort of philosopher has a way of interpreting these consequences of **C** in a benign way, systematically interpreting $\forall$ and $\exists$ in terms of these outer quantifiers. Of course, the posited outer quantifiers seem to range more widely than what actualists take to be the regular quantifiers to range, suggesting, perhaps, that the interpretation of the quantifiers on which CBF^e and Broad Necessitism come out false are not really the unrestricted quantifiers after all. We will not attempt to adjudicate this issue here, but the reader may wish to revisit the remarks we made in Section 0.2 about the relation between natural language quantificational idioms and logical generalizations involving the symbol $\forall$ and $\exists$.

The last consequence of Classicism we will consider is:

Modalized Functionality $\forall_{\sigma\to\tau} XY(\Box\forall_\sigma z(Xz =_\tau Yz) \to X =_{\sigma\to\tau} Y)$

Modalized Functionality weakens the Functionality principle of Section 6.5 in a natural way. In fact, one of the reasons we gave to be suspicious of Functionality was that X and Y could agree in their applicative behaviour with respect to all the actually existing individuals, but still be different because they possibly differ over merely possible individuals. This worry does not extend to Modalized Functionality, which requires that X and Y necessarily agree on all individuals, in every sense of *necessarily*. Modalized Functionality is in fact a theorem of **C**. Suppose X and Y have relational types of the form $\overline{\sigma} \to t$ for some sequence of relational types $\overline{\sigma}$. Let $\overline{z} : \overline{\sigma}$ be a sequence of variables with types matching $\overline{\sigma}$. Then the antecedent of Modalized Functionality implies $\Box\forall_{\overline{\sigma}}\overline{z}(X\overline{z} =_t Y\overline{z})$, which itself implies $\Box\forall_{\overline{\sigma}}\overline{z}(X\overline{z} \leftrightarrow Y\overline{z})$. But now we can apply Relational Intensionalism to infer that $X =_{\overline{\sigma}\to t} Y$.

Observe that this argument rested essentially on the fact that every functional type ends in a 't'—an assumption we made when we restricted our attention to the relational types. However, Modalized Functionality still makes sense once the restriction to relational types is lifted. Once this is done, it implies that necessarily cofunctional entities from types like $e \to e$, $(e \to t) \to e$, $t \to e$, and so on, must be identical. These principles both seem like an obvious extension of the idea that necessarily coextensive entities are the same, and they also naturally fall out of the sorts of model theory we will develop for **C** in Section 18. Thus we will consider 'Classicism' in the full type hierarchy to include Modalized Functionality at all types.[5]

8.2 Some strengthenings of Classicism and their modal consequences

In this section, we will consider a very non-exhaustive list of strengthenings of **C**. Some of these extensions are inspired directly by principles governing the connectives and quantifiers, providing a way to justify further modal principles. But other principles are inspired by modal ideas, giving us an indirect route to motivating strengthenings of the logic of quantifiers and connectives, some of which would seem highly non-obvious when stated without the detour through modal notions.

One route to strengthening Classicism along modal lines is to add to it further principles of propositional modal logic. The most salient modal principle absent from **S4** is Brouwer's principle, B. The theory obtained by adding all instances of the B axiom to **S4** and closing under the rule of necessitation and modus ponens is the well-known system **S5**, often thought to be the modal logic of metaphysical necessity.

B $\forall_t p(p \to \Box\Diamond p)$

(Observe that the usual axiom schema has been replaced with a single universally quantified axiom; the significance of this will be revisited in Remark 8.4.)

Apart from strengthenings of the propositional modal logic, there are also strengthenings that involve the interaction of modality with the quantifiers and identity. Related to the Necessity of Identity and the converse Barcan formula, which are theorems of **C**, we have the Necessity of Distinctness and the Barcan formula which are not. Writing $x \neq_\sigma y$ for $\neg(x =_\sigma y)$ these are respectively:

ND$^\sigma$ $\forall_\sigma xy(x \neq_\sigma y \to \Box x \neq_\sigma y)$

BF$^\sigma$ $\forall_{\sigma\to t} X(\forall_\sigma y \Box Xy \to \Box\forall_\sigma y\, Xy)$

ND$^\sigma$ says that when two entities are different they are necessarily so, and BF$^\sigma$ can be glossed as saying that for any property which is possibly instantiated, there is actually an entity which possibly instantiates it (looking at the contraposed version of BF$^\sigma$).

Remark 8.3. The reader should be aware that the higher-order logics you obtain by adding B, ND$^\sigma$ or BF$^\sigma$ to **C** are not closed under necessitation. By contrast the higher-order logics you get by adding the 'necessitation' of any of these principles—the result of prefixing a '$\Box$' to be beginning of their statements—are closed under necessitation. One should therefore be mindful of the distinction between these principles and their necessitations.

There are close connections between B, ND$^\sigma$ and BF$^\sigma$. Let's begin with Brouwer's principle and the Necessity of Distinctness. The first thing to note is that Brouwer's principle follows from the necessary distinctness of distinct propositions, ND^t. Suppose that p, for some arbitrary proposition p. Since $\Diamond =_{t\to t} \Diamond_\top$, it suffices to show $\Box\Diamond_\top p$. Since $\top$ holds and

$\neg p$ doesn't, Leibniz's law lets us infer that $\neg p \neq_t \top$, so by the necessity of distinctness we have $\Box(\neg p \neq_t \top)$, i.e. $\Box\Diamond_\top p$ as required. In the other direction, we can prove every instance of ND^σ from B. The argument is due to Prior (1964), pp. 206–207 . Firstly note that the Necessity of Identity, $x =_\sigma y \rightarrow \Box x =_\sigma y$, when contraposed implies $\Diamond x \neq_\sigma y \rightarrow x \neq_\sigma y$, and by necessitating and applying K to this we can infer $\Box\Diamond x \neq_\sigma y \rightarrow \Box x \neq_\sigma y$. But by B, $x \neq_\sigma y \rightarrow \Box\Diamond x \neq_\sigma y$, so chaining these implications together we get $x \neq_\sigma y \rightarrow \Box x \neq_\sigma y$, as required. Summarizing:

Proposition 8.3. *ND^σ and* B *are equivalent in* C.

Notice that the Necessity of Distinctness at all types follows from the necessity of distinct propositions, ND^t, since we have seen that ND^t implies B, and B implies ND^σ. We can also see this more directly, for if x and y are distinct entities of type σ, but are possibly identical, then there are a pair of distinct propositions that are possibly identical, namely $x =_\sigma y$ and $x =_\sigma x$. I.e. if ND^σ fails at any type, it fails at type t.

Exercise 8.4. *Suppose $x, y : \sigma$ are distinct but possibly identical. Show that $x =_\sigma y$ and $x =_\sigma x$ are distinct but possibly identical.*

It may strike the reader that the necessity of identity and distinctness should stand or fall together, for what sort of case could one make for one of the principles that one could not make for the other? Contrary to this line of thinking, however, the preceding discussion seems to show that there really is a justificatory asymmetry. The case for the necessity of identity seemed of a logical nature, appealing only to Leibniz's law and the necessity of self-identity claims, whereas the case for the necessity of distinctness appealed to a distinctively modal principle, B. Given a modal logic of S5, which includes all instances of B, they do stand or fall together. Yet given a view of modality in which B has failures, we know at least that distinct propositions may be contingently distinct: for if p is true but possibly necessarily false, p must be distinct from $\bot$ in virtue of being true, but possibly identical to $\bot$ in virtue of the necessary truth that necessary falsehoods are identical to $\bot$. We will shortly discuss one picture of modal reality on which B fails.

Next on the agenda is the Barcan formula, the schema BF^σ. Unlike converse Barcan formula, CBF^σ, the Barcan formula is not a theorem of C, although as with the necessity of distinctness, it becomes one once a modal logic of S5 is assumed. An argument for this was originally due to Prior (1956), who used the full power of S5. Lemmon presented a simplified argument that uses only the necessitation of B, $\Box\forall_t p(p \rightarrow \Box\Diamond p)$, which we will abbreviate $\Box$B. We leave this to the reader as an instructive exercise:

Exercise 8.5.

a. Prove $\Diamond\forall_\sigma y Xy \rightarrow \forall_\sigma y \Diamond Xy$ in C.[i]

b. Prove $\forall_\sigma y \Diamond\Box Xy \rightarrow \forall_\sigma y\, Xy$ from B *in* C.

c. Using parts a and b show that $\forall_\sigma y \Box Xy \rightarrow \Box\forall_\sigma y\, Xy$ follows from $\Box$B *in* C.

Without $\Box$B, BF^σ must be taken as a substantive principle. Indeed, certain modal intuitions concerning the possibility of new objects that do not actually exist seem to militate against BF^e. For instance, many people believe that since Wittgenstein had no children, no actual thing is possibly a child of Wittgenstein.[6] Yet it is surely possible that Wittgenstein had children, in which case we have a counterexample to BF^σ in its dualized form: $\neg\exists_e x \Diamond Wx$ and $\Diamond\exists_e x\, Wx$.

While the view that new things can come into existence militates against BF^e and seems consistent with C, we are forced to reject the view that things can go out of existence, since

this is a consequence of CBF^e which we proved earlier. One might have thought that these wordviews must stand or fall together, for what case could we make for the idea that things can't go out of existence that we couldn't make for the idea that things can't come into existence, and conversely. But as with our discussion of the necessity of identity and distinctness, there is a real justificatory asymmetry, since the argument for CBF^σ had a purely logical nature, whereas the argument for BF^σ requires a distinctive and debatable modal principle, this time the necessity of B.

As previously advertised, we can use BF^σ to motivate seemingly 'non-modal' principles that can be naturally stated in terms of the logical constants. One is the principle of Functionality from Section 6.5:

Functionality$^{\sigma\tau}$ $\forall_{\sigma\to\tau} XY(\forall_\sigma z(Xz =_\tau Yz) \to X =_{\sigma\to\tau} Y)$

Functionality simply says that entities with functional types—properties, operators, relations and so on—are individuated by their applicative behaviour and do not appear to be overtly modal. Nonetheless it is equivalent to the Barcan formula—I leave this as an exercise to the reader:

Exercise 8.6.

a. Show that the Functionality principle entails the Barcan formula for $\Box_\top$ *(and thus, also, for* $\Box$*).*[ii]

b. Show that the Barcan formula at all types entails Functionality. You may either assume that we are working in a relational type system or else interpret Classicism to include the Modalized Functionality principle.[iii]

Another way of thinking about BF^σ is that it states that universal generalizations are just the conjunctions of their instances. E.g. *everybody is tall* is the same as the conjunction *Aaron is tall and Zach is tall and Mary is tall and...* where the ellipses are filled out with a conjunct for every individual. Since this is a view often attributed to Wittgenstein in the Tractatus, we'll call this Tractarianism.

Tractarianism $\forall_\iota p \forall_{\sigma\to\iota} X(\forall_\sigma y(p \leq Xy) \to p \leq \forall_\sigma y\, Xy)$

The formula we have written above corresponds to the conjunction introduction rule: if some premise (represented by p above) entails each of some conjuncts (in this case, the instances Xy) then p entails the conjunction (the universal generalization, $\forall_\sigma y\, Xy$). The conjunction elimination rule corresponds to the converse of Tractarianism—if some premises entail the conjunction then those premises entail each conjunct. Of course, a universal generalization $\forall_\sigma y\, Xy$ entails each of its instances Xy—any provable material implication in H is a provable entailment in C—so the converse of Tractarianism is already a theorem of C and not in need of further proof.

Exercise 8.7. *Prove that Tractarianism is equivalent to* BF^σ.

We summarize these observations with another proposition:

Proposition 8.4. *The Barcan formula, Functionality and Tractarianism are equivalent.*

We can draw another non-obvious connection between the functionality principles from Section 6.5 and the modal principles discussed here. The principle of Functional Plenitude (and thus also Functional Choice) entails Brouwer's principle B. Thus the necessitation of Functional Plenitude entails the necessitation of B and consequently all of the principles discussed in this section.

Proposition 8.5. *Functional Plenitude entails* B.

We prove this in the following exercise.

Exercise 8.8. *a. Use Functional Plenitude show that for any distinct $x, y : \sigma$ there is a predicate $X : \sigma \to t$ such that $Xx =_t \top$ and $Xy =_t \bot$.*

b. Using the previous part derive ND^σ.

c. Let $\sigma = t$, and explain how you can derive B *from Functional Plenitude.*

As previously noted, each of the principles we have discussed in this section can be strengthened by adding a $\Box$ in front of it. We will write $\Box$B, $\Box\mathsf{BF}^\sigma$, $\Box\mathsf{ND}^\sigma$, and so on, to indicate these strengthenings. It is possible to show, using model-theoretic techniques, that the boxed versions of the principles really are stronger.[7] This means that the higher-order theory that results from adding any one of these principles to C (i.e. the set you get from adding these principles to C and closing under Gen and MP) is not closed under the rule of necessitation. On the other hand, if you add any of the boxed versions of these principles to C the result is closed under necessitation, for the same reason that C is: the resulting logic proves that the axioms are broadly necessary, and that the rules preserve broad necessity.

Remark 8.4. The reader who is familiar with propositional modal logic may be aware that $(\Diamond p \to \Box\Diamond\Diamond p) \to \Box(p \to \Box\Diamond p)$ is a theorem of S4, i.e. the necessitation of any instance of the B schema from propositional modal logic is entailed by a different unecessitated instance of B. This means that adding all instances of B to S4, closing under modus ponens but *not* necessitation, is the same as adding all instances of the necessitation of B—i.e. instances of $\Box(p \to \Box\Diamond p)$—to S4 (in either case we get the system S5).

Here we have stated that $\Box$B is strictly stronger than B: the apparent discrepancy arises because we are using the symbol 'B' not to refer to the set of instances of a schema, but to a single universally quantified statement $\forall_t p(p \to \Box\Diamond p)$ so that $\Box$B refers to $\Box\forall_t p(p \to \Box\Diamond p)$. Our observation above ensures that $\forall_t p(p \to \Box\Diamond p)$ entails $\forall_t p\Box(p \to \Box\Diamond p)$ in C, but does not let us infer $\Box\forall_t p(p \to \Box\Diamond p)$. The latter is stronger than the former since it governs both actual and merely possible instances of B; one needs BF^t to close the gap between the two. Similar remarks apply to other naming conventions from modal logic—in first-order modal logic, BF refers to a schema, whereas our BF^e refers a single axiom beginning with a second-order universal quantifier.

Note that the necessitation of Brouwer's principle implies all of the principles we have introduced in this section. One might call the result of adding $\Box$B to C 'S5 Classicism'.

Definition 8.3 (S5 Classicism)**.** *Let* C5 *be the smallest higher-order logic containing* C *and* $\Box$B.

S5 Classicism comes close to being the orthodox view of modality. For according to orthodoxy, metaphysical necessity has a modal logic of S5. If you combine this with the claim that it's metaphysically necessary that propositions are individuated by metaphysically necessary equivalence, one can show that metaphysical necessity is identical with the broadest necessity, and so the broadest necessity has a logic of S5 too.

Exercise 8.9. *Let $\Box_M$ stand for metaphysical necessity and assume it is a factive necessity (i.e.* $\mathrm{Nec}\,\Box_M \wedge \mathrm{T}\,\Box_M$*).*

a. Assume that it's metaphysically necessary that propositions are individuated by metaphysically necessary equivalence: $\Box_M(\forall_t pq(\Box_M(p \leftrightarrow q) \to p =_t q))$. Show $\Box_M\forall_t p(\Box_M p \leftrightarrow \Box_{\top} p)$.

b. Infer that $\Box_M = \Box_\top$.[iv]

One could also obtain **C5** directly from the assumption that metaphysical necessity is the broadest necessity, and the assumption that metaphysical necessity has a logic of **S5**. Nonetheless, there is significant interest in theories weaker than **S5** Classicism—one of the applications is outlined in the next section.

Does the logic of broad necessity have implications for the logic of other necessities? For some of the principles we have discussed, there are no obvious connections. Brouwer's principle can be true or false for broad necessity independently of whether it is true or false for other more restricted necessities.[8] Other claims about the modal logic of broad necessity do have implications for other necessities. A limiting case would be the modal axiom stating that all truths are necessary, $\forall_t p(p \to \Box p)$, which entails there is no contingency in any sense of contingency, and thus entails the corresponding principle for every other necessity.

Comprehension Check 8.1. *Appealing to the definition of* $\Box$*, convince yourself of the truth of* $\forall_t p(p \to \Box p) \to \forall_{t\to t} X(\mathrm{Nec}\, X \to \forall_t p(p \to Xp))$.

A more interesting example is the Barcan formula. Intuitively the Barcan formula BF^e rules out the possibility of 'new' individuals in the broadest sense of possibility, and so one might think that it consequently rules out the possibility of 'new' individuals in any more restricted sense of possibility. After all, if new individuals are possible in any sense they are possible in the broadest sense. But this is not quite so, because the Barcan formula can sometimes fail for necessities that are not infinitely closed, for reasons independent of the possibility of new individuals. Consider our earlier putative examples of necessities that are not infinitely closed. For instance, for each finite number *n* you ought to save more than *n* lives, but it's not true that you ought to: save more than *n* lives for every finite number *n* (because you can't). But of course, this failure does not mean it is permissible that there be more finite numbers than there in fact are. Nonetheless, we can show that the Barcan formula for broad necessity implies the Barcan formula for any infinitely closed necessity. Let us write BF^σ_X for the result of replacing $\Box$ with X in the statement of BF^σ.

Proposition 8.6. *Given the broad Barcan formula,* BF^σ*, every infinitely closed necessity satisfies the Barcan formula:*

$$\mathsf{BF}^\sigma \to \forall X(\mathrm{Nec}_\infty\, X \to \mathsf{BF}^\sigma_X)$$

Proof. Suppose that X is infinitely closed and that $\forall_\sigma x\, X(Fx)$. We want to show that $X(\forall_\sigma xFx)$. Since X is infinitely closed, it suffices to show that anything entailing every X-necessary proposition also entails $\forall_\sigma xFx$. Suppose r entails every X-necessary proposition. Since Fx is X-necessary for every x, $\forall_\sigma x.\Box(r \to Fx)$. By the broad Barcan formula, $\Box\forall_\sigma x(r \to Fx)$ and so $\Box(r \to \forall_\sigma xFx)$. Thus r entails $\forall_\sigma xFx$ as required. Since X is closed under entailment, $X(\forall_\sigma xFx)$. □

We next turn to our analysis of possible worlds. The possible worlds metaphysician is committed to certain modal theses that are not forced on us by our logic alone. The claim that P is possible if and only if it is true at some possible world corresponds to a thesis in higher-order logic—that P is possible if and only if it is entailed by a world proposition. We will call this the 'Leibniz Biconditonal'. Of course, since we have two notions of world, there are two versions of the Leibniz biconditionals—the weak Leibniz biconditionals (WL) and the strong Leibniz biconditionals (SL)—depending on whether we use the weak or strong notion of world. We will begin with the strong Leibniz biconditionals.

$\mathbf{SL}^t$ $\forall_t p(\Diamond p \leftrightarrow \exists_t w(\mathrm{SWorld}\, w \wedge w \leq p))$

Observe that the right-to-left direction is already a theorem of C since world propositions are possible by definition, and you cannot be entailed by a possible proposition unless you are possible. However, the left-to-right direction is not.

Like other principles we have considered in this section, there is a variant of this principle for each (relational) type σ. Let us say that a strong world property is one that is possibly instantiated and necessarily entails every property or its negation. Similar definitions are possible at any other relational type σ distinct from e:

$$\Diamond_\sigma := \lambda X.(X \neq_\sigma \bot_\sigma)$$

$$\mathrm{SWorld}_\sigma := \lambda X.(\Diamond_\sigma X \wedge \Box\forall_\sigma Y(X \leq_\sigma Y \vee X \leq_\sigma \neg_\sigma Y)$$

Later it will be convenient to have a notion of 'accessibility' between worlds, which we'll define as follows:

$$\mathrm{Accessible}_\sigma := \lambda XY.(X \leq_\sigma \Diamond_\sigma Y)$$

A possible worlds metaphysician might be tempted to explain the concept of a strong world property as something that is instantiated at exactly one possible world and by exactly one possible individual. While this may be a useful heuristic, it is not possible to quantify over merely possible individuals so one has to work with the official definition in arguments.

Exercise 8.10.

a. Prove in C *that if* $W : \sigma \to t$ *is a strong world property, then it is necessarily instantiated by at most one thing:* $\Box\forall_\sigma xy(Wx \wedge Wy \to x =_\sigma y)$.[v]

b. Prove in Classicism that it's necessary that true (weak or strong) world propositions are identical: $\forall_t wv((\mathrm{World}\, w \wedge \mathrm{World}\, v \wedge w \wedge v) \to w =_t v)$.

Our general principle states that a property, relation, etc. is possible if and only if it is entailed by a strong world property, relation, etc. Thus, for relational types σ distinct from e:

$\mathbf{SL}^\sigma$ $\forall_\sigma X(\Diamond_\sigma X \leftrightarrow \exists_\sigma W(\mathrm{World}_\sigma\, W \wedge W \leq_\sigma X)$

It will be convenient in what follows to talk about a proposition w 'settling' other propositions when w entails either that proposition or its negation. A strong world proposition, for instance, is a proposition that necessarily settles all propositions. The relation of entailment, between a world proposition and an arbitrary proposition is much like the notion of *truth at world* that is appealed to by the possible worlds metaphysician. According to that metaphysician, a statement is necessary at world w if and only if it is true at every world accessible to w. Given $\Box\mathrm{SL}^\sigma$ we can prove left-to-right direction of this claim by interpreting *truth at* in terms of entailment; the right-to-left can be derived in a strengthening of Classicism to be discussed shortly. The full clauses for truth at a world stated in terms of entailment become the following. For any world proposition, w:

(W¬) $w \leq \neg p$ if and only if $w \not\leq p$

(W∧) $w \leq (p \wedge q)$ if and only if $w \leq p$ and $w \leq q$

(W□) $w \leq \Box p$ if and only if $v \leq p$ for every world proposition v accessible to w

Where v is accessible to w when $w \leq \Diamond v$.

Exercise 8.11.

a. Prove ($W\neg$).

b. Prove ($W\wedge$).

c. Assuming $\Box SL^\sigma$, prove the left-to-right direction of ($W\Box$).[vi]

The left-to-right direction of SL^t (and SL^σ generally) is not a theorem of Classicism, and it is instructive to think about how it could fail. One way it could fail would be if p were an 'atomless' consistent proposition: every consistent proposition entailing p is entailed by a stronger consistent proposition. This ensures that no consistent propositions entailing p settle all propositions, for it will always leave a strictly stronger consistent proposition unsettled. In the theory of Boolean algebras an element which is consistent and entails every other element or its negation is called an *atom*. An atom is clearly just another word for a weak world. We can introduce a corresponding concept for relational types σ distinct from e:

$$\mathrm{WWorld}_\sigma := \lambda W.(\Diamond_\sigma W \wedge \forall_\sigma X(W \leq_\sigma X \vee W \leq_\sigma \neg_\sigma X)$$

Recall that the only difference between a strong world and a weak world is that a weak world needs only to settle the propositions there actually are, whereas a strong world must necessarily settle all propositions, including possibly new propositions. In the theory of Boolean algebras, a Boolean algebra is atomic iff every consistent element is entailed by an atom. The analogue of this atomicity principle for Classicism is the principle we earlier called the weak Leibniz biconditionals, obtained from SL^σ by replacing the notion of a 'strong world' with that of a 'weak world', i.e. an atom. Here again σ may be substituted for the type of a proposition, property, relation, etc.—i.e. any relational type apart from e:

WL$^\sigma$ $\forall_\sigma X(\Diamond_\sigma X \leftrightarrow \exists_\sigma W(\mathrm{WWorld}_\sigma\ W \wedge W \leq_\sigma X))$

As with the strong Leibniz biconditionals, the right-to-left direction is already a theorem of Classicism, and it is the left-to-right direction that is substantive. (Bacon and Dorr (forthcoming) call this direction Atomicity by analogy with the corresponding condition on Boolean algebras.)

The second way SL^t could fail with respect to a proposition p, even granting WL^t, would be if it were possible for there to be new propositions that the actual atoms entailing p do not settle. That is, for every weak world entailing p, it is possible to decompose that weak world into a disjunction of stronger consistent propositions; and since a weak world settles all actually existing propositions this possible decomposition of an actual weak world would have to involve new propositions that do not actually exist. Thus failures of the propositional Barcan formula provide a way for the strong possible worlds picture to fail. Indeed, with one more principle—see Rigid Comprehension below—it is possible to show that, for relational types σ distinct from e, $\Box SL^\sigma$ is equivalent to the combination of $\Box BF^\sigma$ and $\Box WL^\sigma$; see Theorem 8.2 below. The right-to-left direction may be proved directly in Classicism and follows from the corresponding implication between the unboxed versions of these principles:

Exercise 8.12. *Assume WL^σ and BF^σ for all relational types distinct from e. Prove that all weak worlds are strong worlds, and conclude that SL^σ holds for these types.*

Thus the existence of strong possible worlds is tied up with the Barcan formula for propositions, properties and relations (but not individuals). In light of this, one might be interested in exploring the notion of a weak world instead as a foundation for possible world metaphysics that is more friendly to the possibility of new propositions, properties and relations,

and so compatible with the failure of the propositional Barcan formula. Indeed, this is the notion of world adopted by Prior in Prior and Fine (1979) p. 43.[9]

Remark 8.5. There is a very natural model theory for the weak Leibniz biconditionals without BF which generalizes Kripke's semantics for S4 by replacing the transitive reflexive Kripke frames with 'categories' in which one world can be accessible to another in 'several different ways', and propositions are modelled by sets of these ways of accessing other worlds. Versions of this idea are explored in Bacon and Dorr (forthcoming) and in Chapter 17. However, the fact that the most natural models depart from the usual Kripke semantics suggest the underlying metaphysical picture is importantly different from the more familiar possible worlds metaphysics.

In the other direction, we required strong worlds to be possible, but we didn't require them to be necessarily so; without B a proposition may be possible without being necessarily so. One could consider an even stronger notion of world: a strong world that is *necessarily* possible. This is equivalent to necessarily being a weak world. Many uses of possible worlds semantics for systems weaker than S5 do not require world propositions to be necessarily possible. The following exercise illustrates why:

Exercise 8.13. *Consider the analogue of the Leibniz biconditionals for the notion of a necessary weak world:* $\Box\forall_t p(\Diamond p \leftrightarrow \exists_t w(\Box(\mathrm{WWorld}\, w) \wedge w \leq p))$. *Prove the* 5 *principle:* $\forall_t p(\Diamond p \rightarrow \Box\Diamond p)$.

Corresponding to variations in these definitions, there are also variations of possible world semantics depending on: (i) whether you use accessibility relations to model the modalities or simply quantify unrestrictedly over worlds, and (ii) whether you let the propositional quantifiers range over all sets of possible worlds or only some. These different variations will have different validities.[10] Our hierarchy of definitions is one that naturally corresponds to the sorts of possible worlds semantics that allows non-universal accessibility relations and treats the propositional quantifiers as ranging unrestrictedly over sets of worlds.

Exercise 8.14. *In Prior (1967), pp. 79–82, Arthur Prior introduces yet another definition of a world proposition:* $\mathrm{PriorWorld} := \lambda w.\Diamond(w \wedge \forall p(p \rightarrow w \leq p))$. *Given S5 Classicism, prove that* $\mathrm{PriorWorld} =_{t\rightarrow t} \mathrm{World}$.

Exercise 8.15. *Explain how a proposition could be a Prior World but not a World if we are not assuming S5. (There are two sorts of examples here: one exploits failures of* B *but not* G*, the other exploits failures of* G*. You may thus assume the negation of either of these axioms in answering this question.)*

Our focus has been the notion of a broadly possible world (weak or strong). There are principles corresponding to the Leibniz biconditionals governing other notions of possibility: that a proposition is physically possible if and only if it is true at some physically possible world, that a proposition is metaphysically possible if and only if it is true at a metaphysically possible world, and so on. For any given necessity, X, we can introduce the notion of an X-possible world: a broadly possible world that is X-possible. Recall that we earlier wrote $\langle X\rangle$ for the dual of a necessity X, $\lambda p.\neg(X(\neg p))$:

$$\mathrm{SWorld}_X := \lambda w(\langle X\rangle w \wedge \mathrm{SWorld}\, w)$$

$$\mathrm{WWorld}_X := \lambda w(\langle X\rangle w \wedge \mathrm{WWorld}\, w)$$

It's natural to ask whether we can derive the corresponding kind of weak or strong Leibniz biconditional for X-possibility from the corresponding Leibniz biconditional for broad necessity. It turns out that we cannot for an arbitrary necessity, nor is it desirable. Consider our earlier example of having chance 1, and suppose that there is a chance of one that a given dart will land on the dart board, but zero chance it will land on any particular point. Since there is a non-zero chance of the dart landing on the dartboard, this proposition is chance-possible. But any weak or strong world proposition will entail a zero-chance proposition that says exactly where the dart will land, and so no broadly possible world will be chance-possible. This style of counterexample rested explicitly on the fact that this necessity is not infinitely closed. The Leibniz biconditionals for infinitely closed necessities, by contrast, *do* follow from the corresponding Leibniz biconditionals for broad necessity.

Proposition 8.7. *Let X be an infinitely closed necessity. Then the strong and weak Leibniz biconditionals respectively entail the corresponding principle for X necessity:*

$$\langle X\rangle p \leftrightarrow \exists w(\mathrm{SWorld}_X\, w \wedge w \leq p)$$

$$\langle X\rangle p \leftrightarrow \exists w(\mathrm{WWorld}_X\, w \wedge w \leq p)$$

Proof. Both implications are proved in the same way. From the definition of being infinitely closed, anything entailed by the X-necessities must be itself X-necessary. So any X-possibility is such that its negation is not entailed by the X-necessities.

Thus if $\langle X\rangle p$, $X \not\leq_{ts} \neg p$. That is, for some r such that $\forall q(Xq \rightarrow r \leq_t q)$, $r \not\leq_t \neg p$. This means $\langle X\rangle(r \wedge p)$, so by WL^t (SL^t), there is a weak (strong) world proposition w that entails $r \wedge p$. We finally can see that w must be X-possible. For if not, then $X\neg w$, and since r entails every X-necessity, $r \leq \neg w$. But since $w \leq r$, $w \leq \neg w$, contradicting the assumption that w is a weak (strong) world.

The right-to-left direction is obvious. □

The final principle we will consider in this section states that every property or relation is coextensive with a 'modally rigid' property or relation. Our paradigm example of a rigid property is a property expressed by a disjunction of identities, such as: *being identical to either Alice, Bob or Carol*, i.e. $\lambda x.(x =_e a \vee x =_e b \vee x =_e c)$. Call this property F. The purpose of the notion of rigidity is to generalize this beyond the finite. Observe that the property F of being Alice, Bob or Carol cannot shrink or expand across modal space. It cannot shrink, because if you are identical to one of Alice, Bob or Carol, you are necessarily so, by the necessity of identity. It cannot expand because it is impossible that anything other that Alice, Bob or Carol have F. It might be tempting to characterize the general idea that a rigid property cannot expand in the same way we characterized it as not shrinking: by saying that if something is not identical to Alice, Bob or Carol then it is necessarily so. But to justify this in a parallel fashion would require the necessity of distinctness, which is not a theorem of Classicism. In fact, if David is distinct but possibly identical to Carol, then David in fact doesn't have F but he could have. On the other hand, note that if there were a property which each of Alice, Bob and Carol had necessarily, but was such that it was possible that something F didn't have it, then F must have expanded to include something other than Alice, Bob or Carol. So the proper way to express the non-expandingness of rigid properties X is by saying this can't happen—i.e. by the necessitated Barcan formula for X-restricted quantification. Let:

$$\forall_\sigma^X := \lambda Y.\forall_\sigma z(Xz \rightarrow Yz)$$

$$\text{Inextensible} := \lambda X.\Box\forall_{\sigma\to t}Z(\forall^X_\sigma y\Box Zy \to \Box\forall^X_\sigma y\, Zy)$$

Note that the idea that rigid properties don't shrink may be characterized with the converse Barcan formula for X-restricted quantification:

$$\text{Incontractible} := \lambda X.\Box\forall_{\sigma\to t}Z(\Box\forall^X_\sigma y\, Zy \to \forall^X_\sigma y\,\Box Zy)$$

Observe that $\text{Incontractible}\, X$ is equivalent to $\Box\forall_\sigma y(Xy \to \Box Xy)$ (the left-to-right direction follows by instantiating Z with X). Thus Rigidity may be formulated as the conjunction of both of these conditions—or equivalently, replacing the conditional in either of the conditions with a biconditional. Indeed, we can introduce parallel notions of Rigidity for relations of arity two or more. Here is a fully general definition for relations of type $\overline{\sigma} \to t$:

$$\text{Rigid}_{\bar\sigma\to t} := \lambda X.\Box\forall_{\bar\sigma\to t}Z(\forall^X_{\bar\sigma}\bar y\Box Z\bar y \leftrightarrow \Box\forall^X_{\bar\sigma}\bar y\, Z\bar y)$$

Although we have motivated our definition in terms of disjunctions of identities, arguably properties expressed using plural terms, such as *being one of those*, or *being one of the Beatles* are rigid in the sense defined above as well. Since every property, $X : e \to t$, can be rigidified using plural terms to make the property *being one of the Xs* we should expect every $e \to t$ property to be coextensive with a rigid property. Generalizing this to all types, our principle may now be stated.

Rigid Comprehension $\forall_{\bar\sigma\to t}X\exists_{\bar\sigma\to t}Y(\text{Rigid}_{\bar\sigma\to t}\, Y \wedge \forall_{\bar\sigma}\bar x(X\bar x \leftrightarrow Y\bar x))$

Rigid Comprehension is quite powerful. In the following exercises, you will explore two of its consequences. The first exercise concerns the notion of a greatest lower bound which we encountered earlier in our discussion of Tractarianism. The official definition of $p : t$ being a lower bound and a greatest lower bound of $X : t \to t$ are as follows:

$$\text{LB} := \lambda Xp.\forall_t r(Xr \to p \leq r)$$

$$\text{GLB} := \lambda Xp.(\text{LB}\, Xp \wedge \forall_t p'(\text{LB}\, Xp' \to p' \leq p)$$

Let Boolean Completeness be the claim that every property $X : t \to t$, representing a collection of propositions, has a greatest lower bound:

Boolean Completeness $\forall_{t\to t}X\exists_t p.\,\text{GLB}\, Xp$

Exercise 8.16. *In this exercise you will show derive Boolean Completeness from Rigid Comprehension.*

a. Show in Classicism that for any rigid collection of propositions, $X : t \to t$, $\forall_t p(Xp \to p)$ is a greatest lower bound of X: $\text{GLB}\, X\forall_t p(Xp \to p)$.

b. Prove Boolean Completeness from Rigid Comprehension in Classicism.

Let Actuality be the claim that there is a truth that entails all truths:

Actuality $\exists_t w\forall_t p(p \to w \leq p)$

Exercise 8.17. *Prove Actuality from Rigid Comprehension.*

We end the section by proving the promised claim that the necessitated strong Leibniz biconditionals imply $\Box\mathsf{BF}^\sigma$ for $\sigma \neq e$.

Theorem 8.2. *Given Rigid Comprehension, the schema $\Box SL^\sigma$ is equivalent to the schemas $\Box\mathsf{BF}^\sigma$ for $\sigma \neq e$ and $\Box WL^\sigma$ together.*

Proof. The right-to-left direction was an exercise, and clearly SL^{σ} entails WL^{σ} since every world is an atom.

The strategy is to use the instance $\mathrm{SL}^{\sigma\to t}$ of the strong Leibniz biconditionals to prove BF^{σ}. For readability, we will focus on the case where $\sigma = t$, but we indicate how the proof generalizes below.

Suppose that $\Diamond\exists_t p.Xp$. Thus $X : t \to t$ is possible (i.e. has $\Diamond_{t\to t}$), and so $\mathrm{SL}^{t\to t}$, is entailed by a world property $W : t \to t$. Now let H be the rigid collection of world properties, V that are accessible to W (i.e. such that $W \leq_{\sigma} \Diamond_{t\to t} V$). Intuitively H contains properties that are instantiated at exactly one possible world, are instantiated uniquely, and are instantiated by the same possible proposition that has W. Thus the disjunction of the H properties is the 'haecceity' of a possibly X proposition. We can use this actually existing haecceity to define an actual proposition, r, that is possibly W, and thus possibly X: namely the proposition that some proposition possessing the haecceity is true. This proposition is necessarily equivalent, and thus identical to any proposition possessing the haecceity. Stating this directly in terms of H, the proposition r says that one of the world properties having H applies to a truth:

$$r := \exists_{t\to t} V \exists_t p(HV \wedge Vp \wedge p)$$

We establish $\Diamond Wr$ in several steps. (i) Firstly we show that if p is a W proposition then, necessarily, one of the world properties in H applies to p. It follows that if p is W, then necessarily, if p is true, then one of the world properties in H applies to a truth, namely p. This establishes $Wp \to \Box(p \to r)$. (ii) We prove that if p is a W proposition then, necessarily, if any world property in H applies to a proposition that proposition is p. Thus if one of the world properties in H applies to a truth, then p is true. This establishes $Wp \to \Box(r \to p)$. Putting this together we have proven $\forall_t p(Wp \to p =_t r)$ from necessary assumptions, and thus we may conclude $\Box\forall_t p(Wp \to p =_t r)$. Finally, since W entails X, i.e. $\Box\forall_t p(Wp \to Xp)$, it follows that $\Diamond Xr$ as required.

So to complete the argument we need to prove (i) and (ii).

We begin by proving (i), namely $Wp \to \Box\exists_{t\to t} V(HV \wedge Vp)$. Suppose that Wp. Given $\Box\mathrm{SL}^{t}$ it suffices to prove that every world proposition v entails $\exists_{t\to t} V(HV \wedge Vp)$, so let v be an arbitrary world proposition. It may be checked that $U := \lambda q(v \wedge q = p) : t \to t$ is a world property. Also $W \leq \Diamond_{t\to t} U$, since W cannot entail $\neg_{t\to t}(\Diamond_{t\to t} V)$ because p is both W and possibly U. So U is H, and thus necessarily H by the rigidity of H. $v \leq HU$. On the other hand $v \leq v$ and $v \leq p = p$ so $v \leq (\lambda q(v \wedge p =_t q))p$ so $v \leq Up$. Putting this together we have $v \leq (HU \wedge Up)$, and finally $v \leq \exists_{t\to t} V(HV \wedge Vp)$ as required.

Next we prove (ii), namely $Wp \to \Box\forall_{t\to t} V(HV \to \forall_t q(Vq \to p =_t q))$. Given the rigidity, and thus inexpandibility of H, it suffices to show $Wp \to \forall_{t\to t} V(HV \to \Box\forall_t q(Vq \to p =_t q)$, so suppose that Wp and HV. Since Wp and W entails $\Diamond_{t\to t} V$ it follows that $\Diamond Vp$. So V cannot entail $\lambda q(q \neq_t p)$, and since V is a world property, V entails $\lambda q(p =_t q)$ as required.

This completes the proof for the case where $\sigma = t$. In the general case, since σ is a relational type distinct from e, we may suppose that $\sigma = \sigma_1 \to \ldots \to \sigma_n \to t$. We proceed as before: given $\Diamond_{\sigma} y.Xy$ we choose a world property $W : \sigma \to t$ entailing X, and define H to be the rigid collection of world properties of type $\sigma \to t$ that are accessible to W. Then the relation R that witnesses $\exists_{\sigma} y \Diamond Xy$ is the relation that holds between $x_1 : \sigma_1 \ldots x_n : \sigma_n$ iff some H contains a world property which contains a relation instantiated by $x_1 \ldots x_n$:

$$R := \lambda \bar{x} \exists_{\sigma\to t} V \exists_{\sigma} Y(HV \wedge VY \wedge Y\bar{x})$$

□

Using this result we can now finish our argument that the possible world clauses hold with respect to our notion of world and truth at:

Theorem 8.3. *Given the strong Leibniz biconditionals and Rigid Comprehension one can prove that, necessarily, for any strong world w, and proposition p:*

(W¬) $w \leq \neg p$ if and only if $w \not\leq p$
(W∧) $w \leq (p \wedge q)$ if and only if $w \leq p$ and $w \leq q$
(W□) $w \leq \Box p$ if and only if $v \leq p$ for every v accessible to w

Exercise 8.18. *Finish the proof of (W□) from $\Box SL^{\sigma}$ and Rigid Comprehension: show that if $v \leq p$ for every v accessible to w, $w \leq \Box p$. You may assume Theorem 8.2.*

8.3 Logical necessity

In the last section, we considered an argument for **S5** Classicism from the assumption of **S5** for metaphysical necessity and either the assumption that metaphysical necessity is the broadest necessity, or the assumption that propositions are individuated by metaphysically necessary equivalence. However there are many positions in the metaphysics of modality that imply the existence of necessities broader than metaphysical necessity, thus rejecting those auxiliary assumptions and leaving the status of **S5** for broad necessity open. For instance:

1. The view that there can be non-vacuous counterfactuals with metaphysically impossible antecedents implies that counterfactual necessity—i.e. having a negation which counterfactually implies everything, $\lambda p.\forall_t q(\neg p \mathrel{\square\!\!\rightarrow} q)$—is broader than metaphysical necessity.[11]
2. The view that the **S4** principle fails for metaphysical necessity implies that metaphysically necessary metaphysical necessity, $\lambda p.\Box_M(\Box_M p)$, is broader than metaphysical necessity.[12]
3. The view that there is a distinctive propositional modality in which contingent *a priori* truths like *it's raining if and only it's actually raining* are necessary is one in which metaphysical necessity is not the broadest necessity.[13]
4. Certain common views about the tense operators imply that metaphysical necessity is not broader than *always*.[14]
5. The view that there could be propositional (or 'metaphysical') indeterminacy, along with the idea that the vague metaphysically supervenes on the precise implies that metaphysical necessity is not broader than determinacy.[15]

However, there are certain views that not only block the arguments for **S5** Classicism by rejecting the identification of broad necessity with metaphysical necessity, but provide us with positive reasons to reject it. In this section, we will be investigating views that take seriously the idea of a propositional notion of logical necessity: not merely the metalinguistic property of being a logically true sentence, but something that would be more appropriately expressed using a sentential operator and which, as it were, "stands to reality as logical truth stands to language". Versions of this idea trace back at least to Bolzano (1929) and have been explored by several authors since including McKinsey (1945), McFetridge (1990), Hale (1996), and Rumfitt (2010). McFetridge claimed that logical necessity is the broadest kind of

necessity, which in the framework of Classicism means also identifying it with the operator of *having every necessity* (i.e. $\Box$) and with *being identical to* $\top$ (i.e. $\Box_\top$). Given McFetridge's thesis, the status of **S5** Classicism rests on the status of **S5** for logical necessity. But even if it is not the broadest necessity, there are other routes to refuting **S5** Classicism. If the analogue of the necessity of distinctness fails for logical necessity (or indeed any necessity), it will also fail for the broadest necessity giving us a route to argue against **S5** Classicism. For instance, if two things are distinct but logically possibly identical, then they are distinct but possibly identical in the broadest sense of possible. Similarly, if the Barcan formula fails for logical necessity (or any infinitely closed necessity), by Proposition 8.6

The logic of logical necessity, unlike metaphysical necessity, appears to be weaker than **S5**. We begin with a few considerations against the B axiom that stems from the assumption that logical possibility is very broad—for instance, below we consider one conception of logical necessity in which every consistent extension of **C** should correspond to a logical possibility. On this view Extensionalism, which can be axiomatized by adding the Fregean axiom—$\forall_t pq(p \leftrightarrow q \rightarrow p =_t q)$—to **C**, should be logically contingent, since both it and its negation are consistent with **C**. But if Extensionalism is logically, and thus broadly possible then the B axiom cannot be true:

Exercise 8.19. *Let* F *stand for the Fregean axiom*

a. Prove $\Box(\mathsf{F} \rightarrow \Box\mathsf{F})$ *in* **C**.

b. Assume that the Fregean axiom is broadly contingent ($\Diamond\mathsf{F} \wedge \Diamond\neg\mathsf{F}$*). Explain why it is both false and possibly necessary, contradicting Brouwer's axiom.*

The invalidity of B on this conception of logical necessity can be illustrated with many other sentences. Suppose, for the sake of argument, that there are only finitely many individuals so that the first-order sentence saying there are exactly n things is true. But this is surely logically contingent: it is consistent in **C**, and thus logically possible, that there be $n + 1$ things, and that there be $n - 1$ things. The only way for there to be more things than there actually are is for there to be new objects not among the finite list of actual individuals, so the Barcan formula, and thus $\Box\mathsf{B}$ fails.[16] And the only way for there to be less things than there are, given Broad Necessitism, is for two distinct things to be come identical, so ND^e and thus B fails.

Possibly the earliest philosopher to take logical necessity seriously was Bolzano. Indeed, Bolzano's account of logical necessity later became the basis for the metalinguistic notion of logical truth ubiquitous in mathematical and philosophical logic canonized in Tarski (1936b). According to Bolzano, a proposition is logically necessary iff every proposition that results from uniformly replacing the simple individuals, properties, relations, operators etc. that appear in the proposition, but leaving the logical operations alone, is true. Bolzano's account thus bears a resemblance to a certain metalinguistic definition of a logical truth in which a logical truth is a sentence whose substitution instances are all true, although we are not substituting non-logical constants for other expressions of the same type, but simple individuals, properties and relations with other individuals, properties and relations. This idea of a 'metaphysical substitution' has natural applications in the model theory of higher-order logic, which we will explore further in Section 17.2. In higher-order logic we can already make this idea of arbitrary replacement precise using quantification into the positions of names, predicates, operators and so on. Thus, for instance, the logical necessity of $Fa \rightarrow Fa$ should be equivalent to the universally quantified statement

$\forall_{e\to t}X\forall_e y(Xy \to Xy)$.[17] Thus Bolzano's theory of logical necessity could be formalized as a schema in higher-order logic as follows:

Logical Necessity $\Box A(\bar{c}) \leftrightarrow \forall_{\bar{\sigma}}\bar{x}A(\bar{x})$

Here $A(\bar{c})$ is a closed formula whose non-logical constants are enumerated $\bar{c} = c_1, \ldots, c_n$, without repetition, and $A(\bar{x})$ is the result of replacing each constant with a distinct variable of the same type. This schema can be interpreted in **C** by identifying $\Box$ with broad necessity. Despite Bolzano's apparent assumption of a structural theory of propositions, Logical Necessity is consistent in **C**—a consequence of the coalesced sum construction described in Chapter 18.[18]

Note that Bolzano's theory appears to assume something like a structured theory of propositions, and only simple properties, relations, etc are available for substitution. We thus cannot apply our principle unless the simple expressions we are quantifying out denote simple entities. For instance, in our analysis of the logical truth of $Fa \to Fa$, 'F' and 'a' must be simple things. If we had a sentential constant 'P' standing for the logically complex proposition $Fa \to Fa$ and we unthinkingly applied Logical Necessity we'd get that the logical necessity of P amounts to the falsehood $\forall_t p.p$. Yet of course P just is $Fa \to Fa$ so should be logically necessary iff $Fa \to Fa$ is. These sorts of problems do not arise if we are working in what Bertrand Russell calls a 'logically perfect' language, where there is a one-to-one correspondence between the syntactically simple expressions of the language (the constants) and the metaphysically simple entities; for then every constant denotes a simple entity of the appropriate type, and no two constants denote the same entity.

Remark 8.6. Bolzano is often reported as having given us a *definition* of logical truth in terms of truth and (metaphysical) substitutions. However, this is not so: for *John to be logically necessarily tall* would have to be defined in terms of the truth of the propositions resulting from substituting *John* and *being tall* for other individuals and properties. But among the properties we could substitute for *being tall* is *being logically necessarily tall*, so among the definiens of *John's being logically necessarily tall* would be that very proposition itself. Bolzano's account of logical necessity should thus be understood as a constraint, not a definition. Since the constraint is circular it is highly non-obvious that it is satisfiable. A version of this satsifiability problem in the context of propositional modal logic appeared in Harvey Friedman's list of open problems in logic.[19]

Prior to Kripke's work on modal logic, McKinsey (1945) gave an interpretation of propositional modal logic inspired by Bolzano's account of logical necessity. McKinsey showed, given some minimal assumptions about the behaviour of substitutions, that logical necessity should have a modal logic of at least **S4**.

Bolzano's substitutional interpretation of logical necessity invalidates B. This point was originally made by McKinsey, with regard to his own interpretation of propositional modal logic inspired by Bolzano's account of logical necessity in Bolzano (1929). McKinsey points out that if B were valid, then the following inference would be valid:[20]

1. Sugar is sweet and vinegar is not sweet.
2. It's logically necessary that it's logically possible that sugar is sweet and vinegar is not sweet.
3. It's logically possible that sugar is sweet and sugar is not sweet.

The step from 1 to 2 is licensed by B, and the step from 2 to 3 because 3 is a substitution instance of *it's logically possible that sugar is sweet and vinegar is not sweet*.

Exercise 8.20.

a. Using the schema Logical Necessity prove that there is a truth which is possibly necessarily false. You may assume the signature contains any constants you like.[vii]

b. Assume $c \neq_\sigma d$ where $c, d : \sigma$ are two constants. Using Logical Necessity prove that $\Diamond c =_\sigma d$, illustrating a failure of ND^σ.

Observe, however, that the Bolzanoean theory does not substantiate our earlier assumption that Extensionalism is logically possible. Indeed, Logical Necessity implies that any claim that is formulated in purely logical vocabulary—such as the Fregean axiom, or any of the other principles we considered in Section 8.2— is logically necessary or logically necessarily false. This is because if A is a purely logical claim, then $\Box A \leftrightarrow A$ is an instance of Logical necessity where $\bar{c}$ and $\bar{x}$ are the empty sequence, $\forall \bar{x} A$ is just A).

According to another type of account, logical necessity can be (at least partially) characterized by the theorems of a given logic. For any sentence 'A' consistent with the logic in question, this type of account asserts that it's logically possible that A. For any higher-order logic **L** extending **C**, we have the following schema:

Logical Possibility$^{\mathbf{L}}$ $\Diamond A$, where A is a closed sentence consistent with **L**.

For similar reasons to before, we must assume this logic is formulated in a logically perfect language.

When the logic in question is **C**, we can substantiate our previous assumption that Extensionalism is logically contingent, since the principle of Extensionality and its negation are both consistent in **C**. Although accounts of logical necessity based on Logical Possibility$^{\mathbf{C}}$ thus diverge from Bolzano's substitutional account in this way, observe that since logics are closed under the rule of substitution the left-to-right direction of Logical Necessity, $\Box A(\bar{c}) \rightarrow \forall \bar{x} A(\bar{x})$, will be a consequence in **L** of Logical Possibility$^{\mathbf{L}}$. When $A(\bar{c}) \notin \mathbf{L}$ then $\Diamond \neg A(\bar{c})$ is an instance of Logical Possibility$^{\mathbf{L}}$, and so $\Box A(\bar{c}) \rightarrow \forall \bar{x} A(\bar{x})$ will be a consequence of $\Diamond \neg A(\bar{c})$ due to its entailing the negation of its antecedent. Otherwise $A(\bar{c}) \in \mathbf{L}$ and thus so is $A(\bar{x})$, and finally $\forall \bar{x} A(\bar{x})$ by the rule of substitution and generalization so that the remaining instances of the conditional will also belong **L**.

Observe that Logical Possibility$^{\mathbf{L}}$ is not consistent in **L** for every choice of logic **L** extending **C**. For instance, not only are the Fregean axiom and its negation both consistent in **C**, they are also consistent in **C**5, so Logical Possibility$^{\mathbf{C5}}$ asserts the contingency of the Fregean axiom and thus implies the negation of B—a contradiction in **S5** Classicism.

Definition 8.4 (Coherent logic)**.** *We say that a logic* **L** *is* coherent *iff the theory that results from adding Logical Possibility$^{\mathbf{L}}$ to* **L** *is consistent.*

Exercise 8.21.

a. Show that Extensionalism, as formulated in the purely logical signature $\mathcal{L}(\Lambda)$, is coherent.

b. Explain why Extensionalism can fail to be coherent in non-empty signatures.

The concept of a coherent modal logic was introduced in Meyer (1971) and Fine (unpublished), and was related by both to the disjunction property of modal logics—a notion that has analogues for higher-order logics extending **C**:

Definition 8.5 (Disjunction property)**.** *Let* **L** *be an extension of* **C**. **L** *has the disjunction property if and only if:*

For any formulas A and B, if $\Box A \vee \Box B \in \mathbf{L}$ *then* $A \in \mathbf{L}$ *or* $B \in \mathbf{L}$.

The reader may check for themselves that a logic extending **C** has the disjunction property if and only if it is coherent. A central question to the study of logical necessity in Classicism thus asks whether there are any coherent logics. This is settled using model-theoretic means in Chapter 18: we will show that a large class of logics extending **C**, including **C** itself, are coherent. Observe finally that Logical Possibility$^{\mathbf{L}}$ entails the following schema:

Distinctness$^{\mathbf{L}}$ $M \neq_\sigma N$ whenever $M \neq_\sigma N$ is closed and consistent with **L**

for if $M \neq_\sigma N$ is closed and consistent in **L**, then $\Diamond(M \neq_\sigma N)$ is an instance of Logical Possibility$^{\mathbf{L}}$ from which $M \neq_\sigma N$ follows by the necessity of identity. This schema essentially tells us that reality is as fine-grained as possible, consistent with the identities that logic gives us (taking **L** to represent the 'true logic' whatever that might be). The schema Distinctness$^{\mathbf{L}}$ is interesting because unlike Logical Possibility$^{\mathbf{L}}$, it is not formulated modally and so is a natural schema to add to logics **L** that do not extend **C**, and encodes a fine-grained vision of reality.[21] Fine-grained theories of reality contradicting Classicism will be the subject of Chapters 11–13.

8.4 Further reading

While the approach of the previous two chapters has been to work through a particular position in the philosophy of higher-order modal logic, this is a rich and active area and many interesting topics have been omitted from our discussion. This section gives a very brief survey of some of these other strands of research.

The tradition of combining modal and higher-order logic can be traced back to the origins of modal logic, beginning with C.I. Lewis, where languages containing modal connectives and quantification into sentence position were studied (Lewis and Langford, 1959). Interest in this particular fragment of higher-order logic with modal operators has persisted in large part due to the work of Arthur Prior and Kit Fine.[22] Early work on modal logic would often include more expressive fragments of higher-order logic, such as second-order quantifiers (see Barcan (1947) and Carnap (1947), pp. 181–182), but the full power of higher-order and modal ideas together is perhaps most fully on display in Montague (1973) where these ideas are applied in the study of natural language semantics.[23]

Recently, higher-order modal logic has received a renewed interest since the publication of Timothy Williamson's book *Modal Logic as Metaphysics* (Williamson, 2013). The system Williamson adopts is essentially that of Gallin (1975): it is a higher-order language containing a primitive necessity operator constant, which Williamson interprets as metaphysical necessity. Williamson and Gallin both describe model theories for their languages that validate **S5** Classicism when restricted to the logical fragment of the language without the primitive necessity operator (modulo minor differences between their type systems and ours; see the discussion of relational type theory in Section 1.4). Given a hierarchy of logical definitions analogous to the ones presented in Chapter 7, including that of *being a necessity* and *broad necessity*, the models also validate the identity of their primitive necessity with the broadest necessity as defined in Chapter 7. These books provide a good introduction to the logic and model theory of **S5** Classicism, and Williamson, especially to the philosophical issues that surround it. The higher-order logic of Williamson and Gallin is essentially characterized model-theoretically, but it is also possible to derive many of the distinctive features of the Gallin/Williamson higher-order logic axiomatically. Bacon and

Dorr (forthcoming) show, for instance, show that one valid principle of Williamson and Gallin, namely □Functional Choice, implies (in Classicism) all of the principles of S5, the Strong Leibniz Bicondionals, Rigid Comprehension, Boolean Completeness, and the other principles introduced in Section 8.2 (many of these implications are in the exercises of that section). Goodsell and Yli-Vakkuri (unpublished) take this principle, and the Axiom of Infinity, as a basis for an axiomatic foundation for a substantial fragment of the Gallin/Williamson logic.

Among other things, Williamson's book defends one of the more surprising consequences of Classicism—the thesis that everything necessarily exists—on the broadly abductive ground that it offers the simplest account of higher-order modal logic. We briefly touched upon this in Section 8.1; although the principle is usually discussed in terms of metaphysical necessity, given the principle we called Broad Necessitism^e this consequence holds for arbitrary interpretations of 'necessarily exists'. Higher-order ideas can be brought to bear on this claim from several angles. Goodman (2016), for instance, argues for Necessitism from certain 'adjunctive entailments', which he argues (as we do) can be expressed as higher-order identities involving quantifiers (see Table 6.5 from Section 6.4). And Williamson argues for Necessitism about individuals from Necessitism about higher-order entities: roughly higher-order entities that are *about* particular individuals, such as their haecceities, should not exist unless the individuals themselves exist.

'Higher-order contingentism' is a prominent alternative take on higher-order modal logic that rejects Classicism. Higher-order contingentists are not usually being motivated by a finer-grained conception of reality than Classicism permits—typically these philosophers will theorize in a possible worlds framework that accepts Booleanism and its consequences. They rather reject distinctively quantificational theorems of Classicism including the aforementioned theorems stating that individuals, propositions, properties and relations, and so on necessarily exist; indeed they theorize in a 'free' logic, which does not even include the classical axiom of universal instantiation from H (recall the discussion of Free H in Section 5.1). Fine (1977) articulates, in a higher-order framework, a sense in which a proposition, property, relation, etc can be *about* an individual and provides a model theory for higher-order languages in which the existence of a higher-order entity will depend on which individuals it is about. A significant feature of Fine's proposal is that his notion of aboutness makes sense in a setting where entities are individuated by necessary equivalence—one does not need to accept a structured view of propositions in order to accept his notion of aboutness. This idea was later rediscovered by Stalnaker (2012), and many of the details have been ironed out and expanded on in a series of papers by Peter Fritz and Jeremy Goodman: see Fritz and Goodman (2016), Fritz (2018a), Fritz (2018b), Fritz (2016), Fritz and Goodman (2017), and Fritz (2017a).

Many of the concepts explored in this chapter and the previous chapter draw on earlier literature on the metaphysics of modality. The idea of reducing possible worlds to higher-order quantification into sentence position is often attributed to Arthur Prior (see, for instance, Prior and Fine (1979), p. 43), and the formal details are fleshed out by Kit Fine in the postscript. These authors conceive of themselves as reducing possible world talk to necessity and possibility operators, but do not attempt to reduce the modalities to anything more basic, such as logical notions. Prior identifies a world with a weak world, and Fine with a proposition that is possibly both true and entails all truths; a brief comparison of their notion of world with ours can be found in Section 8.2 of this chapter.

The idea of introducing a necessity in terms of propositional identity, as we did with the operator $\Box_\top$ (*being identical to* $\top$), has a long history as well. Roman Suszko and Max Cresswell (Suszko (1971), Cresswell (1967)) study the operator $\Box_\top$ in the context of a propositional logic with a propositional identity connective, $=_t$. Cresswell in particular assumes the analogue of the Rule of Equivalence in his system and proves in that setting that $\Box_\top$ has a modal logic of **S4** (Suszko, by contrast, does not assume that logical equivalence suffices for identity). This definition of necessity in terms of propositional identity and a proof of the **S4** principle was also (re)discovered by Kwasi Wiredu (Wiredu (1979)), where it is offered as a reconstruction of a passage from a passage of C.I. Lewis (Lewis and Langford, 1959, pp. 248–249). Finally, a definition of an operator like $\Box_\top$, this time in the context of a higher-order system, can also be found in Church (1951), although there are some crucial differences that make a direct comparison difficult.[24]

The idea of reducing modal notions to extensional logical notions in Classicism, along the lines explored in Chapter 7, is the subject of Bacon (2018c) and this is where the material from Section 7.1–7.6 is primarily drawn. There the broadest necessity is defined as having every necessity is given and arguments that it is the broadest necessity and has a logic of **S4** are provided; however Bacon (2018c) has a different, much more liberal, notion of necessity, so some of the arguments have been adapted. Proposition 8.1, on the correspondence between finitely axiomatizable modal logics and operators satisfying corresponding definitions, is adapted from Bacon and Zeng (2022).

The strengthenings of Classicism in Section 8.2 are mostly taken from Bacon (2018c), and Bacon and Dorr (forthcoming), and the reader should consult those papers for more in depth discussion. The principle Rigid Comprehension is introduced in Bacon and Dorr (forthcoming), although it has antecedents in Gallin (1975), where a related principle of Rigid Comprehension is connected to other claims such as the weak Leibniz biconditionals.[25] The argument from Functionality to BF and Tractarianism can be found in Bacon (2018c), and the converse implication in Bacon and Dorr (forthcoming). Most of the connections between BF, B and ND discussed in that section are due to Prior; see the citations in this chapter. The discussion of the strong and weak Leibniz biconditionals and the relationship to BF for relational types is new to this book.

The notion of a logically necessary proposition as one with only true substitution instances is due to Bolzano (1929), who was apparently assuming a structured theory of propositions. It was adapted by Tarski (1936b) in order to define the metalinguistic property of *being a logically true sentence*, which he achieved by quantifying over all possible interpretations of the non-logical constants in a higher-order metalanguage. Prior to Tarski logicians were already using higher-order logic to articulate the notion of logical truth in essentially the way we did in Section 8.3.[26] The study of Bolzano's conception of necessity in the context of propositional modal logic continues in McKinsey's interpretation of modal logic (McKinsey (1945)). Technical questions arising in connection to McKinsey's proposal are treated in Humberstone (2016), pp. 165–169, Cresswell (2013), Bacon and Fine (2023).[27] Carnap (1946), Carnap (1947) pp173-174 provides a different theory of logical necessity from McKinsey's; see Bacon and Fine (2023) Section 3.2, and especially Cresswell (2013) for some connections and contrasts between the two. Tarski's higher-order definition of logical truth has had significant influence on the development of mathematical logic, specifically model theory. Schiemer and Reck (2013) outline how language of model theory went from being formulated in a purely logical higher-order language (for instance, in Tarski and Carnap) to being formulated in a first-order theory of sets, namely ZFC.

Drawing on Tarski's higher-order definition of logical truth, Williamson (2013) defines a sentence, $A(\bar{c})$, to be 'metaphysically universal' when it's universal closure $\forall \bar{x} A(\bar{x})$ is true (this way of defining validity is in fact found in Bernays and Schönfinkel (1928), p. 347 prior even to Tarski).[28] The principle Logical Necessity, from Section 8.3, then relates the broad necessity to metaphysical universality in the object language; this principle is introduced in Bacon (2020) and related structural principles concerning the decomposition of propositions, properties, relations, etc. in terms of the fundamental. The principles Logical Necessity and Logical Possibility$^{\mathbf{L}}$ are from Bacon (2020) and Bacon and Dorr (forthcoming) respectively. The question of which extensions of Classicism $\mathbf{L}$ are such that Possibility$^{\mathbf{L}}$ is consistent is still under investigation, however Goodsell (2022) contains some important limitative results concerning logics containing Rigid Comprehension; this result is discussed further in Section 18.6.

We have investigated modal logicism in the context of Classicism, however there are many other theories of granularity in which reductive definitions of broad necessity are possible.[29] Some sufficient conditions under which reductive definitions are possible are given in the final section of Bacon and Zeng (2022). However, there are theories of propositional granularity under which this project may be impossible. One of the conditions an operator must meet is to apply to the propositions with the form of a logical truth. Classicism tells us that there is, in a sense, only one logically true proposition, $\top$, so that it is very easy to formulate this condition. But to investigate modalities in a more general setting, we may need to take some modal notion as primitive. Bacon and Zeng (2022) and Roberts (MS) explore theories that take the notion of being a necessity as primitive, whereas Dorr et al. (2021) instead takes a particular operator, $\Box$, corresponding to broad necessity as primitive. There are some important differences between Bacon and Zeng (2022) and Roberts (MS). The former operate with a liberal notion of necessity, conceived of as an operator possessing the 'worldly' analogue of the modal logic K (cf the present account in Section 7.2). Roberts is following Williamson (2016) in exploring a conception of necessity which may be more restricted, formulates principles ensuring that necessities are closed under certain operations, such as composition and conjunction (Williamson, unlike Roberts, is not working in a higher-order framework).

Finally, we have not touched on another important topic drawing on the distinctive interplay between higher-order and modal notions: the recent literature on higher-order modal mathematics, in connection with the idea that the set-theoretic universe is 'indefinitely extensible'. The reader interested in these topics might start with Linnebo (2013), Linnebo (2018), Fine (2002a), Studd (2013), and Scambler (2021).

Endnotes

1. Dorr et al. (2021), for instance, introduce weaker properties of necessities that require the corresponding modal principles to be merely true, rather than necessary in a suitably strong sense. As a result, these necessities do not end up satisfying the properties corresponding to theorems of the corresponding modal logics.
2. See Burgess (2014).
3. See, for instance, Williamson (2013), Linsky and Zalta (1994), and Goodman (2016).
4. See, for instance, Hazen (1976).
5. When this is done it is easily seen that C may be axiomatized using Modalized Functionality and rule of proof: $\vdash A \leftrightarrow B$ then $\vdash A =_t B$. See Bacon (2018c) for this axiomatization, and Bacon and Dorr (forthcoming) for a proof that one can recover all of Classicism in the relational type system.
6. See Kripke (1980) lecture two.

7. Models for these principles can be found using the sorts of models described in Chapter 18. The details may be found in Bacon and Dorr (forthcoming), Appendix D.
8. It is evidently invalid for some tense operators—such as 'it will always be the case that. . .'—and evidently valid for others—for instance 'it always was, is and will be the case that. . .'.
9. Stalnaker (2010) contains some relevant discussion of possible worlds in a free logical setting where both the propositional Barcan formula and its converse can fail. Unlike Classicism, this system allows for contingently existing propositions.
10. Note that by restricting attention to universal accessibility relations the logic of **S5** is forced, and if the propositional quantifiers were allowed to be restricted even the weak Leibniz biconditionals could fail to be valid.
11. See, for instance, Salerno and Brogaard (2007), Nolan (1997).
12. See, for instance, Chandler (1976) and Salmon (1989). Bacon (2018c) and Dorr et al. (2021) connect this to the thesis that metaphysical necessity is broad necessity.
13. See, for instance, the 'fixedly operator' from Crossley and Humberstone (1981).
14. The view that that metaphysical necessity is not broader than *always* is implicit in the semantics for tense and modal logics in Fine's postscript to Prior and Fine (1979) and Kaplan (1979). Arguments for this view from the supervenience of all truths on the eternal and from the behaviour of actuality operators are endorsed in Bacon (2018b,c), footnote 18 and some critical discussion of them can be found in Dorr and Goodman (2020).
15. See Barnes and Williams (2011) and Bacon (2018d).
16. The instance of BF^e that fails would be given by the property $\lambda z.\ \bigvee_{1\leq i\leq n} x_i =_e z$ where $x_1, \ldots, x_n$ are pairwise distinct individuals.
17. This higher-order definition of logical truth is arguably prior to Tarski—it is, for instance, found in Bernays and Schönfinkel (1928) p347. More recently it has been adopted by Williamson (2013).
18. In fact, given Logical Necessity certain structural notions can be defined in Classicism; see Bacon (2020), Section 4.
19. Friedman (1975a) conjectured that there is a set of sentences of propositional modal logic, X, that contains no propositional letters, contains $A \wedge B$ iff it contains both A and B, contains $\neg A$ iff it does not contain A and contains $\Box A$ iff it contains every substitution instance of A. In other words, an interpretation of propositional modal logic in which $\Box$ expresses a Bolzanoean notion of logical necessity. Moreover Friedman conjectured that there was more than one such set. The existence of such a set was settled in the affirmative independently by Tadeusz Prucnal and Kit Fine. See Prucnal Prucnal (1979), Bacon and Fine (2023). Their models were essentially the same, and the non-uniqueness part of Friedman's conjecture remains open.
20. See Anderson and Belnap (1975), pp. 122–123.
21. Schemas of this form are explored by Fritz et al. (2021).
22. See, for instance, Prior and Fine (1979), Prior (1971), and Fine (1970).
23. See also Tichý (1988).
24. Church is operating in a Fregean system with infinitely many propositional types, $t_0, t_1, \ldots$ where t_0 informally means the type of truth values, and t_{n+1} the type of senses of things of type t_n. Church's operator, $\Box_C$, is of type $t_1 \rightarrow t_0$; and maps a proposition $p : t_1$ to *the true* if the type p is identical to a particular tautology of type t_1. Because input and output of the type of Church's operator are different, it cannot be straightforwardly iterated. And since $\Box_C p$ is simply a truth value, the question of whether statements of the form $\Box_C p$ are necessary are trivial; so questions about the logic of this operator, such as whether it satisfies **S4**, cannot be raised.
25. Gallin's notion of rigidity only makes sense in the context of **S5** and is not a suitable notion of rigidity for the reasons discussed in Section 8.2. In fact, his rigid comprehension principle allows him to derive the **S5** principle. See Dorr et al. (2021), Section 1.5 for more discussion of the present notion of rigidity.
26. See Bernays and Schönfinkel (1928), p. 347. Hilbert and Ackermann (1928), by contrast, adopt a Bolzanoean definition of being a sentence with only true substitution instances except they also require the sentence to be true when we replace the universal type e quantifier with arbitrary restricted quantifiers (although they do not require the higher-order quantifiers to be restricted; see p. 129 in the 1950 translation, Hilbert and Ackermann (1928).
27. One outstanding technical problem relating to McKinsey's analysis makes its way on to Harvey Friedman's list of 101 open problems in logic (Friedman (1975a), Problem 41). A partial solution to the problem was provided independently by Kit Fine and Tadeusz Prucnal. See Prucnal (1979) and Bacon and Fine (2023), Section 2.1.
28. This is essentially how Tarski defines the notion in Tarski (1936b): 'We replace all extra-logical constants which occur in the sentences belonging to L by corresponding variables, like constants being replaced by like variables, and unlike by unlike.' Once this substitution of constants for variables has been performed, one obtains an open sentence of higher-order logic. For Tarski the original sentence is a logical truth if and only if this formula is satisfied by every sequence of entities of the appropriate type: $A(\bar{c})$ is a logical truth if and only if

$\forall \bar{x}\, \mathrm{Sat}(\langle A(\bar{x})\rangle, \bar{x})$. In this paper, Tarski is operating with an informal notion of satisfaction. Tarski notes that a sentence is a logical truth in his sense only if it is in fact true (or more generally, that a logical consequence of truths is true): this is importantly different from the definition of logical truth in terms of set-theoretic models, from which the truth of a sentence may not follow from its truth in all set-theoretic models (see, for instance, Shapiro (1987)).

29. Candidates include the theories of Dorr (2016), Goodman (2018a), and Fine (2017).

Hints for exercises

[i] **Hint:** Use CBF^{σ}.

[ii] **Hint:** Use Functionality to show that $\forall_{\sigma} y \Box_{\top}(Xy)$ implies $X = \lambda y.\top$.

[iii] **Hint:** You should use Modalized Functionality and the Necessity of Identity.

[iv] **Hint:** Use part (a) to infer $\Box_{\top}\forall_{\iota} p(\Box_{M} p \leftrightarrow \Box_{\top} p)$ and appeal to Intensionality.

[v] **Hint:** Prove it by contradiction: suppose $\Diamond\exists xy(Wx \wedge Wy \wedge x \neq_{\sigma} y)$ and get $\Diamond\exists x(W \not\leq \lambda z.(x =_{\sigma} z) \wedge W \not\leq \lambda z.(x \neq_{\sigma} z))$ contradicting the fact that W is a world property.

[vi] **Hint:** Assume $w \leq \Box p$ and start by showing that if v is accessible to w, then $w \leq \mathrm{World}\, v$. Then use $w \leq (v \leq p)$ and the completeness of v to infer that $v \leq p$.

[vii] **Hint:** One strategy is to try and reconstruct McKinsey's counterexample.

Part III

General higher-order languages

9

General λ-languages

In this chapter, and the next, we return to the hypothesis that we can introduce meaningful predicates by abstraction. As we have seen in previous chapters, it is a powerful hypothesis that lets us generate many further entities from ones we already have—things like compositions, converses and reflexizations. In this chapter, we will consider some philosophical positions that view these commitments to be undesirable, and so we here begin to explore various weakenings of the λ-formalism. This chapter, and the next, provide the reader with formal tools for theorizing about more structured theories of propositions and are necessary reading for Chapters 11–13.

A *general λ-language* is a language in a given signature which need not contain all of the λ-terms of the full λ-language over that signature. This chapter is organized as follows. Section 9.1 highlights some of the metaphysical assumptions baked into the full λ-language. Section 9.2 introduces the notion of a general λ-language, and Sections 9.3 and 9.7 investigate several classes of general λ-languages of philosophical interest, they differ concerning whether you allow vacuous variable binding, the binding of variables occurring more than once in the body of a λ-term, and the binding of variables in a different order to which they appear in the body of a λ-term; for each option, there is a version where we have combinators and a version where we do not. Section 9.4 discusses quantification in general λ languages, and Section 9.5 introduces the notion of a general higher-order logic. Section 9.8 treats the variable-free analogues of the languages that have combinators.

9.1 Higher-order ontology and λ-languages

The higher-order logics considered in this book so far, H and its extensions, have substantive higher-order ontological implications—they contain disputed higher-order existential statements. Consider, for instance, the following theorem of H stating, informally, that every relation has a converse:

Converses $\forall_{e\to e\to t}X\exists_{e\to e\to t}Y\forall_e zw(Xwz =_t Yzw)$

Thus for any given relation, say, *loves*, there is another relation, *R*, such that for z to love w is for w to *R* z. In this case, we will call *R* a *converse* of *loves*. For the sake of readability, we will gloss *R* as the relation of *being loved by*, however, the reader should remember that the relationship between the active and passive version of 'loves' in English may be more nuanced, and so should always refer to the official notion of converse. We can derive Converses in H as follows. First, β lets us derive $\forall_e zw(Xzw =_t (\lambda xy.Xyx)wz)$. We can then obtain $\exists_{e\to e\to t}Y\forall_e zw(Xwz =_t Yzw)$ by existentially generalizing on the term $\lambda xy.Xyx$, and finally Converses can be derived using the rule of universal generalization.

DOI: 10.4324/9781003039181-13

Remark 9.1. The reader should note that the presence of a certain λ-term in the language is key to this proof. The standard λ formalism outlined in Chapter 3 for notating abstracted predicates provides us with *two* ways to abstract a binary predicate from the sentential function:

. . . loves ___

There is, however, a real dispute about the existence of converse relations.[1] Natural language sentences sometimes *appear* to involve reference to converses, like 'John loves Mary' and 'Mary is loved by John'. But according to these metaphysicians, these two sentences do not really refer to two distinct relations, *loves* and *is loved by*, but simply one relation with language telling us the order in which to supply it with arguments: the active voice indicates that the relation should receive the individuals named in one order, the passive voice indicates the opposite order.

The idea of the converse of a relation is somewhat puzzling. Languages that evolved from spoken languages have certain features by necessity. The constituents of a sentence are linearly ordered, deriving from the need to say one word before another. Reality itself is not constrained by the needs of human communication, and so it would be a surprising coincidence if the constituents of propositions were ordered in the same way. Yet, it is hard to even spell out the notion of a converse relation without making this assumption about reality. Naïvely, the converse of R is a relation such that, when you plug b into the "first" argument place and a into the "second", yields the same proposition as plugging a into the "first" argument place of R and b into the "second". If a relation is just a proposition with two individual shaped holes, that do not come in a particular order, then the notion of a converse doesn't obviously have a meaning.[2] At the other extreme we might take the notion of order very seriously. The simplest model of this is to identify relations with something linguistic: perhaps a binary relation is a sentence with two distinct gaps. The notion of a converse now make sense, but it is impossible for converses to exist due to the individuation conditions of sentences: if you have two sentences both with two gaps, and you place two names not already appearing in those sentences into those gaps, but in different orders, you will always get two different sentences. This is because sentences are partly individuated by the order of their constituents. In either case, we have reason to doubt the H-theorem, Converses.

The existence of converses thus gives rise to puzzles on a variety of different views about propositional structure. The linguistic model above was an overly concrete model of a structured view. A more minimal assumption about the structure of propositions (acceptable even to structured theories of propositions that reject order) is unique readability: a proposition like *John loves Mary* must have a unique decomposition into a relation and two individuals. But if we believe in converses, the same proposition involves two relations: it can be decomposed as *loves* applied to *John* and *Mary*, or *is loved by* applied to *Mary* and *John*.[3] (The identity of these two decompositions follows from the assumption that *is loved by* is a genuine converse of *loves*: which we took to mean that applying the former to a and b yields the same proposition as applying the latter to b and a.) An even stronger structural thought—found in the linguistic model above—goes beyond the unique decomposability of propositions and asserts that individuals occur in a particular linear order in a proposition. Given that *John loves Mary* and *Mary is loved by John* really are the same proposition (we are assuming *is loved by* is a converse of *loves*) then either *John* comes before *Mary* or

Mary comes before *John*. Whichever it is, the ordering of *John* and *Mary* in the resulting proposition will match the ordering of the argument places of only one of the pair of mutually converse relations, seemingly giving preference to one of *loves* or *is loved by*. But even non-structured theories face puzzles to do with converses. One has to do with fundamental non-symmetric relations. Assume that either *more massive than* or its converse *less massive than* is fundamental, and assume, as it often is, that they cannot both be fundamental on pain of redundancy among the fundamental. Why should it be one and not the other? The awkward choice is circumvented if there is only one relation.[4] We will return to these issues in more detail in the chapters on structure (Chapters 11–13).

Consider, next, two more theorems of H:

Reflexizations $\forall_{e\to e\to t}X\exists_{e\to t}Z\forall_e y(Xyy =_t Zy)$

Vacuity $\forall_t p\exists_{e\to t}Z\forall_e y(p =_t Zy)$

Comprehension Check 9.1. *By analogy with the proof of Converses, convince yourself that these are also theorems of* H. *What features of the standard λ way of representing abstracted predicates are responsible for these theorems?*

Given a relation *R*, say *loves*, one may construct its reflexization $\lambda x.Rxx$, *loves oneself*. One could take the attitude that there isn't a further unary property, *loves oneself* out there in reality, there is just the binary *loves*, but natural languages allow us to construct unary expressions that refer to a binary relation and which feed its single argument to it twice. Or one might insist that there is not a unary property *is such that snow is white* out there in reality, as formalized with vacuous λ-abstraction $\lambda x.P$, there is just the proposition that *snow is white*, with language allowing us to form constructions that throw away arguments. Structured views also have reason to be unhappy with these further properties because they seem to violate the unique decomposition principle. Since for *John to love John* just is for *John to love himself*, $Laa =_t (\lambda x.Lxx)a$, a single proposition seems to involve both the binary *loving* relation and the unary *loving oneself* property. Similarly for *snow to be white* is for *John to be such that snow is white*, a single proposition, that *snow is white*, can be decomposed so as to have John as a constituent, or not. And there is also a stronger structural idea that goes beyond unique decomposition according to which we can say *how many times* a constituent occurs in a proposition. Since *John loves John* and *John loves himself* are the same, does John appear once or twice? Does John appear at all in the proposition that *snow is white*, since it the same as the proposition that *John is such that snow is white*? And doesn't this trivialize the notion of constituenthood altogether? On the other hand, to distinguish all these propositions would not only involve rejecting βη-synonymy but would seem to posit distinct facts where it seems like there is only one fact.

Another class of theorems of H with higher-order ontological implications are theorems asserting the existence of various combinators. For instance, the existence of an identity operation corresponding to the combinator I^t is a theorem

Identity $\exists_{t\to t}X\forall_t p(Xp =_t p)$

This states that there is an operator that when applied to any proposition yields that proposition. Similarly, one can prove that there is an operation that maps each relation to its converse ($\exists_{(e\to e\to t)\to e\to e\to t}X\forall_{e\to e\to t}Y\forall_e zw(XYzw =_t Ywz)$), that maps each relation to its reflexization, and so on. Anyone unhappy with converses, reflexizations, and so on, will of course also be unhappy with these implications. However, the presence of combinators has an importantly different ontological implication—one might think that every relation has a converse without thinking that there is an operation that maps each relation to its converse.[5]

For example, unlike the converse and reflexization combinators, the outputs of the identity operation should always be acceptable (you get out what you put in), yet the identity operation itself is highly contentious. According to some structured theories of propositions, all entities are composed of constituents, which are combined through application. If you apply an operator to a proposition you get a different proposition with the operator as a constituent, so there is no such thing as an identity operation.

Related to the existence of combinators are theorems like the following:

Lifting $\forall_e x \exists_{(e\to t)\to t} Z \forall_{e\to t} Y.(ZY =_t Yx)$

For a given individual, say Socrates, there is the higher-order property, $\lambda X.Xa$, of *applying to Socrates*, which has the feature that for any property F, for F to be $\lambda X.Xa$ is for Socrates to be F. Here we are making use of λ-terms where the abstracted variable appears applied to some other expression. This theorem would be rejected in certain theories of structured propositions. According to a common picture, each proposition can be decomposed into an *outer* operation or relation applied to some arguments—its *immediate constituents*—which themselves can be decomposed into an outer operation with immediate constituents, and so on until we reach a simple entity without any immediate constituents. Thus, for instance, *John loves Mary or John hates Mary* has disjunction as its outer relation, and *John loves Mary* and *John hates Mary* as its immediate constituents, and these respectively have *loves* and *hates* as their outer relation, and *John* and *Mary* as their immediate constituents, which are simple. This sort of structured theorist should therefore also reject λ-terms that bind variables appearing in predicating position, like $\lambda X.Xab$ where $X : e \to e \to t$, in which a bound variable appears in a position where it is being applied to another term. β ensures that applying $\lambda X.Xab$ to R yields the proposition Rab that, we are supposing, has R as its outermost relation. Thus terms like $\lambda X.Xab$ represent entities that do not have an outermost relation, but rather an argument place where the outer relation should be. This contradicts the idea that the immediate constituents of a proposition are those entities filling an argument place of the outermost relation of that proposition: $\lambda X.Xab$ has one argument place, but when it is filled by R the result has R as the outer relation instead of as an immediate constituent.

Finally, because the signature of higher-order logic contains constants corresponding to the logical operations, like $\wedge$ and $\forall_e$, it involves a higher-order commitment to these operations that could be expressed by quantifying into connective and quantifier positions respectively. We noted in Section 4.2 that this is a higher-order ontological implication that was explicitly rejected by the logical atomists, who treated these expressions as syncategorematic. We also noted that given full power of the λ-notation we can recover these logical operations even when they are officially taken as syncategorematic, so this is yet another way in which the standard higher-order formalism can fail to be neutral.

9.2 General λ-languages

It is clear, then, that the weakest system H builds in many substantive higher-order ontological assumptions. To explore views that reject these assumptions we need a more flexible formalism. If we inspect the proofs of the offending existential theorems we can see that they proceed by first finding a λ-term which, given β, corresponds to an entity with the contested property, and then by an application of existential generalization they conclude that the contested entity exists. For instance, in the derivation of Converses, we showed that

$\lambda xy.Ryx$ had the contested "converse like" behaviour by using β, and then we applied existential generalization to conclude that some relation had the converse-like behaviour. There are consequently a few ways we might modify the higher-order formalism of previous chapters: one could remove the principle β, one could remove existential generalization (and its dual universal instantiation), or one could simply remove the offending λ-terms from the language.

Each approach has its own costs. If one relaxes the principle β, the argument for Converses and the other principles discussed in the previous section are blocked because we can no longer prove that $\lambda xy.Ryx$ behaves like the converse of X. The problem is that we then have relational terms like $\lambda xy.Ryx$ hanging around whose behaviour are completely unconstrained. There will be essentially no difference between a λ-term and a non-logical constant of the same type, so there is little purpose having them in the language. By contrast, with β and η the intepretations of λ-terms are very constrained: these principles together pin down the meanings of λ-terms up to synonymy by Proposition 3.

Remark 9.2. Some authors have instead replaced β with a weaker principle ensuring that $\lambda xy.Ryx$ is an *extensional* converse of R: $\forall_e zw.(Rzw \leftrightarrow (\lambda xy.Ryx)wz)$ (see the principle Extensional β in Section 11.1).[6] These weakenings of β still leave the interpretations of λ-terms wildly unconstrained: $\lambda xy.Ryx$ could denote anything coextensive with the converse of R—depending on the empirical facts, *shorter than* could be the interpretation of '$\lambda xy.y$ *is older than* x'. Moreover, the reasons we gave for being skeptical of Converses extend to its extensional counterpart (i.e. the result of replacing $=_t$ in Converses with $\leftrightarrow$), for the notion of an extensional converse seems also to rely on the possibility of coordinating the argument places of two distinct relations. In general, what reason do we have to believe in extensional converses, extensional reflexizations, and so on, if not because of the existence of converses, reflexizations and so on? A suitably general higher-order framework should not, it seems, build in the existence of extensional converses either.

If one relaxes, instead, the principle of existential generalization and universal instantiation one can still prove that $\lambda xy.Ryx$ behaves like a converse of R—$\forall_e zw((\lambda xy.Ryx)zw =_t Rwz)$—but one can no longer prove that a relation with this behaviour exists. Here, one might also wonder if the claim that $\lambda xy.Ryx$ is a converse of R, and the other consequences of β needed in the proofs of the other contested theorems of H, are bad enough on their own given the reasons they are being contested. But even if not, on this way of weakening H, quantificational claims no longer express generalizations in the sense outlined in the introductory section, Section 0.3. A universal generalization no longer implies all its instances, and an existential is no longer implied by each of its instances. Since we don't have analogues of the higher-order quantifiers in English, we relied quite heavily on their logical role when we introduced them in Section 0.3, where we appealed to the fact (Theorem 0.2) that the standard quantifiers rules pin down the quantifiers up to logical equivalence. In a logic without universal instantiation and existential generalization, $\forall_\sigma$ and $\exists_\sigma$ behave like a restricted quantifier $\forall^F_\sigma x$ (i.e. $\lambda Y \forall_\sigma x(Fx \rightarrow Yx)$) in H, restricted by some non-logical predicate $F : \sigma \rightarrow t$, and so are also quite unconstrained.

Apart from the role existential generalization has in answering concerns relating to the intelligibility of higher-order quantification, there are also practical considerations for having true generalizing devices in the language. The ability to express generality with respect to all meaningful instances of a claim is useful for general theorizing of the sort common in philosophy. If one takes a device, like $\forall$, as primitive for forming true generalizations, one

can always introduce the other kind of quantifier as a restriction of it. (Some philosophers, for instance, have argued that there are two important kinds of first-order quantifiers: the metaphysically 'lightweight' quantifiers we have been labeling $\forall$ and $\exists$ for expressing generalizations, and which are pinned down by their logical role, and restrictions of these which express 'ontological commitment' in some more metaphysically serious sense.[7])

The option we take in this book, then, is the final one: we simply consider languages that remove the offending λ-terms from the language. Someone who rejects the existence of, say, converses should be puzzled about what a λ-term like $\lambda xy.Ryx$ means for reasons canvassed in Remark 9.2. It seems a little convoluted to attempt to revise existing logical principles just so that we can have meaningless terms in the language. Thus, when a λ-term belongs to a language we will take it to be governed by β and η. On the positive side, we will find that there's a fairly systematic correspondence between the different metaphysical pictures discussed in the previous section and the sorts of λ-terms they admit. Below I have listed some positions concerning entity combination, ordered roughly by how many ways of constructing complex entities from simpler ones they accept. The reader may wish to think about what kinds of λ-terms each sort of view accepts.

Stage 1: You can build things by application. Whenever you have an entity F of type $\sigma \to \tau$ and another entity a of type σ you can construct another entity by *applying* F to a, to get Fa of type τ.

Example: *John loves Mary*, $Lab : t$, can be made from *loves*, *John*, and *Mary* ($L : e \to e \to t$, $a : e, b : e$). *John loves*, $La : e \to t$, can be made from *loves* and *John*.

Stage 2: Application is not the only way of combining two entities to make a complex entity. There are two ways of combining the binary relation *loves* with *John*. One corresponds to application, but the other corresponds to plugging *John* into the 'second' argument place of *loves*. Similarly, given two operators, such as *not* and *necessarily*, we can form a complex operator by composition, *not necessarily*. We will later introduce a general mode of combination 'complication' which has composition and application as limiting cases.

Example: Although we can make *John loves*, i.e. *La*, by application, we cannot make *loves John* without a λ-term, $\lambda x.Lxa$, but these seem to be in equally good standing. The composition of $\neg$ and $\Box$ cannot be expressed as the result of applying one to the other, instead we need a new mode of combination, composition. The composition can be expressed with a λ-term $\lambda p.(\neg(\Box p))$.

Stage 3: In addition to the previous modes of combination we might admit modes of combination that create new relations by rearranging the argument places of existing relations (i.e. reorder, merge or expand the argument places). One might accept (or reject) any subset of the following ways of creating entities from old: that one can build a new binary relation *is loved by* from the relation *loves*, that one can build the unary property *loves oneself* from *loves*, one can build the unary property *is such that snow is white* from the proposition *snow is white*.

Example: each of these new entities can be expressed in terms of expressions for the old entity along with λs and bound variables: $\lambda xy.Lyx$, $\lambda x.Lxx$, $\lambda x.P$.

Stage 4: In addition to these structural modes of combination, you might also want to admit logical modes of combination.

Example: Given propositions P and Q, you might posit a conjunctive way of combining them to make a new proposition $(P \wedge Q)$.

Stage 5: The previous stages posit ways to combine old entities to make new ones. For each mode of combination that you accept, you might also posit an operation of a higher type that maps the input entities to the combined entity.

Example: Anyone who accepts application as a way of creating a complex entity $Fa : \tau$ from entities $F : \sigma \to \tau$ and $a : \sigma$ might also want to postulate a corresponding combinator, $\lambda X\lambda y.Xy : (\sigma \to \tau) \to \sigma \to \tau$ that combines with F and a (by application) to make Fa. Composition takes two operations $F : \sigma \to \tau$ and $G : \tau \to \rho$ to make their composition $G \circ F$ of type $\sigma \to \rho$. Anyone who accepts this mode of combination might also wish to posit a composition combinator $\lambda XYz.(X(Yz)) : (\tau \to \rho) \to (\sigma \to \tau) \to (\sigma \to \rho)$. Similar points can be made about the logical modes of combination: once you've accepted conjunctive propositions, you might posit a conjunction *connective*, $\wedge : t \to t \to t$.

An appropriate language to formulate the first sort of view would be an applicative language (see Definition 1.4), which only allows one to form new terms by application. In applicative languages (and every other language we will encounter) application is treated in a syncategorematic way. By the same principle, someone willing to ascend to stage 2 does not *need* to introduce λ-terms: they could simply introduce a new syncategorematic rule for each mode of combination they accept. One could have, in addition to the clause 1 in an applicative language, a clause like 2 for composition, a clause like 3 for converses, and so on:

1. If M and N are terms of type $\sigma \to \tau$ and σ respectively, (MN) is a term of type τ.
2. If M and N are terms of type $\tau \to \rho$ and $\sigma \to \tau$ respectively, $[MN]$ is a term of type $\sigma \to \rho$.
3. If M is a term of type $\sigma \to \tau \to \rho$ then M^c is a term of type $\tau \to \sigma \to \rho$.

⋮

Indeed, we shall introduce a notation like this later. However, there are systematic identities which are automatically secured by a λ-treatment (given β and η) which must be put in by hand if we adopt the syncategorematic approach. Take someone who believes we can form a new operator by composing two existing operators, and consider the standard definition of possibility from necessity. We have two ways of expressing that using binary composition: $[[\neg\Box]\neg] : t \to t$ or $[\neg[\Box\neg]] : t \to t$. Indeed, if we can make sense of binary composition, why not ternary composition, which takes three composable operators F, G and H, and produces another: $[FGH]$. Now we have a proliferation of duals of $\Box$: $[\neg\Box\neg]$ in addition to $[[\neg\Box]\neg]$ and $[\neg[\Box\neg]]$. Shouldn't these all be identified? The natural definition of these entities in terms of λ, are $\lambda p.(\neg(\Box(\neg p))$, and by identifying $[MN]$ with $\lambda p.M(Np)$, we get

$\lambda p.(\lambda q.\neg(\Box q))(\neg p)$ for $[[\neg\Box]\neg]$ and $\lambda p.\neg((\lambda q.\Box(\neg q))p)$ for $[\neg[\Box\neg]]$. Our principle β ensures these are also synonymous. The λ-formulation along with β ensures:

> There is no difference between applying the composition of $\neg$ and $\Box$ to A and applying $\neg$ to the result of applying $\Box$ to A: $([\neg\Box]A)$ and $(\neg(\Box A))$ are synonymous.
>
> There is no difference between (i) the result of plugging a into the first argument of R, Ra, and then b into the remaining argument place, Rab, and (ii) the result of plugging b into the second argument of R, $\lambda x.Rxb$, and then a into the remain argument place, $(\lambda x.Rxb)a$.
>
> and so on

The λ-formalism provides us with a simple and elegant theory which systematically predicts all of these identities.

In order to theorize in a suitable general way, we introduce the notion of a general λ-language. A general λ-language in a signature Σ is a subset of the terms of the full λ-language $\mathcal{L}(\Sigma)$ that are closed under certain operations that ensure they behave sufficiently like a language: for instance, they must be closed under substitution, closed under $\beta\eta$-reduction (recall Section 3.4), and so on. We give a precise set of conditions below.

Definition 9.1 (General λ-languages)**.** *A general λ-language $\mathcal{J}(\Sigma)$, in a signature Σ, is a type indexed collection of sets $\mathcal{J}^\sigma(\Sigma)$ such that:*

- $\Sigma^\sigma \subseteq \mathcal{J}^\sigma(\Sigma) \subseteq \mathcal{L}^\sigma(\Sigma)$ *for every type σ.*
- *If $M \in \mathcal{J}^{\sigma\to\tau}(\Sigma)$ and $N \in \mathcal{J}^\sigma(\Sigma)$, $(MN) \in \mathcal{J}^\tau(\Sigma)$.*
- *If $(MN) \in \mathcal{J}^\tau(\Sigma)$ and $M : \sigma \to \tau$ then $M \in \mathcal{J}^{\sigma\to\tau}(\Sigma)$*
- *If $(MN) \in \mathcal{J}^\tau(\Sigma)$, $N : \sigma$ and $N \notin Var^\sigma$, then $N \in \mathcal{J}^\sigma(\Sigma)$*
- *If $M \in \mathcal{J}^\sigma(\Sigma)$ and M $\beta\eta$-reduces to N, then $N \in \mathcal{J}^\sigma(\Sigma)$.*
- *If $M \in \mathcal{J}^\sigma(\Sigma)$, $a \in Var^\tau \cup \Sigma^\tau$, $N \in \mathcal{J}^\tau(\Sigma)$ and N is free for a in M, then $M[N/a] \in \mathcal{J}^\sigma(\Sigma)$.*
- *If $M \in \mathcal{J}^\sigma(\Sigma)$, $x_1, \ldots, x_n \in FV(M)$ are distinct variables and $y_1, \ldots, y_n$ are distinct variables that are free for $x_1, \ldots, x_n$ in M, then $M[y_1/x_1 \ldots y_n/x_n] \in \mathcal{J}^\sigma(\Sigma)$.*

Observe that if an application term (MN) belongs to a general λ-language then M and N belong to it, unless N is a variable, in which case only M need belong to it. The reader does not need to worry too much about the exception where N is a variable until Section 9.7 where we consider languages that do not contain any variables as terms (although variables may appear, free or bound, within complex expressions of the language). The languages we consider in Section 9.3 contain an application term (MN) if and only if they contain both the terms M and N, collapsing the second, third and fourth conditions into one.

Comprehension Check 9.2. *Suppose that $\mathcal{J}(\Sigma)$ is a general λ-language containing all the variables. Show that if a term $\lambda x.M$ is in the language then so is M.*

Our official definition of $\beta\eta$-equivalence required two terms, M and N, to be linked by a chain of immediately $\beta\eta$-equivalent terms from the full language $\mathcal{L}(\Sigma)$. The synonymy of $\beta\eta$-equivalents then is justified by the synonymy of immediate $\beta\eta$-equivalents, and the transitivity of 'is synonymous with'. For someone who accepted as meaningful only terms belonging to a general λ-language, this justification is useless, since some of the terms linking the meaningful terms M and N might be meaningless (and so not synonymous). Thus

one might hope that, when terms M and N belong to a general λ-language, they are $\beta\eta$-equivalent iff they can be linked by immediately $\beta\eta$-equivalent terms *in that language*. This is shown in the following proposition. Recall that the Church-Rosser theorem (Theorem 3.1) from Section 3.4 tells us that if P and Q are $\beta\eta$-equivalent then there is a term, R, which they both $\beta\eta$-reduce to.

Proposition 9.1. *Suppose that P and Q are terms of a general λ-language, $\mathcal{J}(\Sigma)$and that P and Q are $\beta\eta$-equivalent. Then they are linked by a finite chain of immediately β or η equivalent terms* that all belong to $\mathcal{J}(\Sigma)$.

Proof. By by the Church-Rosser theorem (Theorem 3.1) P and Q $\beta\eta$-reduce to a common term R. Thus R can be gotten from P via a chain of immediate $\beta\eta$-*reductions* $P, M_1, \ldots, M_i, R$, and can be gotten from Q via a chain of immediate $\beta\eta$-reductions $Q, N_1, \ldots, N_j, R$. Since a general λ-language is closed under $\beta\eta$-reduction $M_1, \ldots, M_i, N_1, \ldots, N_j$ are all part of $\mathcal{J}(\Sigma)$ so that $M_1, \ldots, M_i, R, N_j, \ldots, N_1$ is a chain of immediately $\beta\eta$-equivalent terms of $\mathcal{J}(\Sigma)$ linking P and Q. □

This fact allows us to define $\beta\eta$-equivalence in the normal way. What if there is a chain of immediate $\beta\eta$-equivalents linking P and Q that involves illegitimate terms (by the lights of $\mathcal{J}(\Sigma)$), and no such chain consisting of only legitimate terms? The above proposition ensures that this cannot happen.

In the next (slightly tricky) exercise you will show that the language you get by adding n-ary composition (for each n) to an applicative language is a general λ-language:

Example 9.1. *Consider the smallest subset of $\mathcal{L}(\Sigma)$ that:*

1. *Contains Σ and every variable.*
2. *Contains $(MN) : \tau$ whenever it contains $M : \sigma \to \tau$ and $N : \sigma$.*
3. *Contains contains $\lambda x.(M_n(\ldots(M_1 x)\ldots)) : \sigma_1 \to \sigma_{n+1}$ whenever it contains terms $M_1 : \sigma_1 \to \sigma_2, \ldots, M_n : \sigma_n \to \sigma_{n+1}$ provided $x \notin FV(M_1) \cup \ldots \cup FV(M_n)$.*

This is a general λ-language.

Exercise 9.1. *Show that the language defined in Example 9.1 is a general λ-language.*

9.3 Relevant, affine, linear and ordered languages

We shall begin by focusing on four special general λ-languages. Each of these languages will contain combinators corresponding to each mode of combination they permit. In Section 9.7 we will look at restricted languages that do not contain combinators.

Let's begin with some informal definitions. A *relevant* term, M, is (roughly) a term in which every bound variable occurs free at least once in the scope of the λ that binds it. An *affine* term is one in which every bound variable occurs free at most once in the scope of the λ that binds it. A *linear* term is one in which every bound variable occurs free exactly once in the scope of the variable that binds it. And a *ordered* term is one in which every bound variable occurs free exactly once and is the rightmost free variable in the scope of the λ that binds it.

Comprehension Check 9.3. *Which of the standard combinators B, C, I, W, K and S are (1) relevant, (2) affine, (3) linear, (4) ordered?*

We'll now turn to making these definitions precise. Let's start with the simplest: the notion of a relevant term.

Definition 9.2 (Relevant term)**.** *The relevant terms over a signature Σ are defined inductively:*

- *Any constant or variable is relevant.*
- *If $M : \sigma \to \tau$ and $N : \sigma$ are relevant, then so is (MN).*
- *If M and $x \in FV(M)$, then $\lambda x.M$ is relevant.*

The definition of a relevant term follows the definition of a term (Definition 3.1) exactly, except for the final clause, which states that we can only form a λ-abstract when the newly bound variable appears free in the body of the abstract. It's worth noting that while relevance is preserved under α and η equivalence, it is not preserved under β-equivalence.

Exercise 9.2. *Construct a relevant term that is β-equivalent to a term that isn't relevant.*

Recall, however, β-reduction is *asymmetric*—$(\lambda x.M)N$ β-reduces to $M[N/x]$, but not conversely—and this relation *does* preserve relevance.

Proposition 9.2. *If $(\lambda x.M)N$ is relevant then $M[N/x]$ is relevant.*

This may be proved by induction given our recursive definition of substitution (Definition 3.4).

Definition 9.3. *The relevant language $\mathcal{J}_R(\Sigma)$ is defined by setting $\mathcal{J}_R^\sigma(\Sigma)$ to be the set of relevant terms in $\mathcal{L}^\sigma(\Sigma)$ for each σ.*

Exercise 9.3. *Appealing to the above remarks as necessary, prove that $\mathcal{J}_R(\Sigma)$ is a general λ-language. (It suffices to show that it contains the constants in Σ, contains (MN) iff it contains M and N, and is closed under β-reduction and the required substitutions.)*

In order to study the other three classes of terms we will regularly make use of three sorts of mathematical objects: finite sets, multisets and sequences. Sets and sequences should already be familiar. A multiset is halfway between a set and a sequence: they are collections of elements in which certain elements can appear more than once. Like a set (but not a sequence) the order of the elements does not matter, and like a sequence (but not a set) the number of times an element appears matters. Each of these mathematical objects comes with a binary operation, $\circ$, that takes two finite sets, multisets or sequences and produces another. For sets $x \circ y$ just denotes the union of the sets, and for sequences their concatenation. For multisets, it is their union, calculated in the natural way: by adding together the number of times each element occurs in the two multisets being unioned. Multisets are notated using square parentheses. Because duplicities matter, $[a] \circ [a, b] = [a, a, b]$ but is distinct from $[a, b]$. As usual, $\langle a, b, c\rangle$ represents the sequence whose first element is a, second element b and third element c. (We understand n-tuples as primitive, so that $\langle a, b, c\rangle$ is not to be reduced to $\langle\langle a, b\rangle, c\rangle$ or $\langle a, \langle b, c\rangle\rangle$.)

Comprehension Check 9.4. *Say which of the following are equal (1) as sets, (2) as multisets, (3) as sequences. (For (2) and (3), you must replace $\{$ with $[$ and $\langle$ and $\}$ with $]$ and $\rangle$ respectively.)*

a. $\{1, 2\} \circ \{2, 3\}$

b. $\{1, 2, 3\}$

c. $\{1, 2, 2, 3\}$
d. $\{1, 2, 3, 2\}$
e. $\{1, 2\} \circ \{3\}$
f. $\{1\} \circ \{2, 3\}$

We'll appeal to these mathematical entities throughout the chapter. Right now these concepts allow us to define a more informative version of the free variable operation, FV, on terms. In Chapter 3 it was defined as a function mapping a term, M, to a *set*, namely the set of variables occurring free in M. But we might also want an operation that returns more information, such as how many times the variable occurs free, or the order in which they occur free.

Definition 9.4 (Free variables revisited)**.** *We define two operations FV_m and FV_s from terms to multisets and sequences respectively. They are defined analogously in either case:*

- $FV_m(c) = \emptyset$ *and* $FV_s(c) = \langle\rangle$ *when c is a constant*
- $FV_m(x) = [x]$ *and* $FV_s(x) = \langle x \rangle$
- $FV_*(MN) = FV_*(M) \circ FV_*(N)$ *where* $* = m, s$
- $FV_*(\lambda x.M) = FV_*(M) \setminus x$ *where* $* = m, s$

$FV_*(M) \setminus x$ denotes the result of removing ***all*** *occurrences of x from the multiset or sequence $FV_*(M)$, but otherwise preserved the number/order of the remaining elements. (So, for instance, $\langle x, y, z, x, y\rangle \setminus x = \langle y, z, y\rangle$.)*

If we reinterpret these definitions with sets, reading $\circ$ as set union, and $X \setminus a$ as the result of removing a from the set X, we get the original definition of FV.

Exercise 9.4. *Suppose $X : e \to t \to t$, $y : e$ and $z : t$. Showing your working, calculate:*
a. $FV_s(Xy)$
b. $FV_s(\lambda z.\lambda y.Xyz)$

The operation FV_m gives us a way of making the claim that a variable x occurs in a term M at most once, or exactly once precise: multisets come equipped with a well-defined operation of member count that takes a finite multiset and an element, and returns a natural number (possibly 0) telling you how many times that element occurs in the set. For instance, the member count of a in $[a, b, a, c]$ is 2 (assuming a, b and c are pairwise distinct), whereas the member count of c in this multiset is 1. A variable x occurs free exactly once in M iff the member count of x in $FV_m(M)$ is 1, and at most once if it is 1 or 0. We are now in a position to define the linear and affine terms in terms of λs binding free variable that occur exactly once or at most once.

Definition 9.5 (Linear and affine terms)**.** *The linear (affine) terms over a signature Σ are defined inductively:*

- *Any constant or variable is linear (affine).*
- *If $M : \sigma \to \tau$ and $N : \sigma$ are linear (affine), then so is $(MN) : \tau$.*
- *If $M : \tau$ is linear (affine) and x appears in $FV_m(M)$ exactly once (at most once), then $\lambda x.M : \sigma \to \tau$ is linear (affine).*

$FV_s(M)$ tells us both how many times each free variable occurs in M, and the order in which they occur. Because it also tells us how many times each variable occurs in M it could also have been used in the above definition, although it contains redundant information.[8] The *rightmost* variable appearing free in M is the last element of the sequence $FV_s(M)$, so we may precisify our gloss on an ordered term as follows:

Definition 9.6 (Ordered terms)**.** *The ordered terms over a signature* Σ *are defined inductively:*

- *Any constant or variable is ordered.*
- *If* $M : \sigma \to \tau$ *and* $N : \sigma$ *are ordered, then so is* $(MN) : \tau$.
- *If* $M : \tau$ *is ordered and* $x : \sigma$ *is the last element of* $FV_s(M)$ *and doesn't appear elsewhere in the sequence, then* $\lambda x.M : \sigma \to \tau$ *is ordered.*

Notice that 'ordered' doesn't mean that bound variables occur exactly once, and the bound variables appear in the order they are abstracted, as the following exercise illustrates

Exercise 9.5. *Show that the term* $\lambda Xy.Xzy$ *is linear by constructing it according to definition 9.5. What happens when you try to construct it using the rules for ordered terms?*

Definition 9.7. *The affine, linear, and ordered* λ*-languages, in signature* Σ, $\mathcal{J}_A(\Sigma)$, $\mathcal{J}_L(\Sigma)$, $\mathcal{J}_O(\Sigma)$ *are defined as the affine, linear and ordered terms of* $\mathcal{L}(\Sigma)$ *respectively.*

As with relevance, β-equivalents of linear or ordered terms need not themselves be ordered, however linearity and order are both preserved under β reduction.

Exercise 9.6. *Show that* $\mathcal{J}_A(\Sigma)$, $\mathcal{J}_L(\Sigma)$, $\mathcal{J}_O(\Sigma)$ *are general* λ*-languages.*

9.4 Quantifiers in general λ-languages

In earlier parts of the book, λ played a dual role. Above we have focused on its role in defining complex properties and relations out of simpler ones, but it was also essential for the way we expressed complex quantificational claims. In the full λ-language we were able to avoid treating quantifiers as primitive variable binders by using λ to do the binding jobs that primitive variable binding quantifiers (Definition 4.5) usually do. Then, in Section 4.2, we showed that even if we did treat quantifiers as primitive variable binders, we could recover quantificational expressions of type $(\sigma \to t) \to t$ from the primitive variable binders using λ, showing that the two approaches are in effect equivalent.

However, primitive variable binding quantifiers can bind variables occurring multiple times and in various orders, thus, unsurprisingly, in general λ-languages the two approaches to quantification come apart. There are thus two approaches we might take to quantification in general λ-languages. We could reintroduce primitive variable binding quantifiers into the language. This is pursued in Chapter 10 (Section 10.3). Alternatively, we could still do the jobs of quantification with constants, like $\forall_\sigma, \exists_\sigma : (\sigma \to t) \to t$, but introduce new logical quantificational constants that let us do jobs that $\forall_\sigma$ and $\exists_\sigma$ cannot do when the relevant kinds of λ-terms are missing.

The two approaches are quite different philosophically. When abstraction is limited, one cannot always construct higher-order entities of quantifier type corresponding to the variable binding quantifiers, making the primitive variable binding approach compatible with a more modest higher-order ontology. Since we treat quantifiers as primitive variable binders in Section 10.3, we will here focus on the second strategy.

If we simply take constants $\forall_\sigma, \exists_\sigma : (\sigma \to t) \to t$ as primitive we will quickly come up against expressive limitations. Consider the following sentences from Exercise 4.4.

1. Someone loves himself.
2. Everybody is loved by somebody.
3. Everyone loves someone who loves them back.
4. Something is such that snow is white.

According to the approach we described in Section 4.2 for the full λ-language, the first claim is analysed in terms of the property of *loving oneself* being instantiated: $\exists_e \lambda x.Lxx$. This analysis therefore requires us to take on a commitment to a further unary property, the property of *loving oneself*, in addition to the binary relation of *loving*. We have no guarantee that such terms exist in an arbitrary general λ-language, and above we have considered languages that ban each of these three types of variable binding.

Of course, there is no need to posit an additional property, $\lambda x.Lxx$, in order to say that someone loves himself. In addition to the higher-order property that unary properties have when they are instantiated by something, there is another higher-order property that relations have when they are instantiated by the same thing in both arguments; that *R* has when something *R*s itself. If the former is a legitimate higher-order property, then so is the latter. Moreover, we have no special reason to think that the latter higher-order property should be reducible to the former—indeed, we have already discussed a reason to be wary of such reductions of the truth-functional connectives and quantifiers to other truth-functional connectives and quantifiers in Section 4.1. It turns out there *is* a reduction when you accept the metaphysics implicit in the full λ-language *and* you accept a coarse-grained metaphysics that allows one to use logical equivalence as sufficient for identity: the quantifier in question can be defined as $\lambda X.\exists_e(\lambda y.Xyy)$. This is certainly a nice feature, but not one that we would have expected to hold antecedently. We shall notate these two higher-order quantifiers as follows:

- $\exists_{\hat{\sigma}} : (\sigma \to t) \to t$
- $\exists_{\hat{\sigma}\hat{\sigma}} : (\sigma \to \sigma \to t) \to t$

The significance of the hats will become apparent shortly. There is an obvious parallel convention for the universal quantifier. General λ-languages that don't allow abstraction on variables occurring multiple times should therefore take the second sort of quantifier as primitive as well as the first.

Comprehension Check 9.5. *Formalize someone loves himself using* $\exists_{\hat{e}\hat{e}}$ *but without using any λs.*

There are also two higher-order relations which take a binary relation, *X*, in its first argument place, and an individual *y* in the second argument place to yield the proposition that something *X*s *y*, and something is *X*ed by *y* respectively. In the full language, they may be defined as $\lambda X.\lambda y.\exists_e z.Xzy$ and $\lambda X.\lambda y.\exists_e z.Xyz$ respectively, but we could also introduce them as new logical constants.

- $\exists_{\sigma\hat{\tau}} : (\sigma \to \tau \to t) \to \sigma \to t$.
- $\exists_{\hat{\sigma}\tau} : (\sigma \to \tau \to t) \to \tau \to t$.

Exercise 9.7. *Formalize everyone is loved by somebody using* $\forall_{\hat{e}}$ *and* $\exists_{\hat{e}\hat{e}}$.

There is clearly a general pattern here. Suppose that R is an n-ary relation between entities of types $\sigma_1, \ldots, \sigma_n$, and that we have chosen to quantify into one or more of these argument places at the same time; let's say exactly k argument places have been chosen (each of these argument places must, of course, be of the same type). Then there is a higher-order relation that takes a relation with Rs type, and then $n - k$ entities of the types that remain from $\sigma_1, \ldots, \sigma_n$. This is the coordinated existential quantifier which, when given an n-ary relation, existentially quantifies into the k positions indicated, and outputs an $n - k$-ary relation.

A *hatting* of a sequence of types, $\sigma_1, \ldots, \sigma_n$, is the result of putting hats on top of some of the types in this sequence, provided the hats only appear above at most one sort of type: for instance $e\hat{e}t\hat{e}$, or $\hat{t}(e \to t)\hat{t}e$. We write $\boldsymbol{\sigma}$ to denote a hatted sequence of types, $\sigma_1, \ldots, \sigma_n$. We write $\breve{\boldsymbol{\sigma}}$ for the result of *unhatting* the hatted sequence $\boldsymbol{\sigma}$: that is deleting all the types from the sequence that are hatted. Unhatting the two previous examples yields et and $(e \to t)e$ respectively. Finally, given a hatted sequence of types $\boldsymbol{\sigma}$ consisting of the types $\sigma_1, \ldots, \sigma_n$ we write $\boldsymbol{\sigma} \to \tau$ to mean the type $\sigma_1 \to \ldots \to \sigma_n \to \tau$, ignoring any hats appearing in $\boldsymbol{\sigma}$. So for every hatted sequence of types $\boldsymbol{\sigma}$ we posit a pair of quantifiers:

- $\forall_{\boldsymbol{\sigma}} : (\boldsymbol{\sigma} \to t) \to \breve{\boldsymbol{\sigma}} \to t$
- $\exists_{\boldsymbol{\sigma}} : (\boldsymbol{\sigma} \to t) \to \breve{\boldsymbol{\sigma}} \to t$

Exercise 9.8. *Using only ordered* λ*-terms paraphrase 'Everyone loves someone who loves them back' using the hatted quantifiers.*

Note that we have made no attempt to paraphrase the final sort of example, involving vacuous λ-abstraction. This is because we do not lose any expressive power by rejecting vacuous quantification: we have no need to say 'something is such that snow is white' when we can just say 'snow is white'.

Exercise 9.9. *Come up with your own notation for quantifiers that let you perform vacuous variable binding. (Your answer may take the form of a generalization of the hatting notation presented above.)*

Let's end our discussion with some remarks about parsimony. Is the theory of quantification in general λ-languages unparsimonious, since we have posited infinitely many different primitive quantificational expressions? Note that the full λ-language is in the same position: for each type σ there is a different quantificational expression, $\exists_\sigma$ and $\forall_\sigma$, for expressing generalizations at that type. One might think the situation here is worse, since one needs infinitely many primitive operations to express all the sorts of quantificational claims at a given type: there are infinitely many first-order quantificational expressions, for example. But even this is not that different from the default. There are infinitely many first-order quantifiers apart from the existential and universal quantifiers: *exactly three*, *many*, *few*, *most*, *just as many* and so on. In the full λ-language it is possible to supply logically complex definitions of these quantifiers: for instance, Frege gave us an analysis of *there are just as many Fs and Gs* in terms of higher-order quantification over binary relations that are one-to-one correlations. But many philosophers—even those skeptical of higher-order quantification—have found the listed expressions to be perfectly intelligible: they just express other kinds of first-order quantification, alongside *all* and *some*. A second-order analysis of them is just as inappropriate as a second-order analysis of first-order existential quantification. Someone subscribing to a structured theory of propositions might insist

that, like *some* and *all*, the quantifiers *many*, *most*, *just as many* and our hatted quantifiers like $\exists_{\hat{e}\hat{e}}$ are metaphysically simple, not structured entities involving higher-order quantifiers, preventing us from providing logically complex definitions of them instead of taking them as primitive.

9.5 General higher-order logics

We are now in a position to formulate the notion of a general higher-order logic. We will assume that a logical signature can contain, in addition to the logical constants Λ, any of the hatted quantifiers we introduced in the previous section. These are governed by an axiom and a rule that are analogous to the axiom of universal instantiation and the rule of universal generalization:

$\widehat{\textbf{UI}}$ $(\forall_{\boldsymbol{\sigma}} F)\check{a} \rightarrow Fa$

$\widehat{\textbf{Gen}}$ From $A \rightarrow F[x/a]$ infer $A \rightarrow \forall_{\boldsymbol{\sigma}} F\check{a}$ provided $x \notin FV(A)$.

As before we write a term F followed by sequence of terms for the result of successively applying F to elements of that sequence. In addition, we use a for a hatted sequence of terms where hats are placed above some subset of the terms in the sequence that have the same type, and $a : \boldsymbol{\sigma}$ to mean that the nth element of a and the nth element of $\boldsymbol{\sigma}$ agree about whether they are hatted, and the former has the latter as its type. $[x/a]$ refers to the result of substituting all the hatted elements of $a : \boldsymbol{\sigma}$ with x, assuming here x has the same type as the hatted elements, and $\check{a}$ is the sequence that results from removing any hatted elements from the hatted sequence a, preserving the order of the remaining elements.

Example 9.2. *Suppose that Bxyz means that x is (non-strictly) between y and z. Then the claim that anything is between a and itself can be represented* $(\forall_{\hat{e}e\hat{e}} B)a$. *An instance of* $\widehat{UI}$ *is then:*

$$(\forall_{\hat{e}e\hat{e}} B)a \rightarrow Rbab$$

that is, if anything is between a and itself, then b is between a and b.

On the other hand, suppose I can prove from no assumptions that if A then x is between a and x. Then, by $\widehat{Gen}$, *I can prove that, if A, everything is between a and itself. Our premise is* $A \rightarrow Rxax$ *and the conclusion of the rule is* $A \rightarrow (\forall_{\hat{e}e\hat{e}} R)a$. *This is an instance of* $\widehat{Gen}$.

Comprehension Check 9.6. *The reader should convince themselves that the above examples conform to the formal definitions of* $\widehat{UI}$ *and* $\widehat{Gen}$.

Comprehension Check 9.7. *The reader should convince themselves that the ordinary quantifiers are special cases of hatted quantifiers, and that the ordinary principles of UI and Gen are special cases of our more general axioms* $\widehat{UI}$ *and* $\widehat{Gen}$.

This brings us to the crucial notion of a higher-order theory and logic

Definition 9.8 (General higher-order theory)**.** *Suppose that* Λ' *is a signature of logical constants (including, now, hatted quantifiers). A higher-order theory, T, for the general* λ*-language* $\mathcal{J}(\Sigma \cup \Lambda')$, *is a subset of* $\mathcal{J}'(\Sigma \cup \Lambda')$ *subject to the following constraints:*

Axioms *Every instance of PC1, PC2, PC3,* $\widehat{UI}$, β *and* η *belongs to T.*

Modus Ponens *If* $A, A \rightarrow B \in T$ *then* $B \in T$.

$\widehat{\textbf{Gen}}$ *If* $A \to F[x/a] \in T$ *then* $A \to \forall_{\sigma} F\check{a} \in T$ *provided* $x \notin FV(A)$.

Definition 9.9 (General higher-order logic)**.** *Let* Λ' *and* $\mathcal{J}(\Sigma \cup \Lambda')$ *be as above. A general higher-order logic in* $\mathcal{J}(\Sigma \cup \Lambda')$ *is a general higher-order theory that is also closed under the rule of substitution: if A is in* **L***, and A′ is the result of replacing constants of type* σ *with elements of* $\mathcal{J}^{\sigma}(\Sigma \cup \Lambda')$*, provided no variables get bound upon making that replacement, then A′ is also in* **L**.

For any general λ-language $\mathcal{J}$, and any appropriate class of logical constants for it, Λ', we can define the smallest general higher-order logic for that language and logical signature

Definition 9.10 ($\mathsf{H}^{\mathcal{J}\Lambda'}$)**.** $\mathsf{H}^{\mathcal{J}\Lambda'}$ *is the smallest general higher-order logic for* $\mathcal{J}(\Sigma \cup \Lambda')$.

Usually, there is a standard set of logical constants appropriate for a λ-language $\mathcal{J}$, in which case we just write $\mathsf{H}^{\mathcal{J}}$.

9.6 Application: propositional aboutness and constituency

Let us look at the relationship between the λ-languages we have just defined and certain views about the structure of propositions. One can get a sense for the different positions you can have concerning propositional structure by looking at what notions they take to be intelligible. A fairly modest kind of structuralism takes the notion of a proposition being *about* a particular individual to be intelligible—for instance *Prior studied Buridan* is about *Prior* and *Buridan*. This notion can have analogues at other types: one entity might *involve* another entity, like the operator *snow is white and* involves the proposition that *snow is white*, and the above proposition involves the relation of *studying*. One can take the idea of propositional aboutness or involvement seriously without thinking there is an intelligible notion of an *occurrence* of a constituent. Stronger forms of structuralism, then, might take the notion of constituent occurrence number seriously: *Prior* occurs exactly once in *Prior studied Buridan* and twice in *Prior studied Prior*. Each proposition can then be associated not merely with the set of individuals it is about, but a *multiset* telling us how many times each individual occurred. Yet richer theories of propositional structure—in which the constituents of a proposition have an order—might let us ask whether *Prior* occurs before or after *Buridan* in the proposition *Prior studied Buridan*, associating each proposition with a sequence of individuals. Similar notions at different types can also be posited. Presumably one cannot take constituent occurrence order to be intelligible without also having a notion of aboutness and constituent number, although someone happy with aboutness and constituency need not take constituent order to be intelligible.

We'll begin with the weakest notion of aboutness. One might think it possible to define the notion of a proposition being about an individual in purely logical terms. According to this idea, a proposition is about an individual iff there is a property which when applied to that individuals yields that proposition:

$$\text{About} := \lambda p y(\exists_{e \to t} X.p =_t Xy)$$

The following exercise shows that this definition of aboutness collapses in H:[9]

Exercise 9.10. *Let us say that a proposition p is* about *an individual a iff* $\exists_{e \to t} X(p =_t Xa)$*. Prove, in* H *that aboutness is trivial. For any proposition p and individual a, p is about a (i.e. there is some property X such that* $p =_t Xa$*. Point out where the proof fails in* $\mathsf{H}^{\mathcal{J}_R}$.

Aboutness is often illustrated using language: a proposition is about an individual if the proposition is expressed by a sentence containing a name for the individual—for instance, the earlier proposition is expressed by 'Prior studied Buridan', and this sentence contains a name for Arthur Prior. But this test is not good if you can find sentences that contain names that don't contribute to the proposition expressed by the sentence. So the linguistic test for aboutness is unfriendly to languages with vacuous λ-abstraction: 'Prior is such that snow is white' contains a name for *Prior*, but expresses the proposition that *snow is white*, which is not about *Prior*. For the linguistic test to be successful synonymous sentences cannot contain different constants, but in the presence of vacuous λ-abstraction sentences containing different constants are $\beta\eta$-equivalent. (Similarly, for the proposed definition of aboutness in exercise 9.10 to be adequate, we cannot have true identities of the form $p =_t (\lambda x.p)a$ for all a.) It follows that sublanguages of the relevant λ-language are more suited to theorizing about aboutness.

Proposition 9.3. *If P and Q are relevant and $\beta\eta$-equivalent, then the same constants appear in P and Q.*

Proof. We begin by showing that if relevant terms P and Q are immediately β-equivalent and a appears in P, then a appears in Q. If they are immediately β-equivalent then either P can be obtained from Q by substituting a term of the form $(\lambda x.M)N$with $M[N/x]$, or conversely. Suppose the former. If a appears in $(\lambda x.M)N$ it must appear in M or N. If it appears in M, then it also appears in $M[N/x]$. Suppose a appears in N. Since $(\lambda x.M)N$ is relevant, $x \in FV(M)$, and so N appears in $M[N/x]$, and thus so does a. Conversely, if a appears in $M[N/x]$ it must appear in M or N, and thus must appear in $(\lambda x.M)N$. Immediate α and η-equivalents obviously contain the same constants.

Now, by Proposition 9.2, we know that if P and Q are $\beta\eta$-equivalent they are linked by a chain of immediately β or η equivalent relevant terms. The above ensures that adjacent terms in this sequence have exactly the same constants. □

A similar linguistic test can be constructed for the notion of constituent count, if it is thought to be intelligible: 'Mary admires Mary' expresses a proposition that contains two occurrences of *Mary* since it contains the name 'Mary' twice. This test is no good if synonymous sentences can contain different numbers of occurrences of a name. The following exercise explores a naïve non-linguistic definition of the notion of a proposition containing exactly n occurrences of an individual.

Exercise 9.11. *Let us say that p contains exactly n occurrences of a iff $\exists_{e^n \to t} X.p = Xa \ldots a$ where $e^n \to t = \underbrace{e \to \ldots \to e}_{n} \to t$, the type of an n-ary relation. In $\mathsf{H}^{\mathcal{J}_R}$ show that if p contains exactly n occurrences of a then it contains exactly one occurrence of a (where n is arbitrary).*

In order to have a naïve notion of constituent *count* we must not only ban vacuous λ-abstraction, but also duplicate binding. The following proposition shows that the linear language is more appropriate for theorizing in terms of the naïve notion of constituent count.

Proposition 9.4. *If P and Q are linear and $\beta\eta$-equivalent, then the same constants appear in P and Q the same number of times.*

Proof. This proceeds mostly as above, by showing that immediate β-equivalents contain a the same number of times. If $(\lambda x.M)N$ contains a n times, then for some k and r, with

$n = k + r$, a occurs in M k times, and in N r times. Since x appears free in M exactly once, a must occur $k + r$ times in $M[N/x]$. The converse is similar.

We then appeal to the fact that $\beta\eta$-equivalent linear terms may be linked by a chain of immediately $\beta\eta$-equivalent linear terms. □

Notice that in *relevant* λ-languages, we can re-express a relevant term containing several occurrences of a constant with only one occurrence, using a λ-term that binds multiple variables. However, we can't go the other way, and re-express a term involving a constant once with a relevant term involving it several times, as this would require using vacuous λ-abstracts. Putting it very roughly, this opens up the possibility of a linguistic criterion of constituent count in the relevant λ-language. One could identify the number of occurrences of a in P with the maximum n for which P can be expressed as an n-ary relation R, applied to a n times, $P = Ra \ldots a$.

Note that a parallel claim for ordered terms would state that $\beta\eta$-equivalent ordered terms, P and Q, contain the same constants in the same order. But this claim is not true: $(\lambda x.Rxb)a$ is ordered, and $\beta\eta$-equivalent to Rab, yet the order of constants in the former is R, b, a and in the latter R, a, b. It follows that a metaphysician using ordered terms to represent propositions and relations whose constituents have an order cannot take the order of expressions in a λ-language to represent the order of constituents in the entity they express. It is better to think of the order in which the constants appear in the term $(\lambda x.Rxb)a$ as representing the order in which the structured proposition Rab was built, according to this particular representation of it, not the order in which a and b appear in that proposition. In particular, this term encodes the following instructions for building Rab: first plug b into the second slot of R to make $\lambda x.Rxb$, then plug a into the only remaining slot in $\lambda x.Rxb$ (i.e. the first slot of R) to make $(\lambda x.Rxb)a$, which is just Rab. But we would have built the very same proposition by first plugging a into R's first slot, and then b into the last slot. The following are useful heuristics: (i) λ-terms may be thought of as encoding instructions for building structured entities. Distinct but $\beta\eta$-equivalent terms can be thought of as different instructions for building the same entity, (ii) the order in which the proposition is built need not be the same as the order of the constituents in that proposition. We will return to these issues in the chapters on structure.

9.7 General λ-languages without combinators

By restricting one's language to the relevant, linear or ordered terms, a metaphysician is in a better position to reason naïvely about the notions of constituenthood, and constituent occurrence number and order, thus permitting a variety of more structural pictures of reality. However, all of the general logics we have considered so far are formulated in languages that include combinators. In this section we examine some general λ-languages that have no combinators. In these higher-order logics one can formulate theories that contradict certain theorems of H like Identity and Lifting.

In order to properly assess the status of combinators in structural theories of propositions and relations, it makes sense to first think about the status of bound variables more generally. The difference between the first-order sentences $\forall x \exists y Lxy$ and $\forall z \exists w Lzw$ looks like the paradigm example of a representational artefact. Bound variables are useful insofar as they allow us to disambiguate the two readings of sentences like 'everybody loves somebody' but they also vastly overgenerate representations: no-one seriously believes that this sentence has infinitely many different readings and corresponding differences in reality. This applies

just as forcefully to bound variables in a λ-language: surely, for instance, $\lambda x.Fx$ and $\lambda y.Fy$ denote the same property (as guaranteed by the principle of α-synonymy).[10]

The structuralist could explain these facts through the more general principle that bound variables do not contribute constituents to the entities denoted by the expressions they appear in:

Bound Variables are Representational Artefacts Suppose M is a closed term of a λ-language, and $c_1, \ldots, c_n$ are the constants appearing in M that each denote metaphysically simple entities. Then the entities denoted by $c_1, \ldots, c_n$ are the only simple constituents appearing in the entity denoted by M.

It follows, then, that expressions made entirely out of bound variables are constituentless. This is already suggested by the principle that β-equivalent terms are synonymous: the proposition expressed by $(\lambda Xyz.Xzy)Rab$ is the same as expressed by Rba, suggesting the only things contributing constituents to this proposition are the entities denoted by R, a and b.

Let us call an entity that is constituentless, 'pure'. The idea of a pure entity is quite alien to the classical theory of structured propositions, according to which every entity is a structured complex built from the logically simple entities. But it is also interesting to make logical space for less structural views that reject the existence of pure entities, but who accept λ-terms that trivialize the naïve notion of a constituent, the number of constituent occurrences, and the order of constituent occurrences.

In this section, we will consider a variety of restricted terms that do not admit combinators. The most austere will correspond to those terms you can make from the constants through an operation we call 'Complication' (defined below). Less austere restrictions allow one to close those terms under different combinations of taking weakenings, converses or reflexizations. Corresponding to the class of relevant, affine, linear and ordered terms we shall now introduce subrelevant, subaffine, sublinear and subordered (or 'structural') terms.

We will thus consider four more general λ-languages that, roughly speaking, correspond to the result of removing combinators from the four languages we have already treated. According to the most austere of these languages, the only way to create new terms from old (apart from $\beta\eta$-reduction) is by an operation that generalizes both application and composition, which we call *complication*:

Complication Given $M : \sigma_1 \to \ldots \to \sigma_{m+1} \to \tau$ and $N : \rho_1 \to \ldots \rho_n \to \sigma_{m+1}$ we write $(MN)^n_m : \sigma_1 \to \ldots \to \sigma_m \to \rho_1 \to \ldots \to \rho_n \to \tau$ for the following λ-term:

$$(MN)^n_m \quad := \quad \lambda x_1 \ldots x_m y_1 \ldots y_n . M x_1 \ldots x_m (N y_1 \ldots y_n)$$

Complication has application as a special case (when $n = m = 0$). But a metaphysical picture that has application as the only way of combining entities to make new entities would be arbitrary: We could combine a binary relation, $R : e \to e \to t$, with an individual $a : e$ in its first place, to make a unary property, $Ra : e \to t$, but we cannot combine that relation with that individual in its second slot, i.e. $\lambda x.Rxa : e \to t$, using application alone. We can write $(Ra)^0_m$ for plugging a into the $m + 1$th argument place of R (the $+1$ is because we've been counting argument places starting from 1, not 0). Thus $(Ra)^0_0$ corresponds to putting a in the first argument place of R, $(Ra)^0_1$ into the second argument place of R, $(Ra)^0_2$ into the third place, and so on.

Complication also has composition as a special case (when $n = 1, m = 0$). Composition generally lets us combine something of type $\sigma \to \tau$ with something of type $\tau \to \rho$ to make

something of type $\sigma \to \rho$. For instance, it lets us combine a predicate like *is tall*, $T : e \to t$, with an operator such as negation, $\neg : t \to t$, to make another predicate, in this case *is not tall*, written $(\neg T)^1_0 : e \to t$. But unary composition is also an arbitrary stopping point. We can combine a *binary* relation, like *loves*, $L : t \to t$ with negation to make another binary relation, *doesn't love*, which we notate $(\neg L)^2_0$, which without abbreviations is just $\lambda xy.(\neg Lxy)$. And we could combine negation with a ternary relation R to make another ternary relation $(\neg R)^3_0$, and so on. In the other direction—i.e. when we let n be less than 0 instead of greater than—we get the notion of composing $\neg$ with 0-ary relation. This is of course the result of 'composing' negation with a proposition, so that we see that application is a special case of composition.

Just as we generalized from application in the first argument to application in the mth argument, we can generalize this notion of composition from the first argument place to other argument places. Just as I can compose being tall with negation in one way, because negation has one argument place, I can compose being tall with conjunction in two ways because conjunction has two argument places: $(\wedge T)^1_0$ and $(\wedge T)^1_1$. These amount to $\lambda xp.(\wedge(Tx)p)$ and $\lambda px.(\wedge p(Tx))$. Similarly, there are two ways of complosing $\wedge$ and a binary relation L, $(\wedge L)^2_0$ and $(\wedge L)^2_1$, and so on. One can also see that the previously discussed example of parallel composition can also be understood in terms of complication. We'll give a natural interpretation of complication in terms of a concrete theory of structured propositions in Chapters 11–13.

Observe that every term built from constants using complication is an ordered term, and that no combinator can be made just using complication. One could then consider metaphysical visions in which we have further ways of constructing new entities from old. For instance, one might consider a picture where you can construct converses of relations. Thus, once you have a term $\lambda x_1 \ldots x_k x_{k+1} \ldots x_n.M$ in the language you should also have the term $\lambda x_1 \ldots x_{k+1} x_k \ldots x_n.M$. Or if you thought you can build reflexizations of relations you already have then you should require that if the language contains $\lambda x_1 \ldots x_k x_{k+1} \ldots x_n.M$ then it also contains $\lambda x_1 \ldots y \ldots x_n.M[y/x_k, y/x_{k+1}]$ (provided y doesn't appear in M already). For weakenings, we should include the rule that if $\lambda x_1 \ldots x_n.M$ is in the language then $\lambda x_1 \ldots x_n y.M$ is too (provided y doesn't appear in M already).

In the following exercise, you will show that, in a sense, all combinators arise from the identity combinator and term formation rules mentioned above.

Exercise 9.12. *Suppose that $\mathcal{J}(\Sigma)$ is a general λ-language and its terms are closed under complication and it contains all instances of the identity combinator $\lambda x.x : \sigma \to \sigma$ for every type σ.*

a. Suppose that $\mathcal{J}(\Sigma)$ contains $\lambda x_1 \ldots x_{k+1} x_k \ldots x_n.M$ whenever it contains $\lambda x_1 \ldots x_k x_{k+1} \ldots x_n.M$. Show that the language contains all instances of the C combinator.

b. Suppose that $\mathcal{J}(\Sigma)$ contains $\lambda x_1 \ldots y \ldots x_n.M[y/x_k, y/x_{k+1}]$ whenever it contains $\lambda x_1 \ldots x_k x_{k+1} \ldots x_n.M$ and y does not appear in M. Show that the language contains all instances of the W combinator.

c. Suppose that $\mathcal{J}(\Sigma)$ contains $\lambda x_1 \ldots x_n y.M$ whenever it contains $\lambda x_1 \ldots x_n.M$ and y does not appear in M. Show that the language contains all instances of the K combinator.

Here we offer definitions of three classes of terms.

Definition 9.11 (Subrelevant term)**.** *The subrelevant terms over a signature Σ are defined inductively:*

- *Any constant is subrelevant.*

- *If $M : \sigma \to \tau$ is subrelevant and $x : \sigma$ is a variable then $Mx : \tau$ is subrelevant.*
- *If $M : \sigma \to \tau$ and $N : \sigma$ are subrelevant, then so is (MN).*
- *If M and $x \in FV(M)$, then $\lambda x.M$ is subrelevant.*

Definition 9.12 (Subaffine term)**.** *The sublinear (subaffine) terms over a signature Σ are defined inductively:*

- *Any constant is sublinear (subaffine).*
- *If $M : \sigma \to \tau$ is sublinear (subaffine) and $x : \sigma$ is a variable then $Mx : \tau$ is sublinear (subaffine).*
- *If $M : \sigma \to \tau$ and $N : \sigma$ are sublinear (subaffine), then so is $(MN) : \tau$.*
- *If $M : \tau$ is sublinear (subaffine) and x appears in $FV_m(M)$ exactly once (at most once), then $\lambda x.M : \sigma \to \tau$ is sublinear (subaffine).*

The final class of terms, the subordered terms, correspond to the structured picture of reality described in Section 11.3 and 11.4. Consequently, we shall also call them the structural terms (not to be confused with the use of 'structural' in 'substructural', to appear in Chapter 10).

Definition 9.13 (Structural term)**.** *The subordered or structural terms over a signature Σ are defined inductively:*

- *Any constant is structural.*
- *If $M : \sigma \to \tau$ is structural, and $x : \sigma$ is a variable, then $Mx : \tau$ is structural.*
- *If $M : \sigma \to \tau$ and $N : \sigma$ are structural, then so is (MN).*
- *If $M : \tau$ is structural and $x : \sigma$ is the last element of $FV_s(M)$ and doesn't appear elsewhere in the sequence, then $\lambda x.M : \sigma \to \tau$ is structural.*

As before, each of these classes of terms form the basis for a general λ-language, which is shown be demonstrating that these classes are closed under $\beta\eta$-reduction, substitution and so on.

Observe that our official definition of a structural term makes no reference to the notion of complication. Nonetheless, it may be proved that complication is sufficient to build all the structural terms, up to $\beta\eta$-equivalence.

Proposition 9.5. *If a term is structural in a signature Σ then it is $\beta\eta$-equivalent to a term that can be constructed from the constants and variables using complication alone.*

Conversely, every term constructed from the constants using complication is structural.

The proof of the first claim is a little tricky given the tools we presently have. We will encounter a more elegant characterization of the structural terms in Chapter 10 which will allow a more direct proof of this claim. The second claim is left as an exercise.

Exercise 9.13.

a. Show that if M and N are structural terms, then $\lambda x_1 \dots x_m y_1 \dots y_n.Mx_1 \dots x_m(Ny_1 \dots y_n)$ also structural provided it is well-typed.

b. Show that if R is a sublinear term then $\lambda xy.Ryx$ is too, provided it is well-typed.

9.8 Variable free approaches

In Chapter 3, we provided translations of the full λ-language into a combinatory language. Indeed, we showed there that every closed λ-term was synonymous with a λ-free and variable-free combinatory term containing only S and K, allowing us to do away with λ-abstraction, and variables, altogether if we wanted. Since each of our general λ-languages consists of special sorts of λ-terms, these translations into the SK-combinatory language are still available. But these translations are not reversible: the SK-combinatory language has expressions that are not synonymous with terms of any of the general λ-languages we have considered in this chapter. For instance, the K combinator, considered as a λ-term, is not relevant, linear or ordered (although it is affine), and the S combinator, considered as a λ-term, is not linear, affine or ordered (although it is relevant). In a combinatory language, of course, combinators like S and K, are not short for terms made of λs and bound variables, but are primitive. But they give rise to many of the same issues that arose for λ-terms. For instance, the defining equation for the W combinator, $WRa = Raa$, apparently duplicates the constituent a. So we shall extend the terminology of relevance, linearity, affinity and structurality to them in the obvious way: if their translations in the full λ-language have the corresponding property.

What we would like, then, is combinatory bases for each general λ-language considered so far that have combinators (the subrelevant, structural, etc. languages do not contain combinators and so cannot be paraphrased in terms of combinators). That is, a combinatory basis for the relevant terms that use only relevant combinators (like S and I, but not K), a combinatory basis for the linear terms that uses only linear combinators (like B, C and I, but not S or K), and a combinatory basis for the affine terms that use only affine combinators (like B, C and K, but not S or W). (Curiously, it seems that we cannot give a finitary combinatory basis for the ordered λ-terms using only ordered combinators, so we set it aside in this section; see conjecture A.1 in appendix A.) The issue we encountered above is that the SK basis cannot provide an exact combinatory basis for any of our four general λ-languages, so a different strategy is required.

A closer look at the more fine-grained analysis of ersatz abstraction, presented in Section 3.5, gives us some hints as to how to give combinatory bases for our general λ-languages. Recall that if P was a constant or a variable we defined $[x].P$ to be the I combinator if P was simply identical to the variable x, and we defined it to be Ky or Kc, respectively, if P was a variable y distinct from x, or a constant, c. If we are trying to provide a combinatory substitute for a relevant or linear term, the second two cases never arise: $\lambda x.y$ and $\lambda x.c$ are not relevant or linear. So for the relevant and linear terms, we only need to use the I combinator, which is both relevant and linear. (If we are providing a translation of affine λ-terms using affine combinators, the $\lambda x.y$ and $\lambda x.c$ cases do arise, but in that case, we can use the K combinator because it is affine.)

This shows how to form ersatz abstracts of basic combinatory terms. Now suppose we have a complex combinatory term, PQ, possibly involving the variable x, and we know how to define $[x].P$ and $[x].Q$. Suppose we want to define the ersatz abstract $[x].PQ$. If our goal is ultimately to find combinatory paraphrases of relevant terms using relevant combinators, then we can use the standard trick of rewriting $[x].PQ$ as $S([x].P)([x].Q)$, since S is relevant. If it is instead to find paraphrases of affine or linear terms, then the cases that need paraphrasing are cases where x appears at most once or exactly once in PQ. That means that x cannot occur free in both P and in Q, or it would occur at least twice in PQ. In the

case that x doesn't occur free in Q we can paraphrase $[x].PQ$ using the C combinator, as in Section 3.6:

$$C([x].P)Q$$

Similarly, in the case where x doesn't occur free in P we can use the B combinator:

$$BP([x].Q)$$

B and C are both affine and linear, as required. The reader should consult Section 3.6 to remind themselves why these definitions work.

We may summarize these definitions as follows:

Definition 9.14 (Relevant ersatz abstraction)**.** *Suppose that P is a term of $CL^v(\{S, I, B, C\}, \Sigma)$ and $x \in FV(P)$. Then $[x]_r.P$ is defined as follows:*

- $[x]_r.x = I$
- $[x]_r.PQ = S([x]_r.P)([x]_r.Q)$ *when x is free in both P and Q*
- $[x]_r.PQ = C([x].P)Q$ *when x appears free only in P*
- $[x]_r.PQ = BP([x].Q)$ *when x appears free only in Q*

Since we are focusing on the case $x \in FV(P)$, the cases $[x].y$ ond $[x].c$ do not arise.

Definition 9.15 (Affine ersatz abstraction)**.** *Suppose that P is a term of $CL^v(\{B, C, I, K\}, \Sigma)$ and x appears in $FV_m(P)$ at most once. Then $[x]_a.P$ is defined as follows:*

- $[x]_a.x = I$
- $[x]_a.y = Ky$, $[x]_a.c = Kc$, *when y is a variable distinct from x, and c a constant.*
- $[x]_a.PQ = BP([x]_a.Q)$ *when $x \notin FV_m(P)$*
- $[x]_a.PQ = C([x]_a.P)Q$ *when $x \notin FV_m(Q)$*

The affine case proceeds similarly:

Definition 9.16 (Linear ersatz abstraction)**.** *Suppose that P is a term of $CL^v(\{B, C, I\}, \Sigma)$ and x appears in $FV_m(P)$ exactly once. Then $[x]_l.P$ is defined as follows:*

- $[x]_l.x = I$
- $[x]_l.PQ = BP([x]_l.Q)$ *when $x \notin FV_m(P)$*
- $[x]_l.PQ = C([x]_l.P)Q$ *when $x \notin FV_m(Q)$*

We may define a translation, †, from the relevant, affine or linear λ-language to *SI*, *BCKI*, and *BCI* combinatory languages respectively. In each translation, variables and constants are translated to themselves, i.e. $c^\dagger = c$, $x^\dagger = x$. Similarly, in each translation, $(MN)^\dagger$ is mapped to $M^\dagger N^\dagger$. The only divergence occurs in the case of λ-abstraction. When $\lambda x.M$ is a relevant term then $x \in FV(M)$, and so we may translate it to $[x]_r.M^\dagger$, using definition 9.14. If $\lambda x.M$ is affine (linear) then x appears in $FV_m(M)$ at most (exactly) once, and we may using Definition 9.15 (Definition 9.16) to define $[x]_a.M^\dagger$ ($[x]_l.M^\dagger$).

Proposition 9.6. *Modulo synonymy, there are invertible translations between the* $\{S, I\}$ *combinatory language and the relevant* λ*-language.*

In Section 3.6 we showed that we may define S from B, C and W. Since B, C and W are themselves relevant, the basis $\{B, C, I, W\}$ is another basis of relevant combinators for the relevant λ-terms.

Proposition 9.7. *Modulo synonymy, there are invertible translations between the* $\{B, C, I, K\}$ *combinatory language and the affine* λ*-language.*

In Section 3.3 we also showed that we may define I from C and K. So it follows that $\{B, C, K\}$ is also a basis of affine combinators for the affine λ-terms.

Proposition 9.8. *Modulo synonymy, there are invertible translations between the* $\{B, C, I\}$ *combinatory language and the linear* λ*-language.*

Endnotes

1. Some relevant literature includes Williamson (1985), Fine (2000), Dorr (2004), Gilmore (2013), Dixon (2018), Bader (2020), and McSweeney (2019).
2. Strictly speaking what is needed for a meaningful notion of converse is a way of identifying argument places across different relations, so we can talk about plugging two individuals into two relations in "opposite ways". This does not require the argument places to be ordering as "first", "second" and so on: we do not need to assume, for instance, that in a ternary relation one of the argument places occurs between the other two. Nonetheless, the reason we have a non-arbitrary way to identify argument places across two different binary predicates in language is because of the left-to-right way we represent them—without order, it is far from obvious that we can identify argument places across relations.
3. See Fine (2000), p. 5.
4. Williamson (1985) formulates another puzzle: that the existence of converse relations gives rise to a form of referential indeterminacy, since we could treat 'loves' as referring to a given relation with one convention for assigning the subject and object of the verb to each of the two arguments of the relation, or we could treat 'loves' as referring to the converse of that relation with the opposite convention for associating the subject and object to arguments of the relation.
5. This follows from Functional Choice from Section 6.5, but that too is a contentious principle.
6. See Muskens (2007).
7. See also the view of Hofweber (2016), which develops a related distinction between external and internal quantifiers.
8. Why bother introducing FV_m when we could have gotten by just with FV_s? The connection between multisets and linear and affine terms is important, and it's helpful to highlight it when it occurs.
9. This definition plays a prominent role in the theories of Goodman (2018b) and Dorr (2016).
10. The word 'proposition' is used to refer to many different sorts of things in philosophy. King (2007) has a structured theory of propositions that have bound variables as constituents. But it is likely that King is providing something closer to a theory of structured representations than a theory about the structure of reality in the sense we are exploring here.

10

Curry typing

In the prior chapters we have essentially followed the definitions of Church (1940) characterizing the class of well-formed λ-terms at each type. There is another prominent approach to generating terms and assigning them types due to Curry based on natural deduction systems, which we explore in this chapter. This approach is more flexible than Church's in several respects. Curry's approach is especially useful when investigating general λ-languages for metaphysical applications, as it affords us a lot of control over what sorts of λ-terms we can create. We will see some precise correspondences between certain Curry natural deduction systems and certain general λ-languages considered in Chapter 9.

We begin, in Section 10.1, by introducing Curry's alternative formalism for typing terms of the full λ-language, and highlight some of its distinctive advantages over Church's theory. Section 10.2 relaxes the structural rules allowing us to give alternative characterizations of the general λ-languages from Chapter 9, and Section 10.3 adds syncategorematic rules for the logical connectives, providing a framework that is neutral about the existence of logical operations.

Throughout this chapter, the reader will notice a very suggestive connection between natural deduction systems for non-classical propositional logics and natural deduction systems for typing terms. This connection provides us with some very intuitive models of general λ-languages drawing on some ideas that were originally developed in the context of relevance logic. This material is somewhat off the main track, and so has been relegated to Appendices A and B. The reader who is interested in the connection, or looking for ways to model general λ-languages naturally, is encouraged to read these appendices.

10.1 Curry typing

In Chapter 3 we treated variables and constants in the same way: the type of a variable is specified prior to building a term, in the same way as a constants type is specified. In this textbook, we have adopted the convention of indicating the type of a variable, when it matters, in the surrounding text. Alternative conventions make this fact more overt, by insisting the type of a variable is explicitly attached to it as a superscript—for instance, a variable of type $t \to t$ would be notated $X^{t \to t}$. But the underlying point is that the type of a variable is built into the variable itself.

This rigidity can be a little constraining, and we've often found ourselves abusing it. Consider, for instance, the following pair:

$$(\lambda Zy.Zyy)((\lambda UVz.U(Vz))((\lambda Zxy.Zyx)X)\,Y)$$
$$\lambda y.Xy(\,Yy)$$

DOI: 10.4324/9781003039181-14

both of the above strings of symbols look as though they could be λ-terms if we associated types to X, Y, Z, U, V, x, y and z in just the right sort of way. But note that, without even checking that this is possible, we can know that *if it were*, then on any consistent way of doing so, the above two terms would be β-equivalent. That's because the above strings look enough like typed λ-terms that we can make sense of β-equivalence for them, and indeed, you can get from the former to the latter just by replacing immediate β-equivalents. In fact, the reader may recognize these terms from the discussion of the definability of S in terms of B, C and W and Exercise 3.25. The reader may refer to that for the proof of β-equivalence.

The derivation thus *also* establishes the β-equivalence of two structurally analogous terms that are made by replacing the variables X, Y, Z, U, V, x, y and z with variables of other types that nonetheless made the resulting two strings both into well-formed λ-terms. In fact, we could perform the manipulations without even knowing that the terms we were working with were such that the variables occurring in them could be assigned types that made them into well-formed λ-terms. Consider instead the following expressions, neither of which correspond to well-typed λ-terms:[1]

$$(\lambda XYZ.X(YZ)(ZY))UV$$

$$\lambda Z.U(VZ)(ZV)$$

Exercise 10.1. *Show that for every way of assigning types to X, Y and Z, the above two terms are β-equivalent to $\lambda Z.U(VZ)(ZV)$ by making the (false) assumption that they can be assigned types in a way that makes these two strings λ-terms, and applying what would be valid instances of β if they were.*

The paradigm for associating types with variables we have been using in this book is called *Church typing*. An alternative, *Curry typing*, invented by Haskell Curry, is a way of substantiating many of the shortcuts we made in previous chapters, where we reasoned schematically with terms before knowing what the types of the variables occurring in them were (or, indeed, before checking that the variables could be assigned types that made the relevant terms well-formed). With Curry-typing, variables don't have a built-in type, rather we can choose to associate a variable with a type as we build the term.

Apart from making reasoning with variables less unwieldy, the Curry way of typing terms allows us to more easily specify classes of substructurally typed λ-terms, and reveals connections to non-classical propositional logics that prove to be surprisingly fruitful.

The basic deliverances of Curry typing are *sequents*: statements of the form $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$. This is often read as stating that, given x_1 has type σ_1, …, and x_n has type σ_n, then $M : \tau$ is a term of type τ. $x_1, \ldots, x_n$ will in general be a list that includes all the variables occurring free in M. If you can derive a sequent ending in $M : \tau$ that means M can consistently be given the type τ. Some terms, like $\lambda x.x$, can be assigned multiple types. For instance, one will be able to find derivations of $\lambda x.x : t \to t$ and $\lambda x.x : (t \to t) \to (t \to t)$. (By contrast, in a Church typed λ-language there will be many different versions of the identity combinator using different variables with different inbuilt types.) Other terms, like $\lambda x.xx$, cannot be assigned any types: in this case there is no derivation ending in $\lambda x.xx : \sigma$ for any σ. In the preceding we are thus departing from our earlier use of the word 'term': the things that can be said to have multiple types or no types cannot be the things we called 'terms' in Chapter 3, which, by definition, each had exactly one type. In order to specify our new notion of term, over a signature Σ, we assume an infinite set of variables Var (this time not separated into disjoint subsets Var^σ, as in Chapter 3):

Definition 10.1 (Untyped λ-term)**.** *The untyped λ-terms over the signature Σ are defined as follows:*

- *Every variable or element of the signature is an untyped term.*
- *If M and N are untyped terms so is (MN).*
- *If M is an untyped term, so is $\lambda x.M$.*

Thus, for instance, $\lambda x.x$ is an untyped term. We can think of it as a Church typed term in infinitely many different ways: for each σ I could think of the untyped variable x as being a variable of type σ in the Church setting, yielding a term of type $\sigma \to \sigma$. On the other hand $\lambda x.xx$ is also an untyped term, but there is no way of thinking of it as a Church term by thinking of x as a term of type σ.

The other ingredient in a sequent are the type assignments: the list telling us how to think about the type of each of the variables. We will also refer to these as *premises* of the sequent (a little attention is needed with this terminology, as sequents themselves can also be premises and conclusions in sequent derivations). For the moment we will simply think of them as sequences, or lists of type assignments of the form $x : \sigma$, assigning x the type σ.

In accordance with our stipulations that type assignments are sequences of simple assignments $x : \sigma$, we shall think of (Γ, Δ) as the operation of concatenating the lists Γ and Δ, and ϵ as the empty sequence. Consequently, it ought to be associative. That is, we accept the identity:

$$((\Gamma, \Delta), \Sigma) = (\Gamma, (\Delta, \Sigma))$$

Since ϵ here stands for the empty sequence, we should also accept the identities:

$$(\epsilon, \Gamma) = \Gamma = (\Gamma, \epsilon)$$

Definition 10.2 (Type assignment)**.**

- *ϵ is a type assignment.*
- *x is a variable, and σ a type, then $x : \sigma$ is a type assignment.*
- *If Γ and Δ are type assignments that contain no variables in common, then (Γ, Δ) is as type assignment.*

Observe that the restriction that Γ and Δ have no variables in common entails that type assignments contain a given variable at most once. We will return to the significance of this restriction shortly.

Convention 10.1. *We will adopt the convention of writing $\vdash M : \sigma$ instead of $\epsilon \vdash M : \sigma$.*

The identification of type assignments with sequences is a natural choice for the sorts of substructural phenomena we will be primarily concerned with. However, for some purposes entities with less structure would have been equally appropriate, treating premises as sets or multisets. Certain rules (such as the rule Permutation below) would be automatic and would not need to be specified separately; the downside is that we could not model substructural type theories that relax these rules. In the other direction, we could also choose to use entities with more structure than sequences, for instance by denying the associativity

Table 10.1 Natural deduction for Curry typing

$\dfrac{}{x{:}\sigma \vdash x{:}\sigma}$	**Identity**	$\dfrac{\Gamma \vdash M{:}\sigma\to\tau \quad \Delta \vdash N{:}\sigma}{\Gamma,\Delta \vdash (MN){:}\tau}$	**Application**
$\dfrac{\Gamma,x{:}\sigma,y{:}\tau,\Delta \vdash M{:}\rho}{\Gamma,y{:}\tau,x{:}\sigma,\Delta \vdash M{:}\rho}$	**Exchange**	$\dfrac{\Gamma,x{:}\sigma \vdash M{:}\tau}{\Gamma \vdash \lambda x M{:}\sigma\to\tau}$	**Abstraction**
$\dfrac{\Gamma \vdash M{:}\tau}{x{:}\sigma,\Gamma \vdash M{:}\tau}$	**Weakening**	$\dfrac{\Gamma \vdash M{:}\sigma\to\tau}{\Gamma,x{:}\sigma \vdash Mx{:}\tau}$	**Concretion**
$\dfrac{\Gamma,x{:}\sigma,y{:}\sigma,\Delta \vdash M{:}\tau}{\Gamma,z{:}\sigma,\Delta \vdash M[z/x][z/y]{:}\tau}$	**Contraction**	$\dfrac{c\in\Sigma}{\vdash c{:}\sigma}$	**Constants**

law stated above, or the law that states that ϵ is a unit. This requires us to adopt a rather abstract conception of the type assignments that at this juncture is not pedagogically helpful. But we will briefly look at substructural type theories that reject those laws at the end of Section 10.2.

The laws for deriving sequents for the language $\mathcal{L}(\Sigma)$ are given in Table 10.1 . In the rules Concretion, Weakening and Contraction, the variables introduced in the conclusion sequent (x in the first two cases, z in the latter) must be 'fresh' in the sense that they do not appear in the type assignments to the left of the sequents in the inference (Γ, Δ, etc). In the case of Contraction, the introduced variable z must also be free for x and y in M.

Constants, Abstraction, Application and Concretion are fundamental rules telling you how to introduce constants, form λ-abstracts and how to apply terms to one another—the three basic operations for forming terms. Application and Concretion both govern the syntactic operation of application. (The observant reader may notice that in the presence of Identity and Application, Concretion is redundant; however keeping it as a separate rule greatly simplifies our derivations, and when we later consider relaxing Identity, Concretion will play a more important role.)

The remaining rules are the so-called structural rules. Here we must attend to how we understand sequents. It's tempting to read a sequent like $X : e \to e \to t, y : e, z : e \vdash Xyz : t$ as simply stating that if X has type $e \to e \to t$ and y and z type e then Xyz has type t. But this suggests there is no difference between this sequent and the permuted sequent $X : e \to e \to t, z : e, y : e \to t \vdash Xyz : t$, so that the rule Permutation states something vacuously true. A more perspicuous way to read the sequent would be to think of it as saying: the thing on the right of the sequent can be built from the things on the left by successively combining adjacent entities. The order and number in which the ingredients are given might matter. (By analogy: if one can make chemical A by adding chemicals X, Y and Z together in that order, it does not following that one can make the chemical A by adding X, Z and Y together in that order—the order in which they react with each other might matter.)

To illustrate the importance of the structural rules, it is helpful to see the role they have in deriving the combinators I, K, C and W. Let us begin by focusing on the case of the empty signature, $\Sigma = \emptyset$, rendering the rule Constants inapplicable. Inspecting the rules, we see that there is only one other rule that has no premises: Identity. Thus all proofs must begin with an instance of Identity. The simplest derivation of a closed term is the derivation of the I combinator. In the present context we identify I with the untyped λ-term $\lambda x.x$; as we mentioned earlier, it can be given any type of the form $\sigma \to \sigma$. All derivations have to begin with Identity, and a single application of Abstraction yields the identity combinator:

$$\dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\,\mathrm{Id}}{\vdash \lambda x.x : \sigma \to \sigma}\,\mathrm{Abs}$$

Another simple derivation is that for the *K* combinator, and its converse. Recall that the former operation takes two arguments and returns the first, ignoring the second, and so the latter operation takes two arguments and returns the second, ignoring the first. The untyped term $\lambda xy.x$ can be assigned any type of the form $\sigma \to \tau \to \sigma$, and its converse, $\lambda yx.x$ any type of the form $\tau \to \sigma \to \sigma$.

We'll derive the converse *K* combinator, $\lambda yx.x$, leaving *K* as an exercise. In figuring out how to derive something in the Curry system, it is sometimes helpful to work backwards. Our goal in the above was to derive $\vdash \lambda y \lambda x.x : \tau \to \sigma \to \sigma$. By inspecting the rules in Table 10.1 , you can see that the only way to derive a sequent whose term on the right begins with a λ is to apply Abstraction. Thus the only way to derive $\vdash \lambda y \lambda x.x : \tau \to \sigma \to \sigma$, by applying Abstraction, is to first derive $y : \tau \vdash \lambda x.x : \sigma \to \sigma$. The term on the right of this new sequent also begins with a λ so we may similarly conclude that the only way to derive it is to first derive $y : \tau, x : \sigma \vdash x : \sigma$. To see how to derive this sequent we can now work forwards. Since every proof begins with identity, and since x appears in both the premise and conclusion of the goal sequent $y : \tau, x : \sigma \vdash x : \sigma$, the natural instance to Identity to start with is $x : \sigma \vdash x : \sigma$. But now we can observe that Weakening lets us move directly from $x : \sigma \vdash x : \sigma$ to $y : \tau, x : \sigma, \vdash x : \sigma$ as required. Here is the derivation in full:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\,\text{Id}}{y : \tau, x : \sigma, \vdash x : \sigma}\,\text{Wk}}{y : \tau \vdash \lambda x.x : \sigma \to \sigma}\,\text{Abs}}{\vdash \lambda y.\lambda x.x : \tau \to \sigma \to \sigma}\,\text{Abs}$$

Exercise 10.2. *Derive the sequent* $\vdash (\lambda x.(\lambda y.x)) : \sigma \to \tau \to \sigma$.[i]

So far we have seen that the structural rule Identity is essential for deriving the identity combinator, and Weakening for deriving the *K* combinator and its converse $\lambda yx.x$. A similar relation holds between the rule Permutation and the converse combinator *C*. In this derivation, we also have to use the rule of Application, which we haven't used yet. As usual, we must begin our derivations with instances of Identity (I'll omit the lines above the initial steps of the proof, indicating they were derived by Identity, as they are self-evident):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X : \sigma \to \tau \to \rho \vdash X : \sigma \to \tau \to \rho}{X : \sigma \to \tau \to \rho, y : \sigma \vdash Xy : \tau \to \rho}\,\text{Conc}}{X : \sigma \to \tau \to \rho, y : \sigma, z : \tau \vdash Xyz : \rho}\,\text{Conc}}{X : \sigma \to \tau \to \rho, z : \tau, y : \sigma \vdash Xyz : \rho}\,\text{Perm}}{X : \sigma \to \tau \to \rho, z : \tau \vdash \lambda y.Xyz : \sigma \to \rho}\,\text{Abs}}{X : \sigma \to \tau \to \rho \vdash \lambda z.\lambda y.Xyz : \tau \to \sigma \to \rho}\,\text{Abs}}{\vdash \lambda X.\lambda z.\lambda y.Xyz : (\sigma \to \tau \to \rho) \to \tau \to \sigma \to \rho}\,\text{Abs}$$

As you can see, proofs become space-consuming quite quickly. Although this derivation is the most complex we've encountered, it can be analysed in the same way as the proofs we considered earlier—by working backwards from the conclusion until you reach a sequent you know how to prove. Try working through the above derivation yourself, and then attempt the following exercises.

Exercise 10.3. *Show that the B combinator,* $\lambda XYz.X(Yz)$ *can be given any type of the form* $(\tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$ *(i.e. derive the sequent* $\vdash \lambda XYz.XYz : (\tau \to \rho) \to (\sigma \to \tau) \to \sigma \to \rho$*).*

Exercise 10.4 (Optional)**.** *Notice how, apart from Identity, you don't need to use any structural rules in the previous exercise. Consider how that would change if we did not identify type assignments with sequences, and consequently distinguished between the type assignments* $(X : \tau \to \rho, (Y : \sigma \to \tau, z : \sigma))$ *and* $((X : \tau \to \rho, Y : \sigma \to \tau), z : \sigma)$.

Exercise 10.5. *Derive the B′ combinator, defined as* $\lambda XYz.Y(Xz) : (\sigma \to \tau) \to (\tau \to \rho) \to \sigma \to \rho$.[ii]

It's worth noting that in the presence of Permutation, there are several equivalent ways to state Weakening. Consider, for instance,

$$\frac{\Gamma \vdash M : \sigma}{\Gamma, x : \tau \vdash M : \sigma}\text{Right Weakening} \qquad \frac{\Gamma \vdash M : \sigma}{x : \tau, \Gamma \vdash M : \sigma}\text{Left Weakening}$$

When we work in systems without Permutation we shall need to distinguish them. Indeed, with Permutation, but not without, Weakening is equivalent to Strong Weakening:

$$\frac{\Gamma, \Delta \vdash M : \sigma}{\Gamma, x : \tau, \Delta \vdash M : \sigma}\text{ Strong Weakening}$$

Exercise 10.6. *a. Derive Left and Right Weakening from Strong Weakening without using Permutation.*

b. Briefly explain how it is possible to prove Strong Weakening from Weakening and Permutation.

c. Derive $\vdash \lambda x.\lambda y.x : \sigma \to \tau \to \tau$ *using Right Weakening, Identity and Abstraction (you may not use Left Weakening or Permutation).*

d. Now derive $\vdash \lambda x.\lambda y.y : \sigma \to \tau \to \tau$ *using Left Weakening, Identity and Abstraction (you may not use Right Weakening or Permutation).*

The only structural rule we haven't encountered yet is the rule of Contraction. As you might have guessed, Contraction is related to duplication of variables, just as Permutation is related to reordering of variables and Weakening to vacuous λ-abstraction. Thus Contraction plays a special role in the derivation of combinators, like the W combinator, that contain bound variables that occur more than once. To properly understand this rule we should pause for a moment to examine a qualification in our definition of a type assignment which we passed over earlier: that no type assignment contains a variable more than once. As a result of this, Contraction is more complicated than the other rules. If we didn't have the restriction ruling out duplicate variables in type assignments, we could have a simpler rule:[2]

$$\frac{\Gamma, x : \sigma, x : \sigma, \Delta \vdash M : \tau}{\Gamma, x : \sigma, \Delta \vdash M : \tau}\text{Pseudo Contraction}$$

The problem is that this rule does not make sense given our ban on variables appearing more than once in type assignments. So why do we need the ban on variables appearing more than once? The reason is that, without it, we could derive combinators containing duplicated variables without using either Contraction or Pseudo Contraction, losing the connection between properties of terms like linearity and the structural rules that makes these systems illuminating in the first place.

Exercise 10.7. *Suppose that we allowed variables to appear multiple times in type assignments. Derive the sequent* $\vdash \lambda Y.\lambda x.\lambda x.Yxx$ *without appealing to Contraction or Pseudo Contraction.*

Notice lastly that the rule of Application allows you to take two well-formed sequents and infer an ill-formed sequent. For instance, if Γ and Δ are individually type assignments (thereby containing each variable exactly once), but have variables in common, then Γ, Δ is not a type assignment. The conclusion sequent of Application would then be ill-formed. Thus, when reading the rule of Application, it is implicitly understood that it only applies when the premise *and* conclusion sequents have well-formed type assignments. Similar remarks apply to Contraction, Weakening and Concretion: the new variable, z, in Contraction cannot be a variable that occurs in the type assignment on the left of the sequent, and similarly for the variable x introduced in the rules Concretion and Weakening.

Finally, then, here is a derivation demonstrating that the W combinator, $\lambda Xw.Xww$ can be assigned any type of the form $(\sigma \to \sigma \to \tau) \to \sigma \to \tau$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X : \sigma \to \sigma \to \tau \vdash X : \sigma \to \sigma \to \tau}{X : \sigma \to \sigma \to \tau, y : \sigma \vdash Xy : \sigma \to \tau}\text{Concretion}}{X : \sigma \to \sigma \to \tau, y : \sigma, z : \sigma \vdash Xyz : \tau}\text{Concretion}}{X : \sigma \to \sigma \to \tau, w : \sigma \vdash Xww : \tau}\text{Contraction}}{X : \sigma \to \sigma \to \tau \vdash \lambda w.Xww : \sigma \to \tau}\text{Abstraction}}{\vdash \lambda Xw.Xww : (\sigma \to \sigma \to \tau) \to \sigma \to \tau}\text{Abstraction}$$

This derivation illustrates how the rule of Contraction is used in practice. If you want to derive a sequent for a term involving a variable occurring twice, like Xww the trick is to first derive a sequent involving a term that contains no duplicated variables, Xyz, and then use contraction to substitute both y and z for w.

As before, Contraction has quite limited applicability when Permutation isn't present: it only allows us to merge adjacent variables. There are two ways to merge non-adjacent variables: the result of the merger could be on the left or the right, giving us two rules which we'll call Left Contraction and Right Contraction. These are:

$$\dfrac{\Gamma, x : \sigma, \Delta, y : \sigma, \Theta \vdash M : \tau}{\Gamma, z : \sigma, \Delta, \Theta \vdash M[z/x][z/y] : \tau}\text{Left Contraction}$$

and:

$$\dfrac{\Gamma, x : \sigma, \Delta, y : \sigma, \Theta \vdash M : \tau}{\Gamma, \Delta, z : \sigma, \Theta \vdash M[z/x][z/y] : \tau}\text{Right Contraction}$$

The metaphysician who rejects converses but is happy with merging argument places should not accept both of these rules at once:

Exercise 10.8. *Type the pair of converse terms $\lambda xy.Rxyxy$ and $\lambda xy.Ryxyx$ using Left Contraction and Right Contraction but without using Permutation.*

We will return to the significance of this choice point in Section 12.4—there are some views in which the choice of whether to use Left Contraction or Right Contraction is simply an arbitrary one of choosing a particular convention for imposing order on an unordered collection of argument places.

Remark 10.1. There is a general version of Contraction that lets you merge n variables simultaneously. In which case there will n different versions of the rule for n non-adjacent variables, corresponding to which of the n possible positions the new variable takes in the conclusion sequent.

So far we have exclusively focused on the pure λ-language with the empty signature. In order to add constants we need a final rule, relative to a signature Σ, telling us that constants of type σ are terms of type σ.

$$\frac{}{\vdash c : \sigma}\ c \in \Sigma^{\sigma} \qquad \text{Constants}$$

Like Identity this rule needs no premises. Notice also that since constants are closed terms, the conclusion sequent itself has an empty type assignment for a premise. The following exercise illustrates the rule for Constants and the rule of Contraction together.

Exercise 10.9. *Consider a signature containing a single ternary relation constant $R : e \to e \to e \to t$. Derive the sequent $x : e, y : e \vdash Rxyx : t$.*[iii]

Notice that the right-hand side of a sequent, $M : \sigma$, consists of a term *and* a type. Because terms here are untyped λ-terms, the type component provides important extra information: a term on its own can have many types (or none). Given the untyped term $\lambda x.x$ alone we wouldn't be able to know whether it could be applied to a given constant $c : \sigma$, say.

Given a closed untyped term M and a type σ, such that the sequent $\vdash M : \sigma$ is derivable, there is often a unique way to associate types to the untyped variables appearing in M that turns it into a Church-typed λ-term of type σ.

Exercise 10.10. *For each of the following pairs of untyped λ-term and type, $M : \sigma$, find ways to associate types with the bound variables so that the result is a Church-typed term of type σ.*

a. $\lambda x.x : (t \to t) \to t \to t$

b. $\lambda xy.x : (e \to t) \to t \to e \to t$

c. $\lambda XYy.X(Y(Xy)) : (e \to t) \to (t \to e) \to e \to t$

d. $\lambda X.X(\lambda y.y) : (((e \to t) \to e \to t) \to (e \to t) \to e \to t) \to (e \to t) \to e \to t$

e. $\lambda XYyz.X(Yz)(Xyz) : (t \to e \to e \to t) \to (e \to t) \to t \to e \to e \to t$

Indeed, it's possible to show that, if you have a derivation of $\Gamma \vdash M : \sigma$ there is always at least one way of assigning types to the variables in M that turn it into a Church-typed term. The initial steps of any such derivation will begin with instances of Constants and, more importantly, Identity, $x_i : \sigma_i \vdash x_i : \sigma_i$, one for each variable appearing in M. These instances of Identity will also all involve distinct variables due to the ban on repetitions in type assignments. It is possible to show (although we will not do so) that assigning x_i the type σ_i, in accordance with the instance of Identity used, will yield a well-formed Church-typed term.

Unfortunately, it is not always possible to uniquely reconstruct the types of the variables appearing in M, given M and one of its assignable types alone, as the following exercise demonstrates.

Exercise 10.11. *Show that for any type σ there is a derivation of $\vdash (\lambda Yp.p)(\lambda z.z) : t \to t$, in which z has type σ (and Y has type $\sigma \to \sigma$, and p has type t).*

However, there are many special cases where it is possible.

Proposition 10.1. *Suppose that M is a relevant term, or is in $\beta\eta$-normal form. If you can derive $\vdash M : \sigma$, then there is a unique way of associating types to the bound variables in M that turns it into a Church-typed term of type σ.*

The reader may consult Hindley (1997), Chapter 2 for these results.

Remark 10.2 (Principal types)**.** You may have noticed that, while each of the typable terms we've looked at can be assigned multiple types, they will always have the same form. For instance the types of $\lambda x.x$ include $t \to t$, $e \to e$, $(e \to t) \to (e \to t)$ and so on, but not $e \to t$ or $t \to (e \to t)$. It's types all have the form $\sigma \to \sigma$ for some σ. Similarly, $\lambda xy.x$ only has types of the form $\sigma \to \tau \to \sigma$, and so on. We will call the 'form' that these types all have in common their *principal type*. (It can be formally represented by introducing type variables, $a, b, c, \ldots$ that behave just like new base types, so that the principal type of $\lambda x.x$ can be represented by $a \to a$, $\lambda xy.x$ by $a \to b \to a$, and so on.)

It turns out this is no accident: for any term M whatsoever, the types assignable to M will share a common form (vacuously so if M cannot be assigned any types). That is, if they can be assigned a type at all they can be assigned a unique principal type. Although we won't explore these results further, they turn out to be very important in computer science for programming in typed languages, as it is quite common to write programs without explicitly typed variables, leaving it to the compiler to figure out what the types of the variables should be.[3]

Now we have examined the structural rules, let us return to the relation between Concretion and Application. Both rules allow one to apply one term to another. Concretion lets you apply a functionally typed term to a special sort of argument term—a variable—without having explicitly construct the argument first. Application allows you to apply a functionally typed term to an arbitrary argument term provided that it has already been constructed. But Concretion can now be seen to be redundant: it is always possible to construct variables using Identity, so that any job you can do using Concretion can be done in a more circuitous way using Identity and then Application. In particular, any derivation that uses Concretion

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash M : \sigma \to \tau\end{array}}{\begin{array}{c}\Gamma, x : \sigma \vdash Mx : \tau\\ \vdots\end{array}}\ \text{Concretion}$$

can be replaced by the following derivation that uses Application and Identity instead:

$$\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash M : \sigma \to \tau\end{array} \qquad \dfrac{}{x : \sigma \vdash x : \sigma}\ \text{Identity}}{\begin{array}{c}\Gamma, x : \sigma \vdash Mx : \tau\\ \vdots\end{array}}\ \text{Application}$$

In this sense, the rule Concretion is redundant: any derivation using it can be replaced by a derivation of the same thing that doesn't use it.

Definition 10.3 (Admissible Rule)**.** *A rule*

$$\dfrac{\Gamma_1 \vdash M_1 : \sigma_1 \qquad \ldots \qquad \Gamma_n \vdash M_n : \sigma_n}{\Delta \vdash N : \tau}$$

is admissible in a given system iff, whenever there is a derivation of $\Gamma_1 \vdash M_1 : \sigma_1 \ldots \Gamma_n \vdash M_n : \sigma_n$*, then there is also a derivation in that system of* $\Delta \vdash N : \tau$.

Let's end this section by considering a simplification of our type system which does not have structural rules. What remains of this section is strictly speaking optional: the rest of

the chapter will not rest on it. However, it relates the type system we have been studying here to a more common, but equivalent, formulation of the Curry type system that you are most likely to encounter up opening a textbook on the typed λ-calculus that is not concerned with substructural type theories.[4] I will consequently call it simplified Curry-typing. Apart from introducing this system, there are a few more challenging exercises at the end of the section which are instructive.

The simplified system has the rule of Abstraction exactly as stated in Table 10.1 , and three new rules:

$$\frac{}{\Gamma, x : \sigma, \Delta \vdash x : \sigma} \quad \text{Var}$$

Unlike Identity, this rule allows side premises Γ and Δ. It is clear that given Weakening, Permutation and Identity, Var is an admissible rule in our system. Instead of Constants we have

$$\frac{}{\Gamma \vdash c : \sigma} \quad \text{Constants}^*$$

where $c \in \Sigma^\sigma$. It is also clear, given Weakening, that Constants* is admissible. Instead of Application you have:

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash (MN) : \tau} \quad \text{Application}^*$$

In Application*, the same type assignment Γ, appears on the left both for the premises of the inference and the conclusion. Application, by contrast, has different type assignments in the two premise sequents, and you join them in the conclusion.

Exercise 10.12. *Given a derivation of $\Gamma \vdash N : \sigma$, using our Curry-typing system (Table 10.1), you can construct a variant derivation of $\Gamma' \vdash N' : \sigma$, where Γ' is the result of replacing the variables in Γ with new variables, and N' the result of the corresponding substitution of the free variables in N with the new ones.*

Given derivations of $\Gamma \vdash M : \sigma \to \tau$ and $\Gamma \vdash N : \sigma$, show that you can derive $\Gamma, \Gamma' \vdash (MN') : \tau$, and explain how you could use this to derive $\Gamma \vdash MN : \tau$. Finally, explain how Application* is an admissible rule in our type system.

It follows that every rule of the simplified system is an admissible rule in our system. Conversely, in the simplified system the structural rules Weakening, Contraction, Exchange are admissible rules.

Theorem 10.1. *Weakening, Contraction and Exchange are admissible rules in the simplified Curry-typing system.*

I'll show that Contraction is an admissible rule. Weakening and Permutation are left as exercises. The idea is to show by induction on the derivation length that any derivation of the premise sequent of Contraction, $\Gamma, x : \sigma, y : \sigma, \Delta \vdash M : \tau$, can be transformed in to a derivation of the conclusion sequent, $\Gamma, z : \sigma, \Delta \vdash M[z/x][z/y] : \tau$. The base case occurs when $\Gamma, x : \sigma, y : \sigma, \Delta \vdash M : \tau$ is an instance of Var. Thus the term M the right of the sequent must be a variable. We have three cases depending on whether the variable on the right of the sequent is identical to one of the two variables, x and y, in the type assignment being contracted (cases 1 and 2) or not (case 3). We need to show each of the following conditionals.

1. If $\Gamma, x : \sigma, y : \sigma, \Delta \vdash x : \sigma$ then $\Gamma, z : \sigma, \Delta \vdash x[z/x][z/y] : \sigma$
2. If $\Gamma, x : \sigma, y : \sigma, \Delta \vdash y : \sigma$ then$\Gamma, z : \sigma, \Delta \vdash y[z/x][z/y] : \sigma$
3. If $\Gamma, x : \sigma, y : \sigma, \Delta \vdash w : \tau$ then $\Gamma, z : \sigma, \Delta \vdash w[z/x][z/y] : \tau$

For 1, $x[z/x][z/y] = z$, so it holds. 2 is similar. 3 holds because w must occur in Γ, Δ. The Application* step is similar. Suppose we have derived $\Gamma, x : \sigma, y : \sigma, \Delta \vdash (MN) : \tau$ by an instance of Application*. So in the previous step we have $\Gamma, x : \sigma, y : \sigma, \Delta \vdash M : \sigma \to \tau$ and $\Gamma, x : \sigma, y : \sigma, \Delta \vdash N : \sigma$. By the inductive hypothesis we know that we can find derivations of $\Gamma, z : \sigma, \Delta \vdash M[z/x][z/y] : \sigma \to \tau$ and $\Gamma, z : \sigma, \Delta \vdash N[z/x][z/y] : \sigma$. So by applying Application to these derivations we get a derivation of $\Gamma, z : \sigma, \Delta \vdash (M[z/x][z/y])(N[z/x][z/y]) : \tau$, and $(M[z/x][z/y])(N[z/x][z/y]) = (MN)[z/x][z/y]$. The Abstraction step is similar.

Exercise 10.13 (Weakening)**.** *Consider the operation that takes a derivation D in the simplified system, and replaces each step of the derivation, $\Gamma \vdash M : \tau$, with the sequent $\Gamma, y : \sigma \vdash M : \tau$ (where y does not appear anywhere in the original derivation). Show, by induction on derivation length, that the result of applying this operation to a derivation is also a derivation of the simplified system.*

Conclude that the rule Weakening is an admissible rule in the simplified system.

A useful fact to note about the simplified system is that if you have a derivation of $\Gamma \vdash M : \sigma$ then every sequent above it in the derivation will have type assignments extending Γ on the left. I.e. they will be of the form $\Gamma, \Delta \vdash N : \tau$ for some Δ. Moreover, all the instances of Var will have the exact same type assignment on the left. This follows from the simple fact that the rules Var and Application* preserve the type assignment, and Abstraction only allows you to shorten the type assignment by lopping off a variable from the right (i.e. moving from $\Gamma, x : \sigma$ to Γ).

Exercise 10.14 (Permutation)**.** *Suppose we have a derivation in the simplified system of $\Gamma, x : \sigma, y : \tau, \Delta \vdash M : \tau$. Consider the operation that replaces each step of the derivation, $\Gamma, x : \sigma, y : \tau, \Delta' \vdash N : \rho$, with the sequent $\Gamma, y : \sigma, x : \tau, \Delta' \vdash N : \rho$. Show, by induction on derivation length, that the new derivation is also a derivation of the simplified system.*

Conclude that the rule Permutation is an admissible rule in the simplified system.

Another significant fact about the simplified system is that there is at most one derivation of any given sequent $\Gamma \vdash M : \sigma$. This can be seen by induction on the complexity of the term M: if M is a variable, x, Γ is a sequence of variable assignments including $x : \sigma$, and corresponds to a unique instance of Var. If M is of the form (PQ), then $\Gamma \vdash (PQ) : \tau$ comes from an instance of Application*.

10.2 Substructural Curry typing

You may have noticed that the rule Weakening allows us to type terms containing vacuous λ-abstracts, like the K combinator and its converse. Similarly, Contraction allows us to type terms abstracting on multiple occurrences of a single variable, and Permutation, to abstraction on variables in a different order from the order in which they appear in the body. This suggests that there ought to be a tight connection between the classes of substructural λ-terms considered in Chapter 9, and the terms that can be typed using certain classes of

structural rules. In anticipation of this connection, let us give some names to some of some weakened systems, obtained by dropping certain structural rules:

Definition 10.4 (Substructural type theories)**.** *We will give names to the result of dropping certain rules from the Curry type system:*

1. Relevant type theory *is the result of dropping Weakening.*
2. Affine type theory *is the result of dropping Contraction.*
3. Linear type theory *is the result of dropping Weakening and Contraction.*
4. Ordered type theory *is the result of dropping Weakening, Contraction and Permutation.*
5. Subrelevant, subaffine, sublinear *and* structural *type theory are the results of also dropping Identity from relevant, affine, linear and ordered type theory.*

There are actually six more systems you could get by dropping Permutation from relevant, affine or linear type theory, and their Identity free variants, but they are less well behaved due to the restrictions on the ways we can apply Weakening and Contraction without Permutation, and we will have fewer philosophical applications for them. We will, however, return to systems that lack Permutation and have one of the generalizations of the contraction rules Left Contraction or Right Contraction in Section 12.4. We will later also consider dropping Identity, so it will be convenient to have a general notation for all of the possible combinations of the rules.

Definition 10.5. *For any $X \subseteq \{Identity, Permutation, Contraction, Weakening\}$, we write $\mathbf{ND}[X]$ for the Curry-typing system with exactly the structural rules in X, and $\mathcal{L}[X](\Sigma)$ for the resulting language.*

Thus relevant type theory is $\mathbf{ND}[IPC]$, affine type theory is $\mathbf{ND}[IPW]$, linear type theory is $\mathbf{ND}[IP]$ and ordered type theory is $\mathbf{ND}[I]$.

The anticipated connection to the last chapter is, of course, a connection between relevant/affine/linear/ordered type theory and relevant/affine/linear/ordered terms. In order to make the connection precise it is convenient to extend the notion of relevance, affinity, linearity and order to sequents, as well as terms.

We will write $var(\Gamma)$ for the set of variables appearing in Γ, $var_m(\Gamma)$ and $var_s(\Gamma)$ for, respectively, the multiset and sequence of variables you get by dropping the types and colons from Γ. $A \subseteq_m B$ denotes multiset containment: every member of A occurs in B at least as many times as it occurs in A.

Definition 10.6. *A sequent $\Gamma \vdash M : \sigma$ is*

1. *(Sub)relevant iff M is a (sub)relevant term, and $FV(M) = var(\Gamma)$*
2. *(Sub)affine iff M is an (sub)affine term, and $FV_m(M) \subseteq_m var_m(\Gamma)$*
3. *(Sub)linear iff M is a (sub)linear term, and $FV_m(M) = var_m(\Gamma)$*
4. *(Sub)ordered iff M is a (sub)ordered term, and $FV(M)_s = var_s(\Gamma)$*

This defines eight properties of sequents (depending whether we include the word 'sub').

Our theorem now has a simple statement.

Theorem 10.2. $\Gamma \vdash M : \sigma$ *is derivable in:*

1. *(Sub)relevant type theory iff it is (sub)relevant.*
2. *(Sub)affine type theory iff it is (sub)affine.*
3. *(Sub)linear type theory iff it is (sub)linear.*
4. *(Sub)ordered type theory iff it is (sub)ordered.*

Proof. We begin with the left to right part of the proof. It is proved by induction on length of proof.

Base case (Identity): This only applies to the relevant, affine and linear ordered cases. $x : \sigma \vdash x : \sigma$ is clearly ordered and thus relevant, affine and linear.

Base case (Constants): $\vdash c : \sigma$ clearly has all eight defined properties.

Step (Application case). Suppose that $\Gamma \vdash M : \sigma \to \tau$ and $\Delta \vdash N : \sigma$ are both relevant (resp. affine, linear, ordered, subrelevant, subaffine, sublinear, structural). Then it is easily seen that $\Gamma, \Delta \vdash (MN) : \tau$ is also relevant (resp. affine, linear, ordered, subrelevant, subaffine, sublinear, structural). E.g., if they are both ordered, then, $FV_s(M) = var_s(\Gamma)$ and $FV_s(N) = var_s(\Delta)$, then $FV_s(MN) = var_s(\Gamma, \Delta)$, as required. Moreover,$MN$ is an ordered term if M and N are. If both sequents are affine then M and N are affine terms, and thus so is MN. Also $FV_m(M) \subseteq_m var_m(\Gamma)$ and $FV_m(N) \subseteq_m var_m(\Delta)$, so $FV_m(MN) = FV_m(M) \cup_m FV_m(N) \subseteq_m var_m(\Gamma) \cup_m var_m(\Delta) = var_m(\Gamma, \Delta)$ as required. The arguments for relevance, linearity and all the others proceed similarly.

Step (Abstraction case). Suppose that $\Gamma, x : \sigma \vdash M : \tau$ is relevant (resp. affine, linear, ordered, subrelevant, subaffine, sublinear, structural). Then $\Gamma \vdash \lambda x.M : \sigma \to \tau$ is also relevant (resp. affine, linear, ordered, subrelevant, subaffine, sublinear, structural). E.g., if the former sequent is ordered, then $FV_s(M) = var_s(\Gamma, x : \sigma)$. Since x is the last variable in the sequence $FV_s(M)$, $\lambda x.M$ is a ordered term. Also $FV_s(\lambda x.M) = FV_s(M) \setminus x = var_s(\Gamma, x : \sigma) \setminus x = var_s(\Gamma)$ as required. The other cases are established similarly.

Step (Contraction case). This only applies to the relevant and subrelevant type theory. Suppose $\Gamma, x : \sigma, y : \sigma, \Delta \vdash M : \sigma$ is relevant (subrelevant). Since $var(\Gamma, x : \sigma, y : \sigma, \Delta) = FV(M)$ it follows that $var(\Gamma, z : \sigma, \Delta) = FV(M[z/x, z/y])$. (Notice how this step fails for affinity, linearity and structurality.) Note that any substitution of free variables, including $[z/x, z/y]$, preserves the property that λs bind at least one occurrence of a variable, thus preserving relevance of the term M.

Step (Weakening case). This only applies to affine type theory. Suppose $\Gamma \vdash M : \sigma$ is an affine sequent. Then $\Gamma, x : \sigma \vdash M$ is affine too, since $FV_m(M) \subseteq_m var_m(\Gamma)$, and $var_m(\Gamma) \subseteq_m var_m(\Gamma, x : \sigma)$.

Step (Permutation). This applies to relevant, affine and linear type theory and their 'sub' variants. The move from $\Gamma, x : \sigma, y : \tau, \Delta \vdash M : \rho$ to $\Gamma, y : \tau, x : \sigma, \Delta \vdash M : \rho$ clearly preserves relevance, affinity and linearity as $var_*(\Gamma, x : \sigma, y : \tau, \Delta) = var_*(\Gamma, y : \tau, x : \sigma, \Delta)$ when $* = m$ or $* = s$.

Step (Concretion): Suppose that $\Gamma \vdash M : \sigma \to \tau$ is relevant (resp. affine, linear, ordered, subrelevant, subaffine, sublinear, structural). Then $\Gamma, x \vdash Mx : \tau$ is too. Consider, for instance, the structural case: $FV(Mx) = (FV_s(M), x : \sigma) = (var(\Gamma), x : \sigma)$, and moreover since M is structural so is Mx.

This completes the induction.

To prove the converses of each of these claims, we proceed by induction on the structure of M (as a term of the untyped λ-calculus).

Base case: M is a lone variable, x (this only applies to the relevant, linear, affine and ordered cases). If $\Gamma \vdash x : \sigma$ is relevant, linear or ordered $\Gamma = \{x : \sigma\}$ and the sequent is provable by Identity. If it is affine, then $x : \sigma$ appears in Γ, and $\Gamma \vdash x : \sigma$ is provable from $x : \sigma \vdash x : \sigma$ by Weakening and Exchanging the premises until we get Γ. The case where M is a constant is trivial.

Step (MN). Suppose $\Gamma \vdash (MN)$ is a linear sequent, so that $var_m(\Gamma) = FV_m(MN)$. It follows that $FV_m(M)$ and $FV_m(N)$ divide $var_m(\Gamma)$ into two: so let Γ_1 and Γ_2 be type assignments such that $var_m(\Gamma_1) \cup_m var_m(\Gamma_2) = var_m(\Gamma)$, and $var_m(\Gamma_1) = FV_m(M)$ and $var_m(\Gamma_2) = var_m(N)$. $\Gamma_1 \vdash M$ and $\Gamma_2 \vdash N$ are thus linear sequents, and derivable by inductive hypothesis. So by Application we can derive $\Gamma_1, \Gamma_2 \vdash MN$. Finally, Γ is a rearrangement of Γ_1, Γ_2 so $\Gamma \vdash MN$ is derivable by Permutation.

If $\Gamma \vdash (MN)$ is ordered, $FV_s(MN) = var_s(\Gamma)$, so Γ can be split up into two sequences Γ_1, Γ_2 involving just the free variables in M in the first case, and N in the second. It's clear that $\Gamma_1 \vdash M : \sigma \to \tau$ and $\Gamma_2 \vdash N : \sigma$ are also ordered, and so are derivable by inductive hypothesis. Thus $\Gamma \vdash (MN)$ is derivable by an application of Application. The linearity condition is checked as above.

The cases of affinity and relevance are similar.

Step Mx. Only the subrelevant, sublinear, subaffine and structural terms have this clause. We will just consider the subrelevant case. Suppose $\Gamma \vdash Mx : \tau$ is subrelevant. If follows that x appears in Γ exactly once so if Γ' is the result of removing x from Γ, $\Gamma' \vdash M : \tau$ is subrelevant, and thus derivable in the subrelevant system by inductive hypothesis. Thus $\Gamma', x : \sigma \vdash Mx : \tau$ is also derivable by Concretion, and finally $\Gamma \vdash Mx : \tau$ is derivable by some number of applications of Permutation. The remaining cases can be proved similarly.

Step $(\lambda x.M)$. Suppose $\Gamma \vdash \lambda x.M : \sigma \to \tau$ is ordered (linear, affine, relevant). Then $\Gamma, x : \sigma \vdash M : \tau$ is also easily seen to be ordered (linear, affine, relevant), and so is provable by the inductive hypothesis. $\Gamma \vdash \lambda x.M : \sigma \to \tau$ can then be obtained by an instance of Abstraction. □

Note that a closed term is typable exactly if $\vdash M : \sigma$. So a corollary of Theorem 10.2 is that the closed relevant (affine, linear, etc.) terms are exactly those derivable in relevant (affine, linear, etc.) type theory.

At the beginning of Chapter 9 we briefly discussed the structured view that every proposition, property, relation, etc. could be decomposed into a unique *outer relation* applied to a number of arguments, its *immediate constituents*. This picture is closely connected with the structural (or subordered) system $\mathbf{ND}[]$, which prevents you from building things like converses and reflexizations that allow you to decompose a proposition in two or more different ways with distinct outer relations and rejects the Identity rule which allows you to create argument places in the position that would normally be occupied by an outer relation, such as in $\lambda X.Xab$. The system $\mathbf{ND}[]$ has a special property that makes it particularly convenient to work with: in $\mathbf{ND}[]$ every derivable sequent has a unique derivation. This means that it is often possible to do recursive definitions on terms in $\mathbf{ND}[]$ on the length of derivation, rather than on the structure of terms. This is useful when we are interested in assigning extensions to terms of $\mathbf{ND}[]$ (see Theorem 13.2), or attempting to define translations from the structural terms to other languages.

Definition 10.7 (Derivation length)**.** *The length of a derivation is defined as follows.*

1. *A single application of a rule that needs no premises (such as Constants) has length 1.*
2. *Derivations ending in a rule that has one subderivation (Concretion, Abstraction) have length 1 plus the length of their subderivation.*
3. *A derivation ending in a rule that has two subderivations (Application) has length 1 plus the maximum of the lengths of the two subderivations.*

Later we will encounter rules for logical operations. The rules for $\forall$ *and* $\neg$ *have one subderivation and so are treated like Concretion and Abstraction, and* $\wedge$, $\vee$ *or* $\rightarrow$ *have two subderivations and so are treated like application above.*

Proposition 10.2. *If* $\Gamma \vdash P : \sigma$ *is a derivable sequent in* $\mathbf{ND}[]$ *then it has a unique derivation.*

Proof. First, observe that it is not possible to derive a typing statement of the form $\Gamma \vdash x : \sigma$ where x is a variable, since none of the rules introduce a lone variable on the right. We can then prove by induction on n that every term that has a derivation of length $\leq n$ has a unique derivation of length $\leq n$ up to variable relabeling. Since this holds for every n, each term that has a derivation must have a unique derivation of any length whatsoever.

Derivations of length one are instances of Constants, and so are uniquely determined by the constant.

Consider a derivable typing statement $\Gamma \vdash P : \sigma$. Since we have ruled out the case where P is a variable, P must either be a constant, a λ-term (have the form $\lambda x.M$), or an application term (have the form (MN)). In the final case, N can either be a variable or another sort of term. We consider each possible case in turn.

- If it is a constant, then it could only have been produced by the rule Constants.
- Any two derivations of length $\leq n+1$ of a λ-term $\lambda x.M$ must end with an application of Abstraction, preceded an $\leq n$-length derivation of M. Both of these shorter derivations must be identical by the inductive hypothesis, so the derivations of $\lambda x.M$ must be identical.
- Any two derivations of length $\leq n+1$ of an application term (MN) where N is a variable, end with an application of Concretion preceded by $\leq n$-length derivations of M, which are the same by the IH (it cannot be derived from Application, since that would require a derivation of N which we observed is impossible when N is a variable).
- Any two derivations of length $\leq n+1$ of an application term (MN) where N is not a variable could only have been derived by an application of Application preceded by $\leq n$-length derivations of M and of N, which are unique for M and N among those lengths by the IH. Either way derivations of length $\leq n+1$ of MN are unique. □

Notice that all of the systems we've considered in this section type the B combinator, and as we saw in Exercise 10.4 this is ensured by the fact that we took concatenation of type assignments to be associative: $((\Delta, \Gamma), \Sigma) = (\Delta, (\Gamma, \Sigma))$. It is fairly easy to relax this assumption about type assignments and consider substructural systems that are even weaker than the ones considered here in which one has more control over what sorts of composition are permitted. The reader may consult Restall (2000), Chapters 2 and 7, especially the table on p. 27, for more ideas along these lines.

We'll end our discussion of the substructural rules by examining a notable feature of the λ-languages obtained by dropping Identity from the structural rules. Recall that in Section 7.6 we showed that in the full λ-language it was possible to define intensional notions, such at $\Box$ and $=_t$, from extensional notions, such as $\forall_\sigma$ and $\rightarrow$. In Identity-free substructurally defined languages such reductions of intensional to extensional notions are no longer possible. To show this we will introduce the metalinguistic abbreviation $M \sim_\sigma N$ to mean M and N are coextensive: $A \sim_t B$ stands for $A \leftrightarrow B$, $a \sim_e b$ for $a =_e b$, and $F \sim_{\sigma\rightarrow\tau} G$ for $\forall_{\hat{\sigma}\hat{\sigma}} xy(Fx \sim_\tau Gy)$.[5] We say an n-ary relation R is extensional whenever coextensive arguments are substitutable *salve veritate*: if $a_1 \sim_{\sigma_1} b_1$, $\ldots a_n \sim_{\sigma_n} b_n$ are true, then $Ra_1 \ldots a_n \leftrightarrow Rb_1 \ldots b_n$ is true.[6]

Exercise 10.15. *Associate each sequent* $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ *with the closed term* $\lambda x_1 \ldots x_n.M : \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \tau$.

a. *Explain why a closed term is typable in* $ND(PWC)$ *if and only if it is associated (in the sense above) with a derivable sequent of* $ND(PWC)$.

b. *For each rule of* $ND(PWC)$*, apart from Constants, show that if the terms associated with the premise sequents of that rule are extensional, so is the term associated with the conclusion.*

c. *Conclude that it is impossible to define intensional expressions from extensional primitives in* λ*-languages based on* $ND(PWC)$*. Briefly explain why this extends to* $ND(PW)$*,* $ND(PC)$*,* $ND(WC)$*,* $ND(P)$*,* $ND(W)$*,* $ND(C)$ *and* $ND()$*. Briefly explain why it does not extend to the corresponding systems containing Identity.*

10.3 Curry typing for logical operations

In first-order logic quantifiers are variable binding devices. In our presentation of first-order logic in Example 1.3 we treated the first-order quantifiers as constants with type $(e \rightarrow t) \rightarrow t$ and left the job of binding variables to λ. This is a clean division of labor when we are working with the full λ-language, but this setup is less than ideal when we are restricted in the ways that λ can bind, as we saw in Section 9.4. For instance, to formalize things like 'everyone loves themselves', 'everyone loves someone who loves them back', and so on, we need quantifiers to bind variables with more than one occurrence or bind variables in a different order in the body from the quantificational order, and this is not possible if λ-binding is restricted and our only means for binding variables. Moreover, the reasons we considered to restrict the behaviour of λ came from a metaphysics that rejected converse relations, like *loves* and *is loved by*, or reflexized properties like *loves oneself*, but these reasons do not directly give us reason to be skeptical of propositions like *everyone loves themselves* and *everyone loves someone who loves them back*, which needn't involve the metaphysically contentious relations and properties as constituents. Indeed, we need a way to simulate these quantificational idioms if we are to do any sort of substantive theorizing in these languages. Thus it might be convenient, in general λ-languages, to treat quantifiers as another kind of variable binding device.

There may be philosophical reasons to do this as well. A standard way to deal with logical words like 'if', 'and', 'all', 'some' and so on, is to have special constants in the language for them: $\wedge : t \rightarrow t \rightarrow t$, $\exists_e : (e \rightarrow t) \rightarrow t$, and so on. This presupposes the existence of special entities corresponding to these words—for instance, a connective of type $t \rightarrow t \rightarrow t$ to be the denotation of $\wedge$—and perhaps also a constituent in a conjunctive proposition corresponding to it. This is not, for instance, how we treat application: we do not need to

suppose that there must first exist an entity $\lambda Xy.Xy : (e \to t) \to e \to t$ in order to apply F to a—the structural language has the sentence Fa but not $\lambda Xy.Xy$: that way lies a regress, for then we need a higher-order operation of application to in order to apply application $\lambda Xy.Xy$ to F and a, and this leads to sequence of further hypothesis that clearly has no end (see Bradley's regress Bradley (2016)). This would be bad for a structured theorist who wished to distinguish propositions containing the different typed versions of application as constituents. Such a structured theorist should not assume that application must be a constituent of a proposition of the form Fa, in addition to property F and individual a. One could have a similar view about conjunction: once you have P and Q it is possible to combine them conjunctively to make $(P \wedge Q)$, but that does not mean we must posit a special entity of conjunction which is a further constituent of this proposition in addition to P and Q. Early proponents of structural theories of propositions, such as Wittgenstein (1922), Russell (1918), and Ramsey (1927) held this view about logical words. They would often draw the analogy between logical worlds like 'and' and the English copula 'is': according to them 'is' doesn't stand for anything in 'Socrates is wise', it is simply there to indicate the way in which the meanings of 'Socrates' and 'wise' are being combined. The corresponding view about 'and' in 'Socrates is wise and Socrates is old' would be that it doesn't contribute a constituent to the proposition expressed by this sentence but rather indicates the way in which the two sentences flanking it should be combined—not by application in this instance, but by conjunction. We can represent the non-commital view of conjunction by adding a rule to our type system as follows:

$$\frac{\Gamma \vdash A : t \qquad \Delta \vdash B : t}{\Gamma, \Delta \vdash (A \wedge B) : t}$$

As usual, Γ and Δ must be disjoint sequences of variables. Call the above rule $\wedge$. Following our conventions about naming Curry systems we can write $\mathbf{ND}[X\wedge]$ for a system containing the rules X as well as $\wedge$.

Note that adding this rule to the full type system $\mathbf{ND}[PCIW]$ will allow you to recover a conjunction connective of type $t \to t \to t$, but if you add it to a system without Identity, such as the structural system, this is not always possible.

Exercise 10.16.

a. Construct the sequent $\vdash \lambda p.\lambda q.(p \wedge q) : t \to t \to t$ *in* $\mathbf{ND}[PCIW\wedge]$

b. Briefly explain why you cannot construct this sequent in a system without Identity such as $\mathbf{ND}[PCW\wedge]$.

The rules for the other connectives are similar. Rules for quantifiers should let us bind multiple occurrences of a variable. Given a type assignment Γ write $\Gamma \setminus y_1 : \sigma \ldots y_n : \sigma$ for the result of removing any of $y_1 \ldots y_n$ that appear in Γ from Γ, otherwise leaving the ordering of variables alone. Thus we can state the rule for the universal quantifier as follows:

$$\frac{\Gamma \vdash A : t}{\Gamma \setminus y_1 : \sigma \ldots y_n : \sigma \vdash \forall_\sigma x : A[x/y_1 \ldots x/y_n] : t}$$

where $y_1 : \sigma \ldots y_n : \sigma$ are variables that are not bound in A and $x : \sigma$ doesn't appear in A. The rule for the existential quantifier is completely parallel. The rules for all the logical operations are listed in Table 10.2 . We will usually consider adding all of the rules in Table 10.2 at once, so we will use $\mathbf{ND}[XL]$ to mean the system with structural rules X and all of the logical rules.

Table 10.2 Natural deduction for logical operations

$\dfrac{\Gamma\vdash A{:}t}{\Gamma\vdash\neg A{:}t}$	**Negation**
$\dfrac{\Gamma\vdash A{:}t \qquad \Delta\vdash B{:}t}{\Gamma,\Delta\vdash(A\wedge B){:}t}$	**Conjunction**
$\dfrac{\Gamma\vdash A{:}t \qquad \Delta\vdash B{:}t}{\Gamma,\Delta\vdash(A\vee B){:}t}$	**Disjunction**
$\dfrac{\Gamma\vdash A{:}t \qquad \Delta\vdash B{:}t}{\Gamma,\Delta\vdash(A\rightarrow B){:}t}$	**Conditionals**
$\dfrac{\Gamma\vdash A{:}t}{\Gamma\backslash y_1{:}\sigma\ldots y_n{:}\sigma\vdash\forall_\sigma x{:}A[x/y_1\ldots x/y_n]{:}t}$	**Universal**
$\dfrac{\Gamma\vdash A{:}t}{\Gamma\backslash y_1{:}\sigma\ldots y_n{:}\sigma\vdash\exists_\sigma x{:}A[x/y_1\ldots x/y_n]{:}t}$	**Existential**

Exercise 10.17. *Consider the natural deduction system* $\mathbf{ND}[L]$ *consisting of the rules Constants, Application, Abstraction, Concretion and the rules for the logical constants. Which of the following sentences can be constructed in this system? Provide derivations for the ones that can.*

a. $\exists_e x.Lxx$

b. $\exists_t p.\neg p$

c. $\forall_e x.\exists_e y.(Lyx \rightarrow Hxy)$

d. $\forall_e x.Laa$

e. $\forall_{e\rightarrow t} X(Xa \rightarrow \exists_e X)$

The above exercise illustrates a deep difference between the syncategorematic approach to quantifiers and other logical words and the approach that adds special logical constants to the signature. In many general λ-languages—such as those characterized by Curry systems without the Identity rule—it is not possible for λ to bind variables appearing in predicating position. Thus our strategy, from Section 4.2, for reconstructing higher-order entities corresponding to the quantifiers fails, since $\lambda X.\Pi_\sigma y.Xy$ involves λ binding X in a predicating position. Indeed, in $\mathbf{ND}[X]$ where X contains L and any combination of the rules P, C and W, one cannot even form the open formula $\Pi_\sigma y.Xy$ since even free variables cannot appear in predicating position in these systems. It seems, then, that the syncategorematic approach allows us to take the Tractarian position on logical operations seriously, and avoid positing logical entities.

Let's end briefly by mentioning the system $\mathbf{ND}[L]$, obtained by adding the logical rules to the structural system. As with the structural system, derivations of sequents are almost unique. In this case, they are not entirely unique because of the rule $\forall$ and $\exists$, in which a bound variable x can replace variables $y_1 \ldots y_n$, thus permanently losing information about how a given sequent was derived (i.e. which variables were there before x replaced them). However, derivations are unique up to a relabelling of variables: $\Delta \vdash M : \sigma$ and $\Delta' \vdash M' : \sigma$ are equivalent up to relabeling of variables provided there is some permutation of variables such that applying it to all the variables (free and bound) in Δ and M yield Δ' and M' respectively.

Proposition 10.3. *If* $\Gamma \vdash P : \sigma$ *is a derivable sequent in* $\mathbf{ND}[L]$ *then it has a unique derivation up to a relabeling of the variables (free and bound) appearing the premise and conclusion of the sequents appearing in that derivation.*

Proof. This is proved in essentially the same way as Proposition 10.2. As before it is not possible to derive a typing statement of the form $\Gamma \vdash x : \sigma$ where x is a variable. We can then prove by induction on n that every term that has a derivation of length $\leq n$ has a unique derivation of length $\leq n$. As before derivations of length one are instances of Constants, and so are uniquely determined by the constant.

Suppose suppose that the IH holds for derivations of length ≤ 1, and that $\Gamma \vdash P : \sigma$ is derivable in $n+1$ steps. Since we have ruled out the case where P is a variable, P must either be a constant, a λ-term (have the form $\lambda x.M$), an application term (have the form (MN)), or a logical term (have the form $(A \wedge B)$, $\neg A$, $\forall x.A$ etc). All cases except the last are proved as in Proposition 10.2. If the term is of the form $\neg A$, $(A \wedge B)$, $(A \vee B)$ or $(A \to B)$ then it could only have been made by the rules $\neg$, $\wedge$, $\vee$ or $\to$ and the derivation is unique up to relabeling of variables by the same reasoning as for the rule Application. If the sequent is of the form $\Gamma \vdash \forall_\sigma x.A$ then it must have been formed by the rule $\forall$ from some sequent $\Delta \vdash B$, where $A = B[x/y_1 \ldots x/y_n]$. This sequent is unique modulo the variables $y_1 \ldots y_n$. For suppose it could also be derived from $\Delta' \vdash B'$ using the rule $\forall$, where $A = B'[x/y'_1 \ldots x/y'_k]$. By Proposition 10.2 it follows that $FV_s(\forall_\sigma x.A) = var_s(\Gamma)$ and $FV_s(B) = var_s(\Delta)$. Since $\forall x.A$ is the result of replacing the variables $y_1 \ldots y_n$ in B with x and then binding x with the universal quantifier, it follows that Δ must be the result of replacing the occurrences of the variable x in $FV_s(A)$ with the variables $y_1 \ldots y_n$, and Δ' the result of doing the same with the variables $y'_1 \ldots y'_k$, and so $n = k$ and Δ and Δ' are the same except for the variables, and similarly B and B' are the same. □

Endnotes

1. In, e.g., the former we have subterms YZ and ZY, and it is impossible assign types to Z and Y that make both well-formed.
2. Note that some authors are less than careful about this issue, either the constraint on duplicate variables appearing in type assignments is omitted, or Contraction is conflated with Psuedo Contraction, or both.
3. The reader can consult Hindley (1997), Chapter 3 for a much more detailed discussion of these results. Hindley also introduces the notion of a principal derivation for a natural deduction system akin to the simplified type system presented below and shows that every typable term M can be associated with a unique principal derivation. This fact somewhat mitigates the non-uniqueness of the Church-terms associated with typable untyped λ-terms mentioned above.
4. See, e.g., Mitchell (1996), or Hindley and Seldin (2008).
5. Unlike in Section 7.6 $\sim_\sigma$ on its own, without terms flanking it, is not an abbreviation for a λ-term, since these terms do not belong to the substructurally defined λ-languages without Identity.
6. In Section 7.6 we defined the notion of 'being extensional' in the object language, but here the quantifiers we used in that definition quantified into predicating position and so are not available the restricted λ-languages in question.

Hints for exercises

[i] **Hint:** This time you will need to also appeal to the rule Permutation.
[ii] **Hint:** You will have to use Permutation in this derivation.
[iii] **Hint:** You will need Constants, Permutation and Contraction in this derivation.

11

Application

Structure I

In the next three chapters, we will apply the tools we have developed in the preceding chapters to more fine-grained grained pictures of reality. These chapters have been divided up according to themes, but they all concern a closely related cluster of questions and should be read together. This chapter concerns the formalization of different sorts of 'structured' theories of propositions. We will restrict attention to languages that make use only of relational types (we consider only terms whose types end with a t).

11.1 Quasi-syntactic accounts of structure

According to a popular version of the structured theory of reality, propositions are structured in a way that is analogous to the way that sentences of a typed language are structured, properties and relations as predicates are structured, and so on. Thus, just as a sentence is composed of constants of different types—predicates, names, operator expressions, and so on—put together in accordance with the syncategorematic rules of the language, a proposition is composed of simple entities of different types—properties, individuals, operators and so on—put together according to the legitimate ways of combining entities (whatever they may be).

There are many versions of the syntactic theory of structured propositions. An embarrassment of riches, even, since for each conceivable language, one could articulate a theory of reality according to which it is structured as that language is. So what sort of language properly reflects the structure of reality? Extant theories have focused on artificial languages that are modeled on, or at least closely resemble human natural language.[1] Note, however, that there are many features of human language that seem to be forced upon us by our physical limitations as humans. This is a surprising feature of the syntactic theories of reality that posit structure in reality that reflects these features. For instance, human languages evolved from spoken languages in which words must be spoken one at a time and in temporal order. This order continued in written language, with the temporal order represented by direction on the page (right-to-left, left-to-right, up-down, and so on depending on the language). It follows that, in any given language, each word has an ordinal position, and we can ask which word comes first, second, third, and so on, and we can make comparisons like these across different sentences, asking whether 'John' occupies the same position in 'John loves Jim' as 'Jack' does in 'Jack marries Jill'. So one striking feature of the flat-footed syntactic approach to the structure of reality is that it makes the posit that each constituent of a proposition has a unique ordinal position in that proposition. Once you have accepted

DOI: 10.4324/9781003039181-15

that all the constituents of a proposition can be linearly ordered some seemingly awkward choices must be made. For instance, there are different versions of the structured theory of reality depending on whether it is modeled on a language that uses the infix notation, the most familiar notation tracing back to Peano (1889), where we write binary operations between their arguments, or prefix notation where the operation goes before the arguments, favoured by the Polish logicians, or postfix notation where the operation goes after the arguments, favoured by the French mathematicians Bourbarki. Each gives a different answer to the question of where conjunction appears in a conjunctive proposition *p and q*: middle, beginning, or end? Perhaps the thing to say is that there are really three distinct primitive notions of conjunction: pre-conjunction, in-conjunction and post-conjunction, each metaphysically simple but each appearing in a different position relative to these arguments. But these notations do not exhaust the options: the original version of higher-order logic formulated by Frege was presented in a completely unfamiliar two-dimensional notation in which it is hard to classify the primitive material conditional connective as between, before or after its arguments. Another feature of natural language is that its syntax tends to have a binary branching structure, whereas artificial languages like the propositional calculus can have ternary branching structure and other sorts of syntactic structure. A possible explanation for the binary branching nature of natural language is that it has something to do with human psychology, in which case it would again be surprising if reality had that same structure.

Given that reality and language have structure in common, one could go further and maintain that semantic relations between language and world must preserve this structure: a sentence expresses a proposition with the exact same structure as that sentence. This is a substantive posit. For instance, in English we appear to have syntactically simple predicates, like 'being a vixen' or 'being a bachelor' that would plausibly stand for complex properties on the structural view—*being a female fox* or *being an unmarried man* in this case. The alternative view must maintain that it is impossible to introduce a simple constant that denotes a complex entity—when we try to use a predicate like 'is a vixen' to pick out a complex property, we pick out a metaphysically simple property that is merely necessarily coextensive with the property of being a female fox. and more generally, introducing simple terms by definition allows us to refer to new simple entities. (Note that such views must hold that there are enough simple properties to allow us to always be able to do this: one could attempt to formalize this in higher-order logic with a special predicate for being simple by a comprehension principle stating that every proposition, property or relation is coextensive with a simple proposition, property or relation.) In this chapter, however, we will make room for the possibility of a simple expression referring to a complex property. In this case, it is often helpful to have a name for the sort of language where this never happens: a *logically perfect language*. This is how Bertrand Russell introduces the notion:

> 'In a logically perfect language the words in a proposition would correspond one by one with the components of the corresponding fact, with the exception of such words as "or", "not", "if", "then", which have a different function. In a logically perfect language, there will be one word and no more for every simple object, and everything that is not simple will be expressed by a combination of words, by a combination derived, of course, from the words for the simple things that enter in, one word for each simple component. Russell (1918), p. 25.

We introduce this with an informal definition.

Definition 11.1 (Logically perfect language)**.** *A language is* logically perfect *when there is exactly one word for each simple entity, and every entity that is not simple can be expressed by a combination of those words.*

The idea of a logically perfect language is useful because a logically perfect language reflects the structure of reality. Note, however, that our 'definition' involved quantification over entities across infinitely many different types at once, and so cannot be taken literally and stated in higher-order logic. In fact we must take the notion of a logically perfect language as primitive, and rather than attempt to define it in terms of semantic words like reference we must simply lay out principles that connect the notion of a logically perfect language to notions like reference and metaphysical simplicity. We will state axioms and rules that are sound only in logically perfect languages in Section 13.2: these principles can be thought of as characterizing the notion of a logically perfect language internally.

Remark 11.1. Observe that a logically perfect language can use, say, infix notation even if reality is in, say, postfix notation. One could define a stronger notion of a language perfectly reflecting the structure of reality that builds things like this in, but our notion is more useful and can be applied in contexts where reality is not syntactically structured.

In a logically perfect language, the simple constituents of a proposition correspond exactly to the constants appearing in it. Note, however, that many formal languages—including our λ-languages—employ bound variables in order to avoid certain structural ambiguities. Do bound variables, like constants, correspond to constituents of a proposition? Should the structuralist posit distinct propositions corresponding to distinct but α-equivalent expressions like $\forall_e x.Lxx$ and $\forall_e y.Lyy$, or $\lambda x.Fx$ and $\lambda y.Fy$? A negative answer is implied by the more general idea that bound variables, unlike constants, do not contribute constituents to the propositions they correspond to. Note that the λ-languages we have considered in this book contain closed expressions that are formed entirely out of bound variables and λ, and so would be completely constituentless according to this general principle. We will take this idea seriously at various points, but note that it requires rejecting the idea that every proposition is either simple or can be decomposed into simpler constituents.

Let us now attempt to formulate a higher-order theory that captures the informal picture of reality as structured like a language. All the languages we have considered in this book have sentences with subject-predicate structure corresponding to English sentences in subject-predicate form like 'Socrates is wise': sentences of the form Fa where $F : e \to t$ and $a : e$. Every sentence in subject-predicate form has a unique subject and a unique predicate. If the sentences 'a *F*s' and 'b *G*s' are the very same sentence of English then 'a' and 'b' are the same names and the verbs '*F*' and '*G*' are the same. This holds true of sentences of our formal languages as well. Note this idea that a subject-predicate sentence must have a unique predicate and argument applies no matter how we notate predication—whether we use infix, prefix or postfix notation for example—and so seems to be a feature of all the quasi-syntactic views we have discussed above. This uniqueness applies to other sentences that have predicates and arguments of higher types: for instance, an operator-sentence sentence has a unique operator that is doing the predicating and a unique sentence that is the argument of that predication. Thus we could articulate this aspect of the syntactic theory of reality by the following schema of higher-order logic, with an instance for each choice of relational types σ and τ:

Predicate Argument Structure $\forall_{\sigma\to\tau} XY \forall_\sigma zw(Xz =_\tau Yw \to X =_{\sigma\to\tau} Y \wedge z =_\tau w)$

Predicate Argument Structure, however, suffers from a couple of serious problems which I think rule out the straightforward syntactic picture as a serious contender as a theory of the structure of reality.

According to β, the proposition that *Mary loves Mary*, *Laa*, can be decomposed into subject-predicate form in multiple ways: it can be thought of as the property of *loving oneself* applied to *Mary* (i.e. $(\lambda x.Lxx)a$) or the property of *loving Mary* applied to *Mary* (i.e. $(\lambda x.Lxa)a$). But these are not the same property—indeed they are not even coextensive properties, since one can love oneself without loving Mary.

Exercise 11.1. *In this question, you will work in the full λ-language $\mathcal{L}(\Sigma)$. You may assume the signature contains as many constants in each type as you like.*

a. Assuming β, find two more ways of decomposing Laa into a property applied to an argument.

b. Assuming β, find a counterexample to Predicate Argument Structure where a single proposition is decomposed into subject-predicate form in two different ways where the arguments and predicates are both different.

c. Assuming β, find a counterexample to Predicate Argument Structure where a single proposition is decomposed into predicate subject form in two different ways where the arguments are different but the predicates the same.

What should we make of these counterexamples? How should we discuss failures of principles like β and η? Do we settle questions about their validity by consulting our intuitions about λ, as we would if we were, say, to settle a question about the logic of knowledge, or parthood? There is an important disanalogy here, for in the latter case we had a pretheoretic grasp of the concepts of knowledge and parthood which we could consult, and weigh against other theoretical benefits and costs of taking a certain principle to be true or not. By contrast, λ is a completely technical device, introduced by Church to do a certain technical job (indeed, a job that has been achieved by several other technical devices, such as combinators; see Section 3.5). The only handle we have on its meaning are the principles he laid down for it when it was introduced in Church (1940): α, β and η.[2] Luckily those principles are sufficient to pin down the meaning of λ uniquely. Theorem 3 showed that if some other device λ' satisfied β, in the sense that $(\lambda' x.M)N$ was synonymous with $M[N/x]$ when N is free for x in M, and λ satisfies η in the same sense then $\lambda x.M$ and $\lambda' x.M$ are synonymous for any term M (recall that this is synonymy in the strong sense that they can be substituted in any context *salve veritate*). Thus β and η jointly pin down the meaning of λ-terms.

Simply rejecting β without saying more would not do, for without β we have nothing to go on. For all we've said the λ-term we previously used to pick out the property of *being old and wise*, $\lambda x.(Wx \wedge Ox)$ could pick out the property of *being tall*. β, of course, rules this out because, for instance, applying the former property to *Socrates* yields the proposition that *Socrates is wise and old*, and not the proposition that *Socrates is tall*. Some philosophers wishing to use the framework of higher-order logic without taking sides on this anti-structured commitment of the standard λ-calculus have opted to weaken the claim that $(\lambda x.M)N$ was synonymous with $M[N/x]$ to the claim that they are merely coextensive. Assuming that $(\lambda x.M)N$ and $M[N/x]$ have type $\bar{\sigma} \to t$, and $\bar{y} : \bar{\sigma}$ (since we are working with the relational types) we can capture this as follows:

Extensional β $\forall y_1 \ldots y_n((\lambda x_1 \ldots x_m.M)N_1 \ldots N_m y_1 \ldots y_n \leftrightarrow (M[N_1/x_1 \ldots N_m/x_m])y_1 \ldots y_n)$

Observe that we can state a similar extensional version of η, but it is already implied by Extensional β. Thus for instance $\lambda x.(Wx \wedge Ox)$ must apply to all and only those individuals

that are both wise and old. Unfortunately this even now does little to pin down the meaning of the λ-terms, for the present λ-term could still denote the property *being tall* provided that all and only tall people are old and wise.

So there is a challenge for those employing λ-terms while also rejecting β and η to tell us more about what those expressions mean, for instance, by laying out some principles that tell us more about what they do or by relating them to concepts we already understand or have some theoretical role for.[3] Luckily, rejecting λ-notation is not the only alternative: in Chapters 9 and 10 we have explored general λ-calculi that do not contain all the problematic λ-terms that allow for multiple decompositions of structured propositions, while letting us maintain β and η. These will play an important role in the next section.

Remark 11.2. One might try to link Church's λ with expressions in mathematical English about which we do have some sort of pretheoretic grip. For instance $\lambda x.(Wx \wedge Ox)$ is sometimes paraphrased as 'is an x such that x is wise and x is old'. This strategy has limited applicability given that it is only available for λ-terms $\lambda x.A$ where x is a variable of type e and A type t. These paraphrases also have to be taken with a liberal pinch of salt, since the first occurrence of x is taking the position of a noun, and the second the position of a name. Perhaps it would be better to say 'is an x such that that x is wise and that x is old', since that is grammatical in the same way that 'is a philosopher such that that philosopher is wise and that philosopher is old' is grammatical. But these renditions are still puzzling, for they seem to require us to think that, in addition to philosophers, cats, dogs and so on, there are xs and ys, which can only be made sense of in ways that make the proposed paraphrase clearly inadequate (e.g. as tokens of the letters 'x' and 'y'). One might also worry that paraphrases involving the phrases 'an x such that' are also implicitly quantificational in nature when the thing being paraphrased is not quantificational.

However, there is a more serious problem for the structured theorist who wishes to keep all the standard λ-terms in their language, and the corresponding higher-order existential commitments that come with having such terms. For whatever replacement for β and η this theorist comes up with, it had better, at bare minimum, imply Extensional β if they are to bear any resemblance to Church's λ. Call the result of replacing the principles β and η in H with Extensional β H_0. Unfortunately, Predicate Argument Structure is inconsistent in H_0. This is due to the Russell-Myhill paradox, initially published in appendix B of Russell (1903), and rediscovered by Myhill in the context of Church's theory of sense and denotation in Myhill (1958).

Theorem 11.1 (The Russell-Myhill Theorem)**.** *Predicate Argument Structure is inconsistent in* H_0.

Proof. Consider the Russell-Myhill term:

$$M := \lambda p.\forall_{t\to t} X(p =_t \forall_t X \to \neg Xp)$$

First we prove $\neg M(\forall_t M)$. Suppose for contradiction that $M(\forall_t M)$. Expanding out the definition of the first M and applying Extensional β yields $\forall_{t\to t} X(\forall_t M =_t \forall_t X \to \neg X(\forall_t M))$. Instantiating X with $\forall_t M$, we get $\forall_t M =_t \forall_t M \to \neg M(\forall_t M)$, and thus $\neg M(\forall_t M)$ contradicting the assumption.

Thus $\neg M(\forall_t M)$. To obtain a contradiction we now prove $M(\forall_t M)$, or in other words, $\forall_{t\to t} X(\forall_t M =_t \forall_t X \to \neg X(\forall_t M))$. Let X be an arbitrary operator such that $\forall_t M =_t \forall_t X$.

An instance of Predicate Argument Structure lets us infer that $X = M$, and since we have shown that $\neg M(\forall_t M)$ it follows that $\neg X(\forall_t M)$. □

In fact, we can adapt the above argument to show that two seemingly weaker principles are also inconsistent:

Predicate Structure $\forall_{\sigma\to\tau} XY(Xz =_\tau Yz \to X =_{\sigma\to\tau} Y)$

Argument Structure $\forall_\sigma zw(Xz =_\tau Xw \to z =_\sigma w)$

Predicate Structure tells us that if two subject-predicate propositions with the same argument are the same, then the predicates are the same. Argument Structure tells us that if two subject-predicate propositions with the same predicate are the same, then the arguments are the same.

Comprehension Check 11.1. *Check that the argument in Theorem 11.1 only appeals to Argument Structure and thus serves to reduce that principle to inconsistency in* H_0.

Exercise 11.2. *Show that Predicate Structure is inconsistent in* H_0 *by considering the operator*

$$M := \lambda p.\forall_{t\to t} X(p =_t Xr \to \neg Xp)$$

where r is some arbitrary closed sentence of type t.

What should we make of this paradox? There are familiar responses to structurally similar paradoxes—such as the liar paradox or Russell's paradox—that involve weakening the logic somehow. Often this takes the form of weakening the propositional logic in some way, and one can indeed construct models of Predicate Argument Structure in the sorts of non-classical logics that also support naïve reasoning about truth and sets (see, for instance, Kripke (1975) and Field (2008)).[4] Another possibility, that is distinctive to paradoxes that involve higher-order quantification, is to revise the quantificational logic—specifically the logic of the higher-order quantifiers. The most common diagnosis of the Russell-Myhill paradox, going back to Russell himself, is that the operator quantifier, $\forall_{t\to t} X$, that appears in the problematic Russell-Myhill operator, M, is impredicative: it quantifies over a domain that includes an operator constructed from that very quantifier. The standard Russellian response, then, is that no quantifier can quantify over a domain that includes entities constructed from that very quantifier. But while it is impossible to employ a quantifier that quantifies unrestrictedly over all operators, nothing stops us from employing restricted operator quantifiers that range only over operators that involve operator quantifiers with a more restricted domain. The logic of restricted quantifiers does not include the principle of universal instantiation, $\forall_\sigma F \to Fa$. This can be illustrated using the first-order restricted quantifier 'every cat': from 'every cat likes cat food' it doesn't follow that 'Mary likes cat food'. There are various ways to implement this idea. Russell's implementation, 'ramified type theory', was quite extreme and involved complicating the syntax of higher-order logic so that (putting it roughly) every type σ comes in a hierarchy of different levels indexed by the natural numbers, $\sigma^1, \sigma^2, \ldots$: not only do quantifiers of a given level only range over entities of lower levels, but one can't even apply something of a functional type to something unless it is of a lower level. Thus, for instance one cannot apply something of a functional type to the same argument twice, since applying an operator once raises the level of the argument. A less restrictive option is to keep the standard syntax of higher-order logic but posit a hierarchy of quantifiers at each type, $\forall^1_\sigma, \forall^2_\sigma, \ldots$ where $\forall^n_\sigma$ ranges over entities of type σ that do not have any quantifiers $\forall^k$ with $k \geq n$ as constituents. However there is a

long-standing challenge to views that weaken the logic in either of these ways, and that is to show that any kind of substantive theorizing can be sustained in these logics. We have seen in previous chapters that substantial metaphysical theorizing can be carried out in the standard framework of higher-order logic, and in Russell's time logicians were doing substantial mathematics in higher-order systems. It is far from obvious that any systematic reasoning is possible once propositional logic has been weakened enough to become consistent with Predicate Argument Structure, or in Russell's ramified version of higher-order logic.

On the other hand, we have seen that Predicate Argument Structure is a commitment of only one theory concerning the structure of reality—the quasi-syntactic theory. We have already considered some problems and artificial features of this model when taken as a theory of the structure of reality (as opposed to the structure of representations). So perhaps we should be looking for different models of the structure of reality: ones in which propositions, properties and relations can still be decomposed into their immediate constituents until we reach simple entities with no immediate constituents, but without necessarily subscribing to the syntactic vision encoded in Predicate Argument Structure.

Remark 11.3. Some philosophers who articulate theories of syntactically structured propositions, such as Soames (1987) and King (2007), might be better understood as providing a structural account of representational objects than giving us a theory of the structure of reality directly. For these authors structured propositions are intrinsically representational unlike sentences of a language, which represent in virtue of the way they are used by a linguistic community. But, nonetheless, they represent the world rather than being directly part of the world, and so it is natural to treat them as using the word 'proposition' differently than we are. For a start, in this book, we have adopted talk of 'propositions' and first-order quantification over them as an informal policy for indicating higher-order claims involving quantification into sentence position. When we use the first-order quantifiers we do not quantify exclusively over representations of individuals, we quantify over the individuals themselves. So quantification into sentence position should not be glossed in terms of quantification over representations either. Nonetheless, one can formulate versions of the syntactic theory of structured propositions understood explicitly as a theory of the structure of reality, and these are a very natural place to start our discussion.

11.2 Pictorial accounts of structure

In contrast to the quasi-syntactic theories of structured entities, there are what I'll call broadly 'pictorial' accounts of structure. We will consider several different structural views falling under this category in this chapter and the next chapter: I have grouped them together loosely under the label 'pictorial' because the account of propositional structure they posit are generally more faithfully represented diagrammatically than by expressions in some language. A metaphysically simple relation, for instance, would be represented by a certain diagram with a number of 'holes', or 'slots', equal to its arity. A complex relation would then be a diagram some of whose holes have been filled with diagrams representing other entities. A diagram all of whose holes have been filled would thus represent a 0-ary relation: a proposition. The picture of a relation as a proposition with holes can be traced back to Frege, and is arguably just as pervasive in the metaphysics literature as the syntactic picture.[5]

Pictorial views motivate a very different set of structural principles to the quasi-syntactic picture. Consider, for instance the $\beta\eta$-equivalent terms $(\lambda x.Rxb)a$ and $(\lambda x.Rax)b$. We may

think of both of these terms as two different instructions for building the structured proposition Rab, corresponding to the two different orders in which we can plug the individuals a and b into the two holes in R:

1. $(\lambda x.Rxb)a$: Start by plugging b into the second slot in R, then plug a into the first (and remaining) slot of R.
2. $(\lambda x.Rax)b$: Start by plugging a into the first slot in R, then plug b into the second (and remaining) slot of R.

Another such example would be the $\beta\eta$-equivalent terms $(\lambda q.\Box(\neg q))P$ and $\Box(\neg P)$. Both $\Box$ and $\neg$ are operators, and thus have a single proposition-shaped hole. These terms represent two instructions for building the same structured entity:

1. $(\lambda q.\Box(\neg q))P$: Start by plugging the $\neg$ into $\Box$s single hole. Since $\neg$ has a hole, the result of doing this (i.e. $\lambda q.\Box(\neg q)$) still has a hole into which P may be plugged to make a complete proposition.
2. $\Box(\neg P)$: Start by plugging P into $\neg$s only hole to make a complete proposition $\neg P$. Then plug that into $\Box$s only hole to make another complete proposition.

Unlike our first example, in which we only glued entities together using application, the first instruction above requires us to glue $\Box$ and $\neg$ together in a completely different way—composition—captured by the fact that the diagram for negation being plugged into the hole of $\Box$ has a hole of its own, that contributes holes to the result of the plugging.

Thus both of these examples accord with our principle $\beta\eta$. Indeed, we will see in Section 11.4 that $\beta\eta$ is in a certain sense sound and complete for this sort of picture thinking: two λ-terms provide instructions for building the same diagram iff they are $\beta\eta$-equivalent. Thus $\beta\eta$-equivalence is the proper notion of sameness for a structured metaphysics on which relations are propositions with holes.

Another difference between the pictorial view and the quasi-syntactic view is that Predicate Argument Structure is quite obviously unsound according to the former view. Both of the examples discussed above illustrate this. The predicate of $(\lambda x.Rxb)a$ is $\lambda x.Rxb$ and the argument is a, whereas the predicate of $(\lambda x.Rax)b$ is $\lambda x.Rax$ and the argument b. Similarly, the predicate of $(\lambda q.\Box(\neg q))P$ is $\lambda q.\Box(\neg q)$ and the argument P, whereas the predicate of $\Box(\neg P)$ is $\Box$, and the argument $\neg P$. Predicate Argument Structure would imply false identities like $P =_t \neg P$ and $\lambda x.Rxb =_{e\to t} \lambda x.Rax$. It is quite clear, too, that the structure of reality is not reflected by the structure of terms in a λ-language.[6] As previously mentioned, we should rather be thinking of the λ-terms as instructions for building an entity, with the possibility of different instructions yielding the same end result. As a result of this mismatch between language and reality, the logical principles we will introduce in order to capture these views will invariably appear more complicated on the page than Predicate Argument Structure, when notated in traditional logical notation such as the λ-calculus. This is hardly surprising: the quasi-syntactic picture looks simple when stated in a logical notation that reflects the sort of structure it claims reality has. The quasi-syntactic picture would presumably look more complicated if we were using a pictorial notation instead of a logical one.

Above we have talked about the holes in a relation R as though there is a first hole, second hole, third hole, and so on, and that the immediate constituent occurrences in a proposition are subsequently ordered by which hole they appear in. This will be reflected in our diagrams, where holes and constituents will appear to the left or right of other holes or constituents. According to this picture, one can ask meaningfully whether the constituent a appears in the same position relative to b in a proposition p as c does to b in q, even when p and q are completely different propositions. This view is often attributed to Russell and appears to be the dominant view about structured relations in metaphysics.[7] (Note that like the syntactic picture the arguments of R in a relational proposition like Rab have an order relative to each other, but unlike the syntactic picture the relation need not: the relation simply has two argument places that are filled by its arguments—the relation itself does not appear before or after the arguments.)

In an important paper, Kit Fine (Fine, 2000) has challenged the orthodoxy, and has described an alternative view—'positionalism'—in which holes and constituents have no objective non-arbitrary ordering. A binary relation, such as loves, will have two holes—one for the lover, the other for the beloved—and individuals may be inserted into either hole, but there is no non-arbitrary sense in which one hole comes before the other, and thus no sense in which an individual occurs *before* another in a proposition obtained by filling those holes in one way or the other. It is possible to interpret our diagrams in a way that is consistent with this metaphysical picture by assuming that the left-to-right ordering of holes in a given pictorial representation was simply the result of an arbitrary choice in the depiction. We will, however, begin with the standard Russellian view that accepts a notion of order; the positionalist view will be treated in Section 12.4. (Observe that even on the Russellian ordered view there is an arbitrary choice when we represent pictorially, for we have to choose to either associate the propositional constituent ordering with left-to-right ordering on the page, or the right-to-left ordering.)

Fine himself does not accept positionalism but rather a related view that accepts the same structural features as it but does away with holes by invoking other primitives. See the following remark.

Remark 11.4. For expository purposes, we have talked of relations possessing 'holes' as though they were first-rate parts of a relation alongside its constituents. Should we take this way of talking seriously, or is it a façon de parler? There is an analogous question concerning the metaphysics of material holes. We talk about holes in physical objects as though they were first-rate entities. Some philosophers treat them as such, whereas others attempt to eliminate them by reducing hole talk to something more ontologically respectable.[8]

An analogous pair of attitudes may be taken with the present talk of holes.[9] Everyone ought to acknowledge that the binary relation *loves* stands to the unary properties . . . *loves Jack*, . . . *loves Jill*, etc. differently to the way it stands to the unary properties *Jack loves* . . . , *Jill loves* . . . , etc: members of these classes are completions of *loves* in a comannered way. Thus one would expect an eliminative account of our talk of holes to be available which replaces sentences mentioning holes with sentences that only talk about the comannered completions of relations. For every structural view we describe in terms of holes, there is a corresponding position that doesn't posit holes but has the resources to make sense of such talk. (See Litland (forthcoming).)

11.3 Relational diagrams

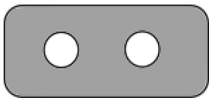
In the following sections, we'll discuss a number of pictorial structured views, that can be roughly categorized by which ways of combining entities to make new entities are allowed. We'll begin with the simplest sort of structured view. According to this view the only way to make a new entity is, roughly, by inserting an entity into one of the holes of another entity, with the resulting entity inheriting any holes in the inserted entity. We will see that this operation has application and composition as two limiting cases, so we shall call this operation *complication*.

The theory will be described first by way of *relational diagrams*, which provide a more faithful representation than is possible in a linear logical notation. We will later see how to develop a more conventional logical notation for them. Let's begin by describing the relational diagrams that represent metaphysically simple entities that do not have any proper constituents. The diagram for a simple proposition is simply a grey box:

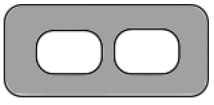
and the diagram for a simple individual is a grey circle.

This provides the two base cases (e and t) for an inductive definition of the diagram for a simple entity with any relational type. We represent holes by a white shape in the diagram for a simple proposition. Say that a 'hole of shape σ' is the result of taking a diagram for a simple entity of type σ and switching grey and white. The diagram for a relation of type $\sigma_1 \to \cdots \to \sigma_n \to t$ consists of a grey rectangle with holes of shapes $\sigma_1 \ldots \sigma_n$, in that order, labeled by a name for that relation in the outermost connected region of grey.

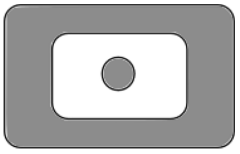
Thus a hole of shape e is simply a white circle, so that the diagram for a simple relation of type $e \to e \to t$ is a grey rectangle with two white circular holes.



By contrast, the diagram for a simple binary connective of type $t \to t \to t$ is a grey rectangle with two rectangular holes.



The diagram for the first-order universal quantifier, which has type $(e \to t) \to t$ is a grey rectangle with an $e \to t$ shaped hole, and thus looks like this:



We will say that a hole of shape σ is a hole *of* a relational diagram (or the diagram *has* that hole) if it appears as a part of the diagram, but is not itself part of another hole that is part

of the diagram. Consider a hole of shape $(e \to t) \to t$, which may be obtained by inverting the shadings in the above diagram. A diagram containing a single hole of this shape (i.e. a simple diagram of type $((e \to t) \to t) \to t$) will also contain a hole of shape e (the inner white circle). However, the white circle is not a hole *of* that diagram since it is a part of a larger hole. This is because $((e \to t) \to t) \to t$ is the type of a *unary* property, and therefor should have only one hole of shape $(e \to t) \to t$—the inner e shaped hole is not counted.

Exercise 11.3. *Draw diagrams for simple entities with the following types, and say how many holes they have:*

a. $e \to t \to t$,

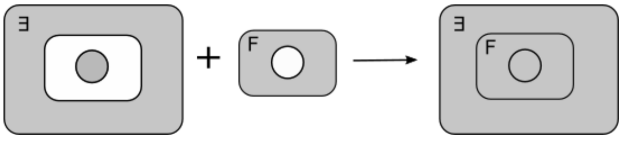
b. $e \to (e \to t) \to t$,

c. $((e \to t) \to t) \to t$

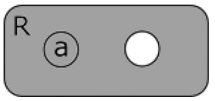
We have now described the geometric properties of diagrams for simple entities. Since we will need to distinguish diagrams for distinct simple entities, we will tag simple diagrams using labels taken from some signature Σ. The label for a relational diagram of type σ must be taken from Σ^σ.

Convention 11.1. *We will use bolded letters like* $\mathbf{R}$ *and* $\mathbf{a}$ *to denote diagrams. If a simple diagram is labeled by a constant c, we will use* $\mathbf{c}$ *to denote the diagram. Conversely, if we use* $\mathbf{c}$ *to denote a simple diagram, we assume it is tagged by the constant c.*

Next we describe how to put diagrams for simple entities together to create diagrams representing structured, or complex entities. The simplest way to put two things together is by applying one thing to another. This corresponds to slotting the diagram for an entity of type σ into the leftmost hole of another diagram. Application is only possible if the hole is of shape σ. Thus, for instance, if we wanted to plug a diagram for a property $F : e \to t$ into the existential quantifier $\exists_e : (e \to t) \to t$ we would depict this as follows:



Similarly inserting an individual diagram for $a : e$ into a relational diagram for $R : e \to e \to t$ we would get



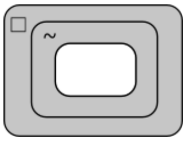
Observe that a diagram of type σ will always fit perfectly into a hole of shape σ, due to the way we have defined them.

Exercise 11.4. *Suppose that* $a : e$, $R : e \to e \to t$, *and* $\exists_e : (e \to t) \to t$. *Draw the diagram for the proposition* $\exists_e(Ra)$. *(Note: in this example, you will have to slot a complex entity into a slot.)*

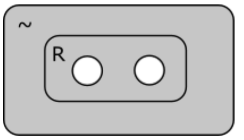
Of course, there is nothing special about the leftmost hole. The obvious generalization of application allows one to insert a diagram of type σ into any hole of shape σ in another relational diagram. If $\mathbf{R}$ is a diagram of type $\sigma_0 \to \cdots \to \sigma_n \to t$, and $\mathbf{a}$ is a diagram of type σ_k where $0 \leq k \leq n$, we will write $(\mathbf{Ra})^0_k$ for the result of slotting $\mathbf{a}$ into $\mathbf{R}$s kth hole. Thus ordinary application corresponds to the mode of combination $(\mathbf{Ra})^0_0$.

Recall that in Chapter 9 we used a similar notation to abbreviate certain λ-terms. There $(Ra)^0_m$ stood for the term $\lambda x_0 \ldots x_m.Rx_0 \ldots x_m a$, which similarly stands for plugging a into the $m+1$th slot of R.

Next we turn to composition. Like application, you compose diagrams by filling a hole of one diagram with another diagram. Application completely fills a hole, and thus decreases the arity of a diagram, whereas composition can increase or leave the arity of a diagram the same. This can happen when the diagram being inserted into the larger diagram has further holes that contribute to the end result. Here is an example: the result of composing $\Box$ with $\neg$ may be described as follows. Both $\Box$ and $\neg$ contain a proposition-shaped hole, so the result of inserting $\neg$ into the only hole of $\Box$ yields a diagram with a proposition-shaped hole:



We can notate this $(\Box\neg)^1_0$. The subscript 0 again indicates that we are inserting in the leftmost (and only) hole, the superscript 1 indicates that we are inserting something that will leave 1 hole behind. If I were to instead insert a diagram involving two holes, such as the diagram for $\mathbf{R} : e \to e \to t$, into $\neg$, we get a diagram with two holes as a result:

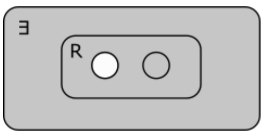


We notate this as $(\Box\mathbf{R})^2_0$. If $\mathbf{S}$ is a ternary relation, then we can write $(\Box\mathbf{S})^3_0$ and so on.

Of course, everything we can do in the leftmost hole we can do in other holes. For instance, we can insert R into the first or second slot of $\wedge$: we notate this $(\wedge R)^2_0$ and $(\wedge R)^2_1$. This is the entirely general way of combining two entities: given diagrams $\mathbf{M}$ and $\mathbf{N}$, there is another diagram $(\mathbf{MN})^n_m$. Since this operation generalizes both application and composition we call it *complication*.

Remark 11.5. Note that since we are counting holes starting with 1, not 0—we say the 'first hole', not 'zeroth hole'—$(\mathbf{MN})^n_m$ corresponds to plugging something into the $m+1$th hole of $\mathbf{M}$, not the mth hole.

We need to make one more remark on complication. Sometimes we wish to complose two entities by plugging a $k+n$ ary relation into a higher-order hole for a k-ary relation, leaving n holes behind. The hole in question will contain k grey bits that can fill k holes of the entity you are inserting. The rule we adopt (for now) is that grey fills the k rightmost holes of the entity that is being inserted (and if the rightmost holes of the entity being inserted do not match the relevant grey shapes, then the entity may not be inserted: they are not composable in this way). Here is an example involving $R : e \to e \to t$ and $\exists_e : (e \to t) \to t$:



This is one point where the order of the holes comes into play, and we will revise this rule when we look at views that reject the idea that the argument places of a relational are ordered.

In summary, the rule for complosing diagrams is

Complication If you have a diagram **M** which has holes of shape $\sigma_1, \ldots, \sigma_k, \ldots, \sigma_n$ in that order, and another diagram **N** that has holes of shape $\rho_1, \ldots, \rho_i, \ldots, \rho_j$ where $\rho_{i+1}, \ldots, \rho_j$ are the types of the holes in a relation of type σ_k, then you can plug **N** into **M**s kth hole, greying out the holes corresponding to $\rho_{i+1}, \ldots, \rho_j$ to form a relation, $(\mathbf{MN})^i_{k-1}$, with holes of type $\sigma_1, \ldots, \sigma_{k-1}, \rho_1, \ldots, \rho_i, \sigma_{k+1}, \ldots, \sigma_n$ in that order.

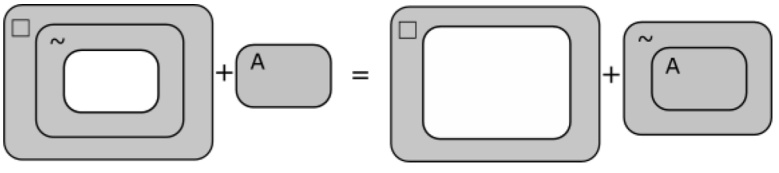
Finally, we stipulate that complication is the only way to make new diagrams from old, so that the set of relational diagrams is the smallest set containing the simple diagrams closed under complications.

Now we have an informal 'picture of reality' on the table, we can immediately note two important differences between this pictorial view and the quasi-syntactic view of structure:

1. $\beta\eta$ appears to be sound with respect to diagrammatic reasoning.
2. Predicate Argument Structure is unsound with respect to the diagrammatic reasoning.

Let's begin with the first point. We have been informally associating diagrams with λ-terms, but to state the soundness of $\beta\eta$ precisely we need to be a bit more careful about this association. We do that in the next section, but putting it crudely, λ-abstraction corresponds to poking a hole out of a relational diagram, and application corresponds to filling a hole. For present purposes, we will treat variables as though they are simple relational diagrams (i.e. the diagrams we used to depict constants). The term $(\lambda x.M)a$ roughly describes the result of poking a hole in M where x is, then filling that hole with a again. Intuitively this is the same diagram you'd get by replacing all occurrences of x in the original unpoked diagram with a, $M[a/x]$. The soundness of η can be justified similarly, except we are doing the poking and filling in the opposite order. Mx corresponds to filling a hole with x, and $\lambda x.Mx$ corresponds to poking x out again, so that we are left with the original diagram.[10]

The unsoundness of Predicate Argument Structure is also illuminating, and suggestive of how one might resist the Russell-Myhill paradox. Consider the proposition $\Box(\neg A)$. We have two distinct ways of building this proposition. We can plug the proposition A into the negation's single hole, and then plug the resulting proposition, $\neg A$, into necessity's single hole. We'd describe this using complication as follows $(\Box(\neg A)^0_0)^0_0$. Alternatively we can put $\Box$ and $\neg$ together to get the structured operator $(\Box\neg)^1_0$, obtained by plugging $\neg$'s diagram into the only hole in $\Box$'s diagram leaving a diagram with a single hole. And we can then plug the proposition A into that final hole $((\Box\neg)^1_0 P)^0_0$. Either way we get the same result:



But the principle Structure says that there's only one way to make a proposition by applying an operator to a proposition: Structure would thus imply the following pair of identities: The composite operator $(\Box\neg)$ is identical to $\Box$, and A is identical to $\neg A$.

Exercise 11.5. *Let $R : e \to e \to t$, $a, b : e$ be constants.*

a. Draw diagrams for the unary properties $(\mathbf{Ra})^0_0$ *and* $(\mathbf{Rb})^0_1$.

b. Show that Predicate argument structure (incorrectly) implies the identity of $(\mathbf{Ra})^0_0$ *and* $(\mathbf{Rb})^0_1$*, and of a and b.*

c. Come up with another diagrammatic counterexample to Predicate Argument Structure.

Exercise 11.6. *Recall that Predicate Structure is the universal closure of the following schema:*

$$Fa =_\sigma Ga \rightarrow F =_{\sigma\rightarrow\tau} G$$

In a previous exercise, we showed that Predicate Structure is also susceptible to the Russell-Myhill parodox. Find a diagrammatic counterexample to Predicate Structure.[i]

We have noticed that a single diagram can be built using complication in different ways, depending on the order in which you complicate things. As we mentioned before, rather than identifying structured entities with syntactic representations—such as the $(\mathbf{RS})^n_m$ notation, or terms of a λ-language—we can think of the syntactic representations of structured entities as providing instructions for building a structured entity. Now we have a concrete diagrammatic representation on the table, we can substantiate our previous claim that two instructions can yield the same result.

11.4 Translating between diagrams and λ-terms

We now turn to a couple of important questions. Since we are investigating the claim that the world is faithfully represented by relational diagrams, we need to say which entities are represented by each relational diagram. One way to answer this question would be to associate an (already interpreted) λ-term to each relational diagram, thereby associating each diagram with something in reality: that is, to translate relational diagrams into λ-terms.

In the other direction, we might want to find a reverse translation, taking a λ-term to a relational diagram. However, one can quickly see that not every λ-term corresponds to a diagram, meaning that not every λ-term stands for a property, relation or other entity on this view of reality. So we also need to answer the question: which λ-terms have diagrams? As a warm up to answering these questions in full generality, the reader should attempt the following exercises, employing an informal grasp of what the diagrams represent. In these exercises, the reader may assume that the first argument place of a λ-term $M : \sigma_1 \rightarrow \ldots \sigma_n \rightarrow t$ corresponds to the leftmost hole of a diagram (that is the σ_1 argument), the second argument place (σ_2) the second leftmost hole, and so on.

Exercise 11.7. *Draw three different complex relational diagrams tagged from a signature* Σ*, and find* λ*-terms from that signature that correspond to each diagram.*

Exercise 11.8. *Some of the* λ*-terms below, taken from a signature with constants* $\Box : t \rightarrow t$*,* $\wedge : t \rightarrow t \rightarrow t$*,* $R, S : e \rightarrow e \rightarrow t$*,* $a : e$ *correspond to relational diagrams. For each term say whether it can be represented by a diagram. If it can, draw the diagram, and if it cannot, briefly describe why it cannot. (Tip: make sure you have correctly drawn the diagrams for the constants before attempting to answer these questions.)*

a. $\lambda p.\Box(\Box p)$

b. $\lambda x.\Box(Rxx)$

c. $\lambda pxy.\wedge p(Rxy)$

d. $\lambda pxy.\wedge(Ryx)p$

e. $\lambda xyzw.\wedge(Rxy)(Szw)$

f. $\lambda x.Raa$

Let's turn to the first question of associating λ-terms to relational diagrams. Since every diagram can be built in at least one way from simple diagrams by complication, we can associate a λ-term diagram inductively, relative to a way in which it is built. If the diagram can be put together in several different orders, this means we will associate several different λ-terms to the diagram. Thus it suffices to associate each simple diagram with a constant, and then figure out how to express the worldly analogue of complication using λs. If I have relational diagrams $\mathbf{R}$ and $\mathbf{S}$ representing relations R and S, what relation does the diagram $(\mathbf{RS})^n_m$ represent, as a λ-term? Above we have been associating the generalizations of composition and application described with λ-terms in a piecemeal way, but there is obviously something more systematic going on here. If I have relation R of type $\sigma_1 \to \ldots \sigma_k \to \ldots \sigma_n \to t$, and another relation S that is of type $\rho_1 \to \ldots \rho_i \to \ldots \rho_j \to t$ where $\sigma_k := \rho_{i+1} \to \ldots \rho_j \to t$, then the relation denoted by the complication of diagrams for R and S of type $\sigma_1 \to \ldots \sigma_{k-1} \to \rho_1 \to \ldots \rho_i \to \sigma_{k+1} \to \ldots \sigma_n \to t$, is given by the λ-term:

$$\lambda x_1 \ldots x_{k-1} y_1 \ldots y_i x_{k+1} \ldots x_n . R x_1 \ldots x_{k-1} (S y_1 \ldots y_i) x_{k+1} \ldots x_n$$

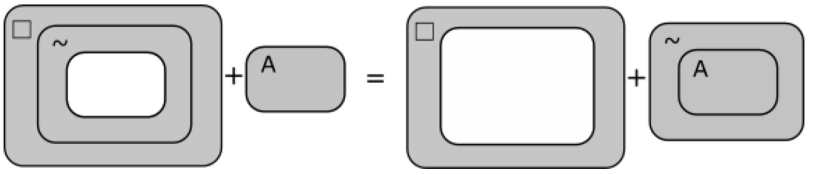
This η-reduces to

$$\lambda x_1 \ldots x_{k-1} y_1 \ldots y_i . R x_1 \ldots x_{k-1} (S y_1 \ldots y_i)$$

Recall that in Section 9.7 we introduced $(RS)^m_n$ as a shorthand for exactly this λ-term, and there also called it complication. Given the above observation this overloading of notation and terminology is very natural. Thus we have the following:

> Each simple diagram $\mathbf{c}$ of type σ may be associated with the constant $c : \sigma$ it is tagged by.
>
> If $\mathbf{M}$ and $\mathbf{N}$ are relational diagrams associated with λ-terms M and N, then the complication of diagrams $(\mathbf{MN})^n_m$ may be associated with the λ-term $(MN)^n_m$ (i.e. $\lambda \bar{x}\bar{y}.M\bar{x}(N\bar{y})$ where $\bar{x} = x_1, \ldots, x_m$ and $\bar{y} = y_1, \ldots, y_n$ are sequences of variables of the appropriate type).

Note that this association between relational diagrams and λ-terms is one-many. Our inductive association above assigned λ-terms not directly to diagrams, but to a particular way of building a diagram from simple diagrams using complication. Since a single diagram can be built in different ways a single diagram will be associated with multiple λ-terms. For instance, our earlier example:

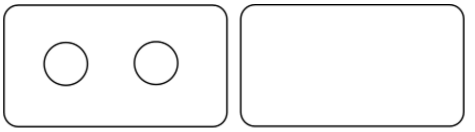


is associated with the terms $(\lambda q.\Box(\neg q))P$ and $\Box(\neg P)$ according to whether we put $\Box$ and $\neg$ together first, or $\neg$ and P together first. Note that, that terms associated with a given diagram will always be $\beta\eta$-equivalent.

Remark 11.6. There are translations that assign each relational diagram, $\mathbf{d}$, a unique λ-term $\mathbf{d}^{\lambda}$ in '$\beta\eta$-normal form' (see Section 3.4). This translation is a bit harder to describe, however, so we have not included it in the main thread. Clearly every simple relational diagram $\mathbf{c}$ can be associated uniquely with the constant it is tagged by, c, since constants are in $\beta\eta$-normal form. Every complex relational diagram $\mathbf{M}$ has a 'main relation', the outermost box, labeled by a constant R, some of whose holes have been filled with other occurrences of relational diagrams $\mathbf{S}_1, \ldots, \mathbf{S}_k$ in that order. Assume for induction that $\mathbf{S}_1, \ldots, \mathbf{S}_k$ have been assigned λ-terms $\lambda\bar{y}_1.N_1, \ldots, \lambda\bar{y}_k N_k$ in normal form, where no N_i starts with λ. Let $\bar{x}_i$ be a sequence of variables matching (in length and type) the uninterrupted stretch of unfilled holes from R in the diagram $\mathbf{M}$ immediately preceding the point where $\mathbf{S}_i$ occurs. Note that $\bar{y}_i$ is a sequence of variables matching the holes in $\mathbf{S}_i$, and each N_i is in normal form too. Then we can associate $\mathbf{M}$ with the normal form λ-term:

$$\lambda\bar{x}_1\bar{y}_1 \ldots \bar{x}_k\bar{y}_k.R\bar{x}_1(N_1)\ldots\bar{x}_k(N_k)$$

Now let us return to the reverse translation, taking λ-terms to relational diagrams. Before we can do this, we need to know which λ-terms can be associated with relational diagrams at all. We will begin with some examples, and then we will make a conjecture. First, note that λ-terms that bind variables in a different order to the order they appear in the body cannot be represented by diagrams. Consider $\lambda xy.Ryx$, where R is a constant: one might naïvely attempt to represent it by taking the diagram for $R : e \to e \to t$, and swapping the left and right holes around. But holes have no distinguishing features—unlike simple diagrams, they are not tagged in anyway—so the result of doing this is no different from the diagram for R. Thus the structured picture we have been describing does not posit converses (in fact, we briefly mentioned in the introduction to Chapter 9 that converses make trouble for structured views). Similarly, we cannot represent terms where λ binds a variable occurring more than once. $\lambda x.Rxx : e \to t$ is a unary property, and there's no way to turn a simple diagram for R into a diagram with only one hole, unless we insert some other thing into Rs hole, or insert R into something larger, like $\exists_e$, which contains some grey that can fill that hole. We have no method for merging two holes into one. We also cannot represent terms where λ abstracts a variable that doesn't occur in the term, such as $\lambda x.Raa$. This is a unary property, but there's nothing we can do to the diagram for Raa to create a new hole without plugging it into a larger diagram and introducing new constituents. Finally, we cannot represent combinators, like $\lambda Xyz.Xyz$, or $\lambda p.p$, since that would seem to involve a diagram that consists entirely of holes, which cannot be built by complication:



Of course, we have discussed all of these restrictions on λ-terms, and have given them precise characterizations in general λ-languages, both in a traditional way in Section 9.7, and using substructural Curry typing systems in Chapter 10. The terms we appear to be looking at, then, are the structural terms (Definition 9.13). They can be presented in a traditional inductive way as follows:

- Any constant is structural.
- Application for variables and terms

  - If $M : \sigma \to \tau$ is structural, and $x : \sigma$ is a variable not appearing in M, then $Mx : \tau$ is structural.[11]
  - If $M : \sigma \to \tau$ and $N : \sigma$ are structural, then so is (MN).

- If $M : \tau$ is structural and $x : \sigma$ is the last element of $FV_s(M)$ (the rightmost variable of M), then $\lambda x.M : \sigma \to \tau$ is structural.

Corresponding to this inductive characterization of the structural terms, there is an equivalent inductive way of characterizing relational diagrams. Instead of taking complication as our basic way of making complex diagrams, we take the operation of application as basic—completely filling the leftmost hole of a diagram with another diagram that fits perfectly—and the operation of punching out a hole from a diagram. To do this we need to introduce diagrams representing variables. These are represented exactly like the diagram for a constant, except it is tagged by a variable. This gives us an extended notion of a relational diagram: variable diagrams are not themselves counted as relational diagrams, but can be constituents of relational diagrams. It will turn out that closed λ-terms correspond to relational diagrams in the original sense (that do not contain any variable diagrams as parts).

- $\mathbf{c}$ is a relational diagram for any constant c.
- You may plug a variable diagram or relational diagram into the leftmost hole of a relation diagram, provided it fits.
  - If $\mathbf{R}$ is a relational diagram and $\mathbf{x}$ a variable diagram that fits into the leftmost hole of $\mathbf{R}$ and doesn't appear in $\mathbf{R}$ then $(\mathbf{Rx})^0_0$ is a relational diagram.
  - If $\mathbf{R}$ is a relational diagram and $\mathbf{a}$ a relational diagram that fits into the leftmost hole of $\mathbf{R}$ then $(\mathbf{Ra})^0_0$ is a relational diagram.

- If $\mathbf{R}$ is a relational diagram, then the result of punching out the rightmost variable diagram in $\mathbf{R}$ is a relational diagram.

Observe that no hole can appear to the left of a variable diagram in a relational diagram: the only rule for making holes involves punching out variables, and you are only allowed to punch out the rightmost variable.

Exercise 11.9. *Using these rules, build the diagrams for the following terms, drawing a diagram for each step of the process*

a. $\lambda p.\Box(\Box p)$

b. $\lambda pxy.\wedge p(Rxy)$

c. $\lambda xyzw.\wedge(Rxy)(Szw)$

Playing around a little, it should be clear that this alternative set of rules generates the same collection of relational diagrams as the original rules involving complication. However, proving this at this juncture would not really be worth the effort. (The argument would be roughly parallel to the argument that the inductive characterization of the structural terms is equivalent to the characterization in the structural Curry typing system—Theorem 10.2. See also the final exercise in this Section relating the Curry system to relational diagrams).

Note that each of these four rules for making diagrams correspond exactly to the formation rules for structural terms. This gives us a way of mapping each λ-term, M, to a relational diagram M^d.

- $c^d = \mathbf{c}$.
- $(Mx)^d = (M^d\mathbf{x})^0_0$ (provided x does not occur in M).
- $(MN)^d = (M^dN^d)^0_0$.
- $(\lambda x.M)^d =$ the result of punching a hole where x occurs in M^d (provided x occurs once and is the rightmost occurrence of a variable in M).

Convention 11.2. *It's natural to extend the convention of bolding constants to make a name for a relational diagram to arbitrary λ-terms. Thus we will sometimes write $\mathbf{M}$ for M^d. (Note that in some circumstances the M^d notation is clearer, such as in the clause $(MN)^d = (M^dN^d)^0_0$ above.)*

We can now give an informal proof sketch that $\beta\eta$ is sound with respect to our diagrams:

Proposition 11.1 (Soundness of $\beta\eta$ for relational diagrams)**.** *If P and Q are immediately β-equivalent or η-equivalent then $\mathbf{P} = \mathbf{Q}$.*

Proof. Firstly, the instructions associated with $(\lambda\mathbf{x}.\mathbf{M})\mathbf{N}$ are: punch a hole in the diagram $\mathbf{M}$ where the diagram $\mathbf{x}$ occurs to make the diagram $\lambda\mathbf{x}.\mathbf{M}$. As observed previously, no variable appears to the right of a hole, so this new hole will be the leftmost hole of the diagram, and so applying it to $\mathbf{N}$ involves reinserting $\mathbf{N}$ where $\mathbf{x}$ was previously, which can be seen to be the diagram associated with the term you get by replacing x with N in M, i.e. $\mathbf{M}[\mathbf{N}/\mathbf{x}]$. (Intuitively, M provides an instruction for building a relational diagram using the rules for filling and punching out holes. $M[N/x]$ is the very same set of instructions, except at the point where x is introduced by the second rule letting you plug variables into the rightmost hole, you instead plug the diagram for N.) η has an even easier justification. $\mathbf{Mx}$ is the diagram you get by first plugging $\mathbf{x}$ into the leftmost hole of $\mathbf{M}$, and $\lambda\mathbf{x}.\mathbf{Mx}$ is the result of punching $\mathbf{x}$ out again, giving us $\mathbf{M}$. Of course, to show $\beta\eta$ is sound we also need to show that for any term M and N that result from substituting immediate β-equivalents or immediate η-equivalents have the same relational diagrams: $\mathbf{M} = \mathbf{N}$. This follows from the more general fact that if $\mathbf{P} = \mathbf{Q}$ then $\mathbf{M} = \mathbf{M}[\mathbf{P}/\mathbf{Q}]$ (i.e. if $P^d = Q^d$ then $M^d = (M[P/Q])^d$), which follows from the same idea that M and $M[P/Q]$ provide the same instructions for building a relational diagram, except at the point at which you insert $\mathbf{Q}$ you insert $\mathbf{P}$ instead; if these are identical then so is the end result. □

Since we have raised the question of the soundness of $\beta\eta$ with respect to relational diagrams, we should also say something about completeness.

Proposition 11.2 (Completeness of $\beta\eta$ for relational diagrams)**.** *For any structural λ-terms P and Q, if $\mathbf{P} = \mathbf{Q}$ then P and Q are $\beta\eta$-equivalent.*

The key to establishing this fact is to show that the translation described in Remark 11.6 undoes the translation from terms to diagrams up to $\beta\eta$-equivalence: i.e. $(M^d)^\lambda$ is $\beta\eta$-equivalent to M for every structural term M, which can be proved by induction on the structure of structural terms.

We finish the section with some exercises exploring relational diagrams in connection with the characterization of the structural terms in a Curry-style natural deduction typing system. Recall that natural deduction typing systems deliver sequents of the form $\Gamma \vdash M : \sigma$. So instead of assigning a relational diagram to the term M, we will look into associating relational diagrams to sequents. Intuitively, the relational diagram associated with this sequent is the diagram you get by taking the relational diagram associated with M and poking out all of the diagrams for variables appearing in it, as listed in Γ. (Thus the relational diagram for a sequent $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ really corresponds to the relational diagram for the λ-term $\lambda x_1 \ldots x_n.M : \sigma_1 \to \cdots \to \sigma_n \to \tau$ as opposed to $M : \tau$.) Since every closed λ-term corresponds to a derivable sequent with nothing on the left of the sequent, $\vdash M : \tau$, this translation also maps every closed λ-term to a diagram.

Now instead of assigning λ-terms to diagrams by induction on the structure of the term, as we did above, we will instead assign diagrams by induction on the length of *proofs* of the λ-terms in the relevant Curry system. Recall that the rules for the structural terms are: Constants, Application, Abstraction and Concretion. A notable fact about this system is that every term has a unique derivation (see Proposition 10.2). For example, we know that any judgment of the form $\Gamma \vdash \lambda x.M : \sigma \to \tau$ must have been derived by Abstraction from $\Gamma, x : \sigma \vdash M : \tau$, so that we can define the translation of the former typing judgment in terms of the translation of the latter, which we may assume has already been assigned an interpretation inductively.

- Constants: $(\vdash c : \sigma)^s = \mathbf{c}$.
- Concretion: $(\Gamma, x : \sigma \vdash Mx : \tau)^s = (\Gamma \vdash M : \sigma \to \tau)^s$.
- Abstraction: $(\Gamma \vdash \lambda x.M : \sigma \to \tau)^s = (\Gamma, x : \sigma \vdash M : \tau)^s$.
- Application: $(\Gamma, \Delta \vdash MN : \tau)^s = ((\Gamma \vdash M : \sigma \to \tau)^s (\Delta \vdash N : \sigma)^s)^n_m$, where m and n are the length of Γ and Δ respectively.

Observe that the translations for Concretion and Abstraction are trivial. This is because the diagram assigned to the sequent $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ is informally being understood as a diagram for the λ-term $\lambda \bar{x}.M$, so the premise of Abstraction, $\Gamma, x : \sigma \vdash M : \tau$ and conclusion $\Gamma \vdash \lambda x.M : \sigma \to \tau$ are associated with the same term. Similarly, the premise and conclusion for Concretion, $\Gamma \vdash M : \sigma \to \tau$ and $\Gamma, x : \sigma \vdash Mx : \tau$ are associated with η-equivalent terms, and we will think of these as representing the same diagram.

Exercise 11.10. *Find derivations of the following λ-terms in the structural Curry system, and draw diagrams for the terms at each line.*

a. $\lambda p.\Box(\Box p)$

b. $\lambda pxy.\wedge p(Rxy)$

c. $\lambda xyzw.\wedge(Rxy)(Szw)$

In this final exercise, you will prove that this process assigns the same diagrams to λ-terms as our original inductive characterization.

Exercise 11.11. *Say that a sequent $\Gamma \vdash M : \tau$ is* correct *if and only if $(\Gamma \vdash M : \tau)^s$ is the same diagram as the result of poking all the variable diagrams out of $\mathbf{M}$ (i.e. is the same as the diagram $\lambda \mathbf{x_1} \ldots \mathbf{x_n}.\mathbf{M}$ where $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$).*

a. The Constants and Abstraction rules trivially preserve correctness: $\vdash c : \sigma$ is correct, and if $\Gamma, x : \sigma \vdash M : \tau$ is correct, then so is $\Gamma \vdash \lambda x.M : \sigma \to \tau$. Briefly explain why.

b. Show that Concretion preserves correctness: if $\Gamma \vdash M : \sigma \to \tau$ *is correct, then so is* $\Gamma, x : \sigma \vdash Mx : \tau$.

c. Show that Application preserves correctness: if $\Gamma \vdash M : \sigma \to \tau$ *and* $\Delta \vdash N : \sigma$ *are correct then so is* $\Gamma, \Delta \vdash MN : \tau$.

11.5 Unique decomposition

The view we have explored in Sections 11.3 and 11.4 rejects the existence of converse relations: we have no way to represent them diagrammatically, and the λ-terms needed to talk about converses are not structural terms. Of course, for a given relation, such as *loves*, there could be another relation, *R*, that just happens to hold between individuals *a* and *b* if and only if *b* loves *a*. There could even be a relation for which this connection held necessarily, but such a relation would not be the converse of *loves*: if *R* were a true converse of *loves* then the proposition that *a* loves *b* must be the very same as the proposition that *b* *R*s *a*. But a natural structural principle states that a proposition can only have one decomposition into a binary relation and two arguments: the aforementioned proposition either contains *loves* as its constituent, or *R*, but not both.

On the other hand, we clearly have the active and passive voice in English: we have two binary predicates 'loves' and 'is loved by'. This creates a *prima facie* puzzle for the structured view. The question is not whether both of these predicates are meaningful—they clearly are—but how we are to understand them. One might posit, as contemplated above, another relation to be the denotation of 'is loved by': a new primitive relation *R* that necessarily holds between *a* and *b* iff *b* loves *a*, but is not a converse. This posit has little to recommend itself, however, since it is not only unparsimonious but also posits an unexplained necessary connection between *loves* and *R*.[12] An alternative is that the predicates 'loves' and 'is loved by' are semantically related to one and the same relation—depicted by a single relational diagram—with voice indicating which way around to correlate the argument places of the predicate with the argument places of the relation.[13]

A parallel set of remarks can be made about reflexizations, and vacuous λ-abstraction. Let's start with the former. The pictorial view we have been exploring rejects the existence of reflexizations of relations: there is a relational diagram or structural λ-term corresponding to the unary property *loves himself*, in addition the binary relation *loves*. A true reflexization of *loves*, *P*, not only applies to an individual *a* iff *a* loves *a*, but is such that the proposition that *a* is *P* is the same as the proposition that *a* loves *a*. Again this contradicts a natural decomposition principle stating that a single structured proposition cannot be fully decomposed in two different ways, one involving a binary relation, the other a unary property. As before, there is a structure friendly way of making sense of reflexizing constructs in English: there are not two entities that are the semantic values of 'loves' and 'loves oneself', there is a single relation, *loves*, with the reflexizing language telling us to feed the argument of the predicate to that relation twice. The case of vacuous λ-abstraction proceeds similarly: one doesn't need to posit a new property *being such that snow is white*, one can treat the English construction as simply disregarding the argument.

The pictorial view thus seems to be underwritten by some sort of unique decomposition principle. We'll next attempt to formulate this principle in higher-order language. Suppose you have two simple relations *R* and *S*, and you plug the (possibly complex) arguments *a* and *b* in that order into the former, the arguments *c* and *d* in that order into the latter, and the result is the same. Since every relational diagram (excluding type *e* diagrams)

has a unique 'outer relation'—the outermost grey box—R and S must be the same. Also the things filling the first and second argument places must be the same, so $a = c$ and $b = d$. Assuming we are working in a structural λ-language with a signature including all the logical constants, such as $=_\sigma$, $\rightarrow$ and the relevant kinds of quantifiers (see Section 9.4), and that this language is furthermore *logically perfect*—which guarantees, among other things, that distinct constants refer to distinct simple entities—we can formulate this as a schema:

$Rab =_t Scd \rightarrow (R =_{e\rightarrow e\rightarrow t} S \wedge a =_e c \wedge b =_e d)$ provided R and S are constants.

Note that this bears some resemblance to Predicate Argument Structure, but we have restricted the predicates to be constants. We saw already that when you allow complex predicates to take the place of R and S, the principle is not sound for the pictorial view: see for instance the ways of expressing $\Box(\neg P)$ as a complex operator applied to P or a simple operator applied to $\neg P$.

This idea obviously calls out for further generalization. For instance, if R and S are simple relations, and the result of plugging a and b into the 7th and 10th slot of R is the same as the result of plugging c and d into the 7th and 10th slot of S, then $R = S$, $a = c$, and $b = d$. Generally, in a logically perfect language we have the following schema:

$$(\ldots(Ra_1)^0_{n_1}\ldots a_n)^0_{n_k} =_\tau (\ldots(Sb_1)^0_{n_1}\ldots b_n)^0_{n_k} \rightarrow R =_{\sigma_1\rightarrow\cdots\rightarrow\sigma_{n_k}\rightarrow\tau} S \wedge a_1 =_{\sigma_1} b_1 \wedge \cdots \wedge a_{n_k} =_{\sigma_{n_k}} b_{n_k}$$

provided R and S are constants and $n_1 < n_2 < \cdots < n_k$. Finally, the above concerns the case of plugging entities into relations that fit the holes perfectly. But the same principle should hold if we use the more general operation of complication—plugging a hole with an entity that leaves some holes remaining. The idea is the same, however we need to be careful to get the superscripts and subscripts right: suppose I want to plug binary relations R and S into the two argument places of a binary connective C. Once you have plugged R into the first slot of C to make $(CR)^2_0$, to plug S into the second slot of C, you have to plug S into the *third* slot of $(CR)^2_0$, since R has filled up the first hole of C, but added two new holes. So, counterintuitively, the correct expression is $((CR)^2_0S)^2_2$. Thus the fully general principle can be stated as follows:

Outer Structure

$$(\ldots((Ra_1)^{n_1}_{r_1}a_2)^{n_2}_{r_1+n_1+r_2}\ldots a_k)^{n_k}_{r_1+n_1+\cdots+r_{k+1}} = (\ldots((Sb_1)^{n_1}_{r_1}b_2)^{n_2}_{r_1+n_1+r_2}\ldots b_k)^{n_k}_{r_1+n_1+\cdots+r_{k+1}} \rightarrow (R = S \wedge a_1 = b_1 \wedge \cdots \wedge a_k = b_k)$$

where, again, R and S are constants in a logically perfect language, and of course, the terms a_i and b_i have the correct type for complication to apply.

Exercise 11.12. *The complicated form of Outer Structure is due to the fact that expressions that represent plugging entities in from left to right adds holes which need to be carried forward in the subscripts. This could be circumvented by using expressions for the same relational diagram that complicate from right to left instead—since we count holes from the left, this leads to an intuitive numbering on the subscripts which corresponds to the argument place of the outer relation. Instead of writing* $((CR)^2_0S)^2_2$ *we write* $((CS)^2_1R)^2_0$ *for plugging S into the second hole first, and R into the first hole. Rewrite Outer Structure in this form.*

Finally, since no two relational diagrams can have the same outermost box—i.e. the same outer relation—we can similarly assert the distinctness of any pair of entities that have outer relations with different types, or have the same type but are being supplied with a different number of arguments, or arguments in different argument places. This may also be formulated as a schema in a logically perfect language:

Mismatch

$(\ldots((Ra_1)^{n_1}_{r_1}a_2)^{n_2}_{r_1+n_1+r_2}\ldots a_k)^{n_k}_{r_1+n_1+\cdots+r_{k+1}} \neq (\ldots((Sb_1)^{n'_1}_{r'_1}b_2)^{n'_2}_{r'_1+n'_1+r'_2}\ldots b_j)^{n'_j}_{r'_1+n'_1+\cdots+r'_{j+1}}$ R and S are logical or non-logical constants of different type or $k \neq j$, or $r_i \neq r'_i$ or $n_i \neq n'_i$ for some i.

In the next exercise, you will substantiate the informal arguments we gave above that converses, reflexizations and vacuous λ-abstraction violate the decomposition principles.

Exercise 11.13. *In this exercise, you will derive contradictions from Outer Structure and Mismatch from the assumption that there are simple converses, reflexizations and vacuous properties. You may assume constants $a, b : e$, $R, S : e \to e \to t$, and $F, G : e \to t$, each denoting simple entities, and you may assume that $a \neq_e b$, $R \neq_{e\to e\to t} S$ and $F \neq_{e\to t} G$. You should work in the structural higher-order logic $\mathsf{H}^{\mathcal{J}_S}$.*

a. *Obtain a contradiction from Outer Structure and the claim that R and S are converses: $\forall_{\hat{e}\hat{e}}\forall_{\hat{e}e\hat{e}}\lambda xyzw.(Rxy =_t Szw)$.*

b. *Obtain a contradiction from Mismatch and the claim that F is a reflexization of R: $\forall_{\hat{e}\hat{e}\hat{e}}\lambda xyz.(Rxy =_t Fz)$.*

c. *Obtain a contradiction from Outer Structure and the assumption that F is a vacuization of Ga: $\forall_{\hat{e}}\lambda x(Fx =_t Ga)$.*

Endnotes

1. See King (2008) for an overview.
2. Church was using the λ notation prior to Church (1940) but in the context of an untyped λ language which turned out to be inconsistent, and so had no interpretations. See Church (1932) and Church (1933).
3. If you have the principle Intensionalism from Section 7.4, and the necessity of Extensional β, one can obtain analogous unique theorems. This approach is taken up in Zeng (MS). Other strategies may be available, but like Intensionalism, will generally make substantive assumptions about the granularity that seem to be incompatible with structuralism.
4. Often these logics can be characterized by an algebra of truth values that enjoy certain fixed point properties. It is possible, in logics such as these, to construct models of Predicate Argument Structure using the general fixed point properties of the algebra characterizing the logic.
5. Russell (1908) and Whitehead and Russell (1910-1913) similarly treat their 'propositional functions' as propositions with constituents punched out. More recent discussions of this include Fine (2000), Leo (2008), Leo (2010), Gilmore (2013), and Dixon (2018).
6. Note that this sort of mismatch does not preclude a λ-language from being logically perfect: this requires the constants to line up one-to-one with the simple entities—it is perfectly fine for a logically perfect language to have several complex terms denoting the same entity.
7. See Russell (1903), Part IV 'Order', Section 218. Such attributions can be found in, for instance, MacBride (2007). For a more recent defense of the view, see Dixon (forthcoming).
8. See e.g. Lewis and Lewis (1970) and Casati and Varzi (1994).
9. Fine, for instance, denies the ontological primacy of holes in relations and instead works with a primitive notion of a proposition being a completion of a relation relative to the individuals a and b. A similar eliminative position should be possible here.
10. The reader may find it illuminating to recall the discussion of β and η in Section 3.3, where we noted that β and η corresponded to the idea that λ-abstraction and application were left and right inverses of each other.

11. The definition of structural term we adopted in Section 9.7 is slightly more liberal in allowing multiple occurrences of free (but not bound) variables within a single term. Such terms cannot be formed in the corresponding Curry system, **ND**[], and it's more convenient in the present context to ban them, so so we have accordingly modified the definition here.
12. For this sort of criticism see Bader (2020).
13. This is the sort of position explored in Williamson (1985).

Hints for exercises

[i] **Hint:** Consider diagrams that involve two occurrences of the same entity that can be built in two different ways.

12

Application

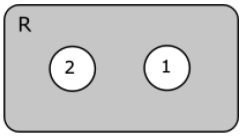
Structure II

In this chapter we continue our discussion of structured theories of reality. In Sections 12.1–12.3, we will consider variants of the structured picture of the previous chapter that allow for more ways of combining simpler entities to make complex ones—views that allow for converses, reflexizations and vacuous properties are treated in Section 12.1, views that allow for logical ways of combining entities are treated in Section 12.2, and views that posit combinators and other constituentless entities are treated in Section 12.3. In the final section we investigate the positionalist view of argument places in the higher-order framework.

12.1 Converses, reflexizations, vacuous λ-abstraction

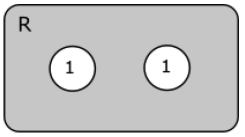
In this section we consider more liberal structured views that allow for more ways to put entities together than complication, or even $\bar{P}$-complication that will be introduced in Section 12.4. We'll consider adding devices that let us create converses and reflexizations of relations, thus increasing the number of λ-terms that are meaningful. We'll see that this will require us to expand our diagrammatic representations. As noted in Section 11.5, allowing for things like converses and reflexizations requires us to relax some structural principles by allowing a single structured entity to be decomposed in multiple ways.

We'll begin with converses and reflexized relations. We will try, as much as possible, to retain the diagrammatic picture thinking we applied to the previous structured view. A true converse, R^c, of a simple relation R, has the feature that for any individuals a and b, $R^c ba$ is Rab. Thus the diagram for R^c should be pictorially related to the diagram for Rab, in something like the same way that the diagram for R is related to the diagram for Rab: You get the latter diagram by plugging a into the first slot of R and b into the second slot. The diagram for R^c should be such that if you plug b into the first slot, and a into the second slot you get the very same diagram. Assuming we keep the pictorial representation of 'plug' the same—filling a hole with another diagram—this leaves only the interpretation of 'plug into the first hole' and 'plug into the second hole' open for modification. Given the above, the leftmost hole of R^c must be different from the first hole, and the rightmost different from the second hole. We can make the number of a hole explicit by tagging it with that number. Thus the $\lambda xy.Ryx$ gets represented as follows:



DOI: 10.4324/9781003039181-16

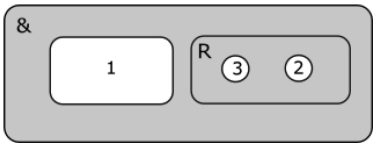
Every diagram considered previously can be thought of as an implicitly numbered diagram, where the numerical tags go from left to right, increasing by one between adjacent holes. A similar idea can be employed to deal with reflexizations. This time we allow multiple holes to be numbered by the same number, although conumbered holes must now have the same type. We modify the interpretation of 'plug into the nth hole' to simply mean fill every hole tagged by n with the diagram in question. Thus $\lambda x.Rxx$ gets represented as follows:



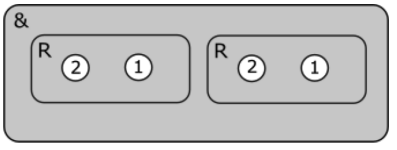
Officially, then, a numbered relational diagram can be identified with an ordered pair of a relational diagram and a surjective function mapping the holes of that diagram to the set $\{1, \ldots, n\}$ for some n. Surjectivity ensures that there are no gaps in the numbering (we could relax this to allow for vacuous λ-abstraction, but we won't treat that just yet).

Exercise 12.1. *Draw a diagram for predicate conjunction $\lambda x.(Fx \wedge Gx)$, where $F, G : e \to t$ and $x : e$.*

We are not yet done: we still need to say how complex diagrams are numbered. Lets start with a simple example of plugging an entity that fits perfectly into a hole: e.g. if we plug a into the first hole of the diagram for $\lambda xy.Ryx$ (i.e. the rightmost hole) then the resulting diagram has a remaining hole, that was originally number 2. Clearly this hole must now be renumbered to a 1, to reflect the fact that it is the first and only hole in the resulting property. Now consider a case of complication that leaves holes behind: I want to plug $\lambda xy.Ryx$ into the second hole of the conjunction connective, $\wedge$. Intuitively the first hole of the result is the leftmost hole, the second hole the rightmost hole, and the third the middle hole, so the result is as follows:



If I want to plug $\lambda xy.Ryx$ into the only hole of $\lambda p.(\wedge pp)$ we instead get something with two pairs of conumbered holes.



Now lets consider the general case. Suppose you have numbered relational diagrams $\mathbf{R}$ and $\mathbf{S}$. Then the complication of them $(\mathbf{RS})^n_m$ is the result of plugging $\mathbf{S}$ into all the holes numbered with an m. Each hole remaining in $\mathbf{S}$, which is originally numbered with a k, get numbered by $k + m - 1$ in the result of the complication (the mth hole of $\mathbf{R}$ is filled, so you take off 1). The holes numbered by a number j greater than m in $\mathbf{R}$, are renumbered by $j - 1 + n$ (again the mth hole of $\mathbf{R}$ is filled so you take off 1, and n holes added after the $(m - 1)$th hole so you add n). And the holes in $\mathbf{R}$ numbered by numbers less than m retain their original numbering in $\mathbf{R}$.

Exercise 12.2. *Draw the diagram for the term* $\forall_t \lambda p.\forall_t \lambda q.(p \wedge q =_t q \wedge p)$.[i]

We may also extend our diagrams to allow for vacuous λ-abstraction by dropping the surjectivity constraint. Thus there can be gaps in the numbering of holes in a diagram corresponding to 'invisible' holes. The idea here is that when you plug an entity into one of these holes it simply disappears and does not contribute a constituent to the resulting entity. There is a complication however: if $x : \sigma$ and $y : \tau$ are variables with different types, then the diagrams for $\lambda x.P$ and $\lambda y.P$, where P is a propositional constant, are the same, and indeed the same as the diagram for P itself. Intuitively the first has an invisible hole of type σ, the second an invisible hole of type τ, and the last no invisible holes. However, these three terms denote distinct entities of type $\sigma \to t$, $\tau \to t$ and t respectively. The problem is that when you allow for diagrams with invisible holes, the types of those holes cannot be read off from their pictorial representations. Thus in order to account for this, we must now think of a numbered diagram as an ordered triple of a relational diagram, a (possibly non-surjective) function from its holes to some set $\{1, \ldots, n\}$, and finally, a function from the $\{1, \ldots, n\}$ to types, specifying the type of each hole. The extra information would have to be represented along with the pictorial part of the diagram, e.g. by a list with a type written text to each number.

Exercise 12.3. *State rules for complicating numbered diagrams that allow for invisible holes. Make sure to explain how the function from numbers to types changes after complication.*

The modifications to relational diagrams we have just described correspond to different classes of λ-terms. The modification that allows for converses and reflexizations corresponds to the subrelevant λ-language, which can be characterized by the natural deduction system $\mathbf{ND}[PC]$ (you have the rules Permutation and Contraction, but not Identity or Weakening). If you allow invisible holes you get the subintuitionistic λ-language $\mathbf{ND}[PCW]$ (all the rules except Identity). And other possibilities can be represented too: if you require the numbering function to be a bijection, then you get the sublinear λ-language, $\mathbf{ND}[C]$, and so on. The reader interested in exploring these options should now have the tools to be able to define the translation between the relevant class of λ-terms and the relevant sort of diagram, and the question of the soundness and completeness of $\beta\eta$ with respect to these diagrams can be raised an settled using similar methods to the ones we have explored previously.

12.2 Logical modes of combination

Next we look at a completely different way of combining entities. How should structured theorists of various stripes treat logical connectives? One option is to posit the existence of special entities—logical connectives like $\wedge : t \to t \to t$, $\neg : t \to t$ and so on—that can be combined with propositions to make other propositions. Similar posits could be made for the quantifiers; in this case we may need to posit a wider range of primitive quantifiers if we are working in a general λ-language (such as $\forall_\sigma$ and $\exists_\sigma$ from Section 9.4). The most natural thing to say about these logical operations is that they are metaphysically simple.

Interestingly, the logical atomists, such as Russell and early Wittgenstein, did not think of the logical symbols '$\wedge$', '$\forall$', and so on, as standing for entities in the same way that names, predicates, and other non-logical constants do.[1] A subject-predicate proposition, such as *Socrates is human*, has *Socrates* and *being human* as constituents, and they are put together by applying the former to the latter. Application simply describes the way that these two constituents were put together, it is not itself a constituent of the proposition that *Socrates*

is human. According to Russell, even though this sentence contains the word 'is' to indicate subject-predicate application, it would be wrong to assume that this word stands for anything in reality; in the logical languages we have studied, and many natural languages too, we simply write the predicate and subject next to each other to indicate application, so there is no temptation to treat application as a constituent.[2] Russell and Wittgenstein took the same attitude to words like 'if', 'and', 'or', 'not' and so on.[3] The corresponding picture of reality, then, would be to posit new logical modes of combination in addition to the structural ones we have considered so far, such as application, complication and the like. The view is one in which there are disjunctive propositions, but no such thing as disjunction, negated propositions but no such thing as negation, and so on.

How would we describe a logical system that allows us to have disjunctive sentences without positing a connective constant corresponding to disjunction? In various logical contexts Russell treated conjunctions, disjunctions, and so on as he treated predicate application: rather than introducing constants 'denoting' special logical entities which combine with sentences to make disjunctive sentences, disjunctive sentences were introduced directly with their own syncategorematic rules. Thus in addition to the term formation rules like, 'whenever M is a term of type $\sigma \to \tau$ and N is a term of type σ, (MN) is a term of type τ', we would also have rules like 'whenever A and B are terms of type t, $(A \vee B) : t$ is a term of type t'. One can add such term formation rules to any of the formation rules we gave in Chapter 9 for specifying general λ-languages. Note that in the full λ-language, you can always recover the operation of disjunction from the syncategorematic rule, since propositional variables p and q on their own are terms of type t, and the syncategorematic rule for disjunctive sentences lets you form the formula $(p \vee q)$, and finally the disjunction connective $\lambda pq.(p \vee q)$ by abstraction. But it is not possible to recover disjunction in arbitrary general λ-language (recall the discussion in Section 10.3).

When we are adding syncategorematic rules to λ-languages which do not allow λ to bind multiple instances of variables, or variables out of order one also must make stipulations that prevent you from disjoining P and Q when they have variables in common. The cleanest way we had to deal with these issues was to employ the Curry typing systems considered in Section 10.3. The rules we gave there allowed us to create, for instance, disjunctive formulas without automatically introducing a term for disjunction:

$$\frac{\Gamma \vdash A : t \qquad \Delta \vdash B : t}{\Gamma, \Delta \vdash (A \vee B) : t}$$

As usual, Γ and Δ must be disjoint sequences of variables. The rules for the other truth-functional connective were similar. What about the quantifiers? Unlike λ, we had no special reason to think that quantifiers can't bind multiple variables at once or out of order. Thus our rule was:

$$\frac{\Gamma \vdash A : t}{\Gamma \setminus y_1 : \sigma, \ldots, y_n : \sigma \vdash \forall_\sigma x : A[x/y_1, \ldots, x/y_n] : t}$$

where $y_1 : \sigma, \ldots, y_n : \sigma$ are variables that are not bound in A, $x : \sigma$ doesn't appear in A, and $\Gamma \setminus y_1 : \sigma, \ldots, y_n : \sigma$ is the result of removing any of $y_1, \ldots, y_n$ that appear in Γ from Γ, otherwise leaving the ordering of variables alone. Recall that we used the letter L to indicate the presence of the logical rules in a Curry type system, so that $\mathbf{ND}[PL]$ for instance, denotes the system containing Permutation and the logical rules.

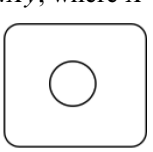
The possibility of denying the existence of the logical connectives by maintaining that propositions are closed under conjunction without positing an operation of conjunction in

its own right raises the question of whether we could eliminate other putative entities in the same way. The endpoint of this eliminativist project would be a view that rejects the existence of entities in any functional type $\sigma \to \tau$, by treating all apparent references to such entities in the same way that the Tractatus treats conjunction.

12.3 Combinators and pure entities

The structural views we have considered thus far are naturally formulated in λ-languages that do not contain combinators: expressions made entirely out of bound variables and λ. In Curry type systems one can create combinators as soon as one has the Identity axiom—Identity lets one derive the sequent, $x : \sigma \vdash x : \sigma$, from which one can immediately make the identity combinator using Abstraction, and various other combinators can be made depending on the other structural rules. What sort of entities are combinators from a structural perspective? Two expressions that differ only in their bound variables, such as $\lambda x.Fx$ and $\lambda y.Fy$, denote the same entity. It's extremely natural to explain this by appealing to the general thesis that bound variables in a term do not contribute constituents to the entity expressed by that term. Thus, for instance, the entities expressed by R, $\lambda xy.Rxy$, $\lambda zw.Rzw$, $\lambda xy.Ryx$ and $\lambda x.Rxx$ all contain the same constituents (although the converse and reflexization modes of combination are being applied to those constituents in the last two cases). By these lights, then, a combinator has no constituents at all for it is composed entirely of bound variables. The thesis that combinators are constituentless is further supported by the principle β, for the result of applying a combinator, such as $\lambda Xy.Xy$, to two simple entities F and a, yields something seemingly composed of the same simple entities $(\lambda Xy.Xy)Fa = Fa$, meaning that the combinator is not itself contributing any constituents.

One might try to naïvely represent combinators in the pictorial framework we developed in Section 11.3 by representing them by diagrams consisting entirely of holes. This is, at least, what would happen if you naïvely applied the rules about punching out constituents to get from the diagram for Xy to $\lambda Xy.Xy$, where $X : e \to t$ and $y : e$:



Unfortunately the pictorial view breaks down, for we now have two sorts of diagrams associated with entities in the type of a combinator, such as $(e \to t) \to e \to t$. We have the diagram above, which looks like a diagram for a simple entity of type $e \to t$ with the outer grey replaced with white, and we have the diagram for a simple entity of type $(e \to t) \to e \to t$ which consists of a grey rectangle with holes of shapes $(e \to t)$ and e punched out. This means we cannot have a uniform account of a hole of shape $(e \to t) \to e \to t$, into which both sorts of diagram can be slotted. Although this means we must reject the naïve picture thinking we have been adopting until now, it does not mean that the idea of a constituentless entity is incoherent: we must simply rely on logical tools we already have, such as the λ-calculus, to reason about them, or else find new diagrammatic representations.

Let us introduce the special word *pure* for such a constituentless entity. Pure entities do not appear to be a part of the most familiar structured pictures of reality, which require everything to be built up from fundamental or simple entities. But nonetheless they are a theoretically fruitful posit. Positing operations corresponding to combinators, for instance,

means we can quantify, in the object language, over modes of combination. Take the proposition that *John talks*, for example. There is sense in which this proposition can be built from *John* and *talks*. To explain this sense, however, we had to appeal to linguistic or pictorial metaphors: a way to build this proposition is to plug *John* into the only hole in *talks*. With combinators in the object language, these instructions for building this proposition can be represented by a combinator: the combinator corresponding to plugging an individual into a unary property is the application combinator $\lambda Xy.Xy$. Sometimes there are multiple ways to build a proposition: to make *John loves Mary* we can start with *loves* and plug *John* and then *Mary* into the two holes in that order, or we can first start with *loves* and plug *Mary* into the second hole and *John* into the first. Or indeed, we could start with *John* and then *Mary* and create the higher-order property of applying to *John* and *Mary* ($\lambda X.Xab$), into which we can plug *loves*. With combinators in the range of the higher-order quantifiers, we can count these ways of making this proposition: $\lambda Xyz.Xyz$, $\lambda Xyz.Xzy$ and $\lambda yzX.Xyz$.

12.4 Positionalism

I now want to examine the positionalist view described in Fine (2000), and mentioned in Section 11.2. This is a structured view that, like the view we have explored in Sections 11.3–11.5, thinks of relations as propositions with holes in them, and like that view rejects the existence of converses, reflexizations and the like, on the grounds that they allow a single proposition to be decomposed in different ways. But unlike the structured view we have been exploring, the positionalist does not posit a notion constituent order that linearly orders the holes and the immediate constituents of a relational proposition. According to our pictorial theory, the left-to-right appearance of holes and subdiagrams was taken to have metaphysical significance: the fact that a appears 'to the left' of b in Rab, and c comes 'to the left' of d in Scd correspond to an objective sense in which a and c occupy similar positions in these respective propositions. Of course, calling this ordering 'left-to-right' is tendentious: left and right are properties of the representations. The choice to associate the left-to-right ordering on the page with the ordering of constituents was entirely arbitrary—we could have use right-to-left, up-down, or some completely different ordering. But while the representation of the ordering is representation dependent, the fact that there is an objective constituent ordering allows us to make representation independent comparisons of position across distinct propositions. It makes sense to ask questions like 'does a appear in the same position relative to b in P as c does to d in Q?'—the answer could be yes, even if that is represented by a and c appearing, respectively, to the left of b and d in one representation, and to the right of them in another.

Fine's positionalist rejects the meaningfulness of these interpropositional comparisons of order. A binary relation like *loves* has two holes, and there are consequently two ways of plugging *John* and *Mary* into those holes. There are also two holes in the relation *hates*, but there is no non-arbitrary sense in which the two holes in *loves* can be correlated with the two holes in *hates*, as they can on the ordered view (where one can correlate the first argument place of *loves* with the first argument place of *hates*). It is possible to represent the positionalist view about simple relations using simple relational diagrams. When we represent *loves* this way, for instance, we are arbitrarily associating the left hole in the diagram with one of the argument places of *loves* and the right hole with the other; we make similar arbitrary choices when we represent *hates* and other binary relations. The fact that the four holes in *loves* and *hates* can be paired off—two left holes and two right holes—depends on this

arbitrary representational choice, and a different choice would pair them off in a different way. These remarks apply equally to language. There is a clear sense in which the two argument places of the English predicates 'loves' and 'hates' line up: one of them comes first, the other second. In a typed language predicates similarly have argument places in a particular order. On the present view, one must make an arbitrary choice about how to match the first and second argument places of a relational constant with the argument places of the underlying relation. Note, however, that once the arbitrary choice has been made for the relational constants, there may be natural conventions for assigning the arguments of complex predicates to the corresponding complex relations: for instance we will argue that the two argument places of *doesn't love* correlate in a straightforward way with the two argument places of *loves*, so there is only one natural way to associate argument places of the relation *doesn't love* to the arguments places of the predicate 'doesn't love' once you have made a given choice about 'loves'. (Although we will encounter further complications about how to make these associations when we consider positionalists who want to make room for reflexizations and other operations that merge argument places.)

We should remark on one more thing before we move on. The view we are considering rejects the idea both of a non-arbitrary trans-propositional notion of order of constituents in a proposition, and also a non-arbitrary trans-propositional notion of position in a proposition. These are not the same. One might be able to say that *John* occupies the same position in *John loves Mary* as *Jack* does in *Jack hates Jill* without saying that that position comes before or after the position that *Mary* and *Jill* occupy in these respective propositions. Perhaps there are simply two positions, *pos* and *arg*, that one can occupy in any binary relation, but there is no ordering on them. Similarly, perhaps all ternary relations have three positions, *pos*, *arg*, and *slot*—that any ternary relation shares with any other—while there is no non-arbitrary sense in which one of the argument places comes between the other two. The important point here is that there is an view intermediate between the ordered view discussed in Section 11.3, and positionalism, which rejects objective order but accepts objective positions.

What about complex entities? Despite rejecting converses, the positionalist, presumably, should still think that it is possible to plug *John* into either hole of *loves*, to compose *not* with *loves* to make *doesn't love*, and so on. That is, they should accept all the things we can make by complication. Thus we should begin by saying how to represent complications of diagrams. (Whether complication is *enough* to generate all the structured entities that the positionalist accepts is a question we will take up shortly.)

The complex relation *doesn't love* has two argument places, but unlike the case of simple diagrams we cannot associate these argument places with the two holes in the relational diagram $(\neg\mathbf{L})^2_0$ arbitrarily (cf. the diagram for $(\neg\mathbf{R})^2_0$ from earlier). Each argument place of *loves* is identical to one of the argument places of *doesn't love*, so if we have associated the leftmost hole in a diagram with an argument place p when we made the arbitrary choice when we represented *loves*, then we must associate the leftmost hole for that argument place in *doesn't love*. It follows that once we have made an arbitrary choice about how to associate holes in simple relational diagrams to argument places of simple relations, we have no choice about how we associate holes in complicated diagrams with the argument places of the corresponding complications of relations.

Since the positionalist thinks there is no non-arbitrary way to correlate the argument places of two simple relations, like *loves* and *hates*, the fact that we can sometimes identify argument places in a complex relation with argument places in its component relations is

not obvious and deserves an argument. The relation *loves* and the relation *doesn't love*, i.e. L and $\lambda xy.\neg(Lxy)$ (i.e. $(\neg L)^2_0$), both have two holes. Call the holes in *loves*, o and p, and the holes in *doesn't love* q and r. Suppose that plugging *John* and *Mary* into o and p in *loves* makes *John loves Mary*. We know that applying negation to the result, *John doesn't love Mary*, is the same as composing *not* and *loves* to make *doesn't love* love, and then either plugging *John* in q and *Mary* into r, or plugging *John* into r and *Mary* into q (since one of these has to make *John doesn't love Mary*). In the first case, we know that o corresponds with q and p with r, in the second o with r and p with q. The simplest way to explain this correspondence is if the holes where in fact identical: thus $o = q$ and $p = r$, or $o = p$ and $p = q$. Note, however, that the move from this correspondence between argument places to their identification is not inevitable, and may in some cases be problematic. For instance, a metaphysician who believes in reflexizations, such as $\lambda x.Lxx$, finds a correlation between the single hole in $\lambda x.Lxx$ and both holes in L, yet they cannot by identical, since L must have two holes.

Let's work through another example. *loves* is a proper constituent of *doesn't love*. Can there be cases of identification/correlations of argument places between pairs of relations where neither is a constituent of the other? The next example seems to suggest this can happen too. Let R be a ternary relation, and consider two binary relations, one obtained by plugging a into one of the argument places, the other by plugging b into a different argument place: say, $\lambda xy.Raxy$, $\lambda xy.Rxby$. Neither of these relations is a part of the other. Call the argument places of $\lambda xy.Raxy$ o and p, and the argument places of $\lambda xy.Rxby$ q and r. Without loss of generality, suppose plugging b into o and c into p will give you $Rabc$. Now $Rabc$ must also be the result of either plugging a into q and c into r, or a into r and c into q. In the first case we find that p and r are correlated, and in the second case we p and q are correlated.

Fine offers an argument against positionalism based on identifications of argument places across relations. Take a symmetric relation, like identity. One might be attracted to the idea that *being identical to John* and *being such that John is identical to you* are the very same property:

$$(\lambda x.x =_e a) =_{e\to t} (\lambda x.a =_e x)$$

But if identity has two argument places, o and p, then plugging *John* into o leaves a property whose argument place is p, and plugging *John* into p leaves a property with argument place o. Since the argument places are different, so are the properties. Now Fine's argument is invalid as it stands. For all we've said the result of plugging *John* into o produces a property with a completely new argument place q, and q could be the same argument place in the property that results from plugging *John* into p. However, the considerations above show that we could make identifications between the argument places of a non-symmetric relation R and the argument places of $\lambda x.Rax$ and $\lambda x.Rxb$, and the simplest and most systematic way to do this is to assume that plugging behaves uniformly: if plugging an individual into a non-symmetric relation preserves the identities of the remaining holes, so does plugging an individual into a relation like identity.

Let's now consider, in an exploratory spirit, a potential problem for positionalism. It common to assume that whenever you have two propositions, properties, relations, etc. there is another proposition, property, or relation that is their disjunction. The idea of a univocal disjunction of two relations, however, appears to let us introduce a non-arbitrary interpropositional notion of sameness of position. For the considerations analogous to the above allow us to identify the two positions in *loves* with the two positions in *loves or hates*. If you

plug *John* and *Mary* into the two positions of *loves* in one of the two possible ways, you get a proposition that entails the result of plugging *John* and *Mary* into the two positions of *loves or hates* one way around but not the other. The argument place into which *John* is plugged in the original proposition and the disjunctive proposition can be correlated when one of the results entail the other, and similarly the argument places into which *Mary* is plugged can be correlated. By completely symmetric reasoning the positions of *hates* may be correlated with the positions in *loves or hates* finally allowing us to non-arbitrarily correlate the two positions in *loves* with the two positions in *hates*.

Firstly observe that this is an argument for correlating argument positions in different simple binary relations, and thus perhaps reinstating the meaningfulness of trans-propositional claims of the form '*a* occupies the same position in *P* as *b* does in *Q*'. But it does not reinstate a non-arbitrary ordering of holes, or the meaningfulness of a non-arbitrary trans-propositional notion of constituent order. If all one cared about was avoiding objective order, one could simply retreat to the weaker view mentioned above that merely rejects order, but accepts objective comparisons of position. The stronger version of positionalism we've described, however, must reject the univocal notion of a disjunction of relations. We will consider in turn the idea that there are several disjunctions of *loves* and *hates*, and the idea that there are no disjunctions of these relations.

One might naïvely think that if there are to be multiple ways of disjoining *loves* and *hates* then there should be four ways of disjoining them. For this is what one gets by counting the possible λ-terms that bind one variable from each disjunct: $\lambda xy.(Lxy \vee Hxy)$, $\lambda xy.(Lxy \vee Hyx)$, $\lambda xy.(Lyx \vee Hxy)$ and $\lambda xy.(Lyx \vee Hyx)$. But this idea doesn't work: two of these four disjunctions are converses of the other two and so must be rejected for the same reason that we rejected the converses of simple relations (for instance *John loves or hates Mary* may be decomposed into two different relations applied to *John* and *Mary* in different orders). What we should really say is that there are only two disjunctions: we can think of this as the result of starting with the four-place 'Quinean' disjunction (see Quine (1967)), $\lambda xyzw.(Lxy \vee Hzw)$ and then either 'merging' the first and third argument places and the second and fourth to make a two place relation, or merging the first and fourth, and the second and third. These are the only two ways of merging argument places belonging to different disjuncts of the Quinean disjunction. It simply is not possible to make a converse if your only operation is merging of argument places. The feeling that there ought to be four disjunctions no doubt comes from the feeling that it would be entirely arbitrary to select one of each of the two pairs of converse λ-terms as legitimate and not the other. But I think this feeling stems from the fact that there are two equally good choices about how to impose an order on the argument places after merging. Supposing we arbitrarily order the argument places in the Quinean disjunction, $\lambda xyzw.(Lxy \vee Hzw)$, as 1–4 (from left to right), where 1 and 2 correspond to places in *loves* and 3 and 4 correspond to places in *hates*. If we merge places 1 with 4, and 2 with 3, do we consider the former resulting argument place to be to the left or the right of the latter result (for representational purposes)? Neither option is more natural than the other. One choice makes the pair of terms $\lambda xy.(Lxy \vee Hxy)$ and $\lambda xy.(Lxy \vee Hyx)$ natural to represent the two disjunctions, and the other makes $\lambda xy.(Lyx \vee Hxy)$ and $\lambda xy.(Lyx \vee Hyx)$ natural instead. We can represent the general sort of merging needed to create these using a variant of our substructural rule of Contraction that lets us contract non-adjacent variables. There are two ways of doing this depending on whether the result of the contraction leaves behind the variable on the left, or on the right.

The first option is:

$$\frac{\Gamma, x : \sigma, \Delta, y : \sigma, \Theta \vdash M : \tau}{\Gamma, z : \sigma, \Delta, \Theta \vdash M[z/x][z/y] : \tau}$$

and the second option is:

$$\frac{\Gamma, x : \sigma, \Delta, y : \sigma, \Theta \vdash M : \tau}{\Gamma, \Delta, z : \sigma, \Theta \vdash M[z/x][z/y] : \tau}$$

In Chapter 10 we called these rules Left Contraction and Right Contraction. Our original rule of Contraction, C, is simply the special case of either rule where Δ is empty. Thus, the positionalist that allows for merging of argument places could choose (arbitrarily) a typing system that contains one of Left Contraction or Right Contraction (but not both). We'll use C_L and C_R to denote these rules respectively.

Exercise 12.4.

a. Show that $\lambda xy.(Lxy \vee Hxy)$ *and* $\lambda xy.(Lxy \vee Hyx)$ *may be typed in* $\mathbf{ND}[C_L]$.

b. Show that $\lambda xy.(Lxy \vee Hxy)$ *and* $\lambda xy.(Lyx \vee Hxy)$ *may be typed in* $\mathbf{ND}[C_R]$.

Obtaining the remaining two ways of representing the two disjunctions—namely (i) $\lambda xy.(Lyx \vee Hyx)$ and $\lambda xy.(Lxy \vee Hyx)$ and (ii) $\lambda xy.(Lyx \vee Hyx)$ and $\lambda xy.(Lyx \vee Hxy)$—would require us to modify the conventions about order baked into our natural deduction system another way, but this could be done also.[4]

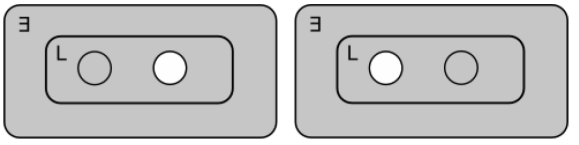
Note, finally, that one of these λ-terms—namely $\lambda xy.(Lxy \vee Hxy)$—looks more natural than the others, and indeed is the 'standard' definition of relational disjunction. However, the naturalness of this definition also depends on the artificial choice we made to associate the left and right places of the predicates 'L' and 'H' with the corresponding argument places of the relations: If we had made a different choice for 'L' but not 'H', then the natural looking term '$\lambda xy.(Lxy \vee Hxy)$' would have expressed the other disjunction.

Remark 12.1. There are still a number of remaining puzzles for the view that there are two disjunctions of a pair of relations. Suppose we use the Curry system $\mathbf{ND}[C_L]$ for representing relations with λ-terms. For each notion of disjunction, we seem to have two ways to disjoin *loves* and *hates*—*loves or hates* and *hates or loves* (these are conceptually different, even if they appear logically equivalent: consider, for instance, the relational analogue of material implication which is not symmetric). The problem is that, on the assumption that $(Lab \vee Hba) =_t (Hba \vee Lab)$, $\lambda xy.(Lxy \vee Hyx)$ and $\lambda xy.(Hxy \vee Lyx)$ are converses of each other, and one can decompose a single proposition, $Lab \vee Hba$ into two different (non-coextensive) relations applied to a and b.

On the further assumption that we accept the existence of *operations* corresponding to these two forms of disjunction we get a secondary puzzle. We can represent the operations as $\vee^1_{e\to e\to t} := \lambda XYxy.(Xxy \vee Yxy)$ and $\vee^2_{e\to e\to t} := \lambda XYxy.(Xxy \vee Yyx)$. The puzzle is that one of these notions of relational disjunction is symmetric and the other is not. For instance $R \vee^1 S$ is coextensive with $S \vee^1 R$, but this does not hold for $\vee^2$. Perhaps this means we need to multiply our notion of coextensiveness and identity between relations as well as disjunction: perhaps two relations can be coextensive and identical in a correlated and an anti-correlated way (where which counts as 'anti-correlated' is merely an artefact of representational choices).

The other way to respond to our argument is to reject disjunctions altogether. The options above require us to adopt contraction-like rules, Left Contraction and Right Contraction, that let us build reflexized relations. Someone who took the argument against converses from the unique decomposition seriously, should also be moved to reject terms where λ binds variables occurring more than once, such as $\lambda x.Lxx$. Relational disjunctions, like $\lambda xy.(Lxy \vee Hxy)$ have the same problem that x and y both appear twice, allowing us to decompose *John loves or hates Mary* into the binary relation $\lambda xy.(Lxy \vee Hxy)$ applied to *John* and *Mary*, or the quaternary relation $\lambda xyzw.(Lxy \vee Hzw)$ applied to *John*, *Mary*, *John* and *Mary*. (Note, of course, that the quaternary relation, $\lambda xyzw.(Lxy \vee Hzw)$, is fine by the present lights since it can be constructed by complication: $((\vee L)^2_0)H)^2_0$.)

The next order of business is to ask whether complication, or perhaps complication and merging, is complete for the positionalist view: can every entity acceptable to the positionalist be made from the simple entities by complication? We have shown how to reinterpret our relational diagrams so that they are acceptable to the positionalist. But there was a point in our description of the operation of complication that essentially appealed to order. Consider an example where we fill a relational diagram containing an $e \to t$ hole, with a binary relational diagram of shape $e \to e \to t$, leaving a hole behind. The rule is that you fill the rightmost hole of the inserted diagram with grey and leave the leftmost hole behind. But if the choice to associate one of the argument places of a relation with left or right was arbitrary, then the stipulation that this is the only legitimate way of combining these diagrams also seems arbitrary. The positionalist should think, for instance, that there are two ways of combining *someone* with *loves*: you can make *loves someone* and *someone loves*. We can use the grey circle in the diagram for *someone* to fill either hole of the diagram for *loves*.



Now suppose we had an $(e \to e \to t) \to t$ diagram into which *loves* perfectly fits—for instance the higher-order property of being *transitive*. By the same reasoning, there should be two ways to plug *loves* into this higher-order property: there are two grey circles in the larger diagram, and so there are two ways around we could associate those grey circles with the holes in *loves*. In this case, one could imagine that the diagram for *loves* is written on a transparency and can be slotted into the hole in two ways depending on whether we turn it over or not. Intuitively the resulting propositions are different: one says that *loves* is transitive, the other that *is loved by* is transitive. Note, however, that we do not really need to posit the existence a converse relation, *is loved by*, to obtain both propositions: we just have two ways of plugging a single relation into a higher-order property of relations. Generally, for $n \geq k$, there are $\frac{n!}{(n-k)!}$ ways of plugging a k-ary relation between individuals into a higher-order property of n-ary relations, corresponding to all the ways of pairing off a grey circle with a hole. To properly repesent this diagrammatically we need to be able to distinguish different grey circles so it is clear which grey circle is plugging each hole (the holes, we may assume, are still being distinguished from one another by their position left to right). We could do this by tagging each grey circle with a unique identifier. As a λ-term this mode of combination can be written as follows. Given relations R and S, you can always make another relation

$\lambda\bar{x}\bar{y}.R\bar{x}(\lambda\bar{z}.S\bar{w})$

Here, as usual, the overline notation denotes a non-repeating sequence of variables. Here $\bar{w}$ is an $n+k$ length sequence of variables that contains exactly one occurrence of each of the variables in $\bar{y} = y_1, \ldots, y_n$ and $\bar{z} = z_1, \ldots, z_k$ with the stipulation that $y_1, \ldots, y_n$ appear that order (i.e. y_i appears befor y_j in $\bar{w}$ only if $i < j$). It is important that we do not let the $x_1, \ldots, x_r, y_1, \ldots, y_n$ appear out of this order in the body, since this would have the consequence of letting in pairs of entities that are converses of each other. We have been focusing on relations between individuals, but the above makes sense for relations of any type, provided the above λ-term is well-typed.

Example 12.1. *Suppose* $T : (e \to e \to t) \to t$ *is a higher-order property of relations meaning* transitive*, and* $R : e \to e \to t$ *is some binary relation, such as* loves*. T and R can be combined in two ways: TR and* $T(\lambda xy.Ryx)$*. These can both be seen to be* $\beta\eta$*-equivalent to instances our general mode of combination above where* $\bar{x}$ *and* $\bar{y}$ *are empty. TR is the case where* $\bar{z} = \bar{w} = z_1z_2$*, ad* $T(\lambda xy.Ryx)$ $\bar{z} = z_1z_2$ *and* $\bar{w} = z_2z_1$*.*

Suppose $S : e \to e \to e \to e \to t$ *is a quaternary relation. Then there are 12 ways of combining T and S:* $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1y_2z_1z_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1y_2z_2z_1)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1z_1y_2z_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1z_1z_2y_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1z_2y_2z_1)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sy_1z_2z_1y_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_1y_1y_2z_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_2y_1y_2z_1)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_1y_1z_2y_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_2y_1z_1y_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_1z_2y_1y_2)$, $\lambda y_1y_2.T(\lambda z_1z_2.Sz_2z_1y_1y_2)$.

Does this extension of complication capture all the possible ways of combining entities that is acceptable to the positionalist? I think it is unproductive to pretend there is a single 'positionalist view', and that there's a definite answer to this question. Rather I will explore further modes of combination that seem in keeping with the structured position we are exploring here, which we may think of as corresponding to different ways of developing the positionalist view. That said, I think there is a natural positionalist view that accepts the most general mode of combination discussed below—what I call $\overline{P}$-complication. For reasons of space, we will spend less time developing diagrammatic representations that can account for these new modes of combination (although I believe they are possible) and focus on articulating the view using the tools we have developed in previous chapters.

There are two directions in which we could generalize. We claimed there were two ways of plugging *loves* into the higher-order 'transitivity' property, to create the proposition that *loving* is transitive, and the proposition that *being loved by* is transitive. This allows us to say things we would normally say using converses without positing the existence of converses. Perhaps it is also possible to do analogous things for reflexizations and vacuous abstractions. We might posit a way of plugging the binary relation *loves* into the unary quantifier *someone* to get *someone loves himself*. The thought is not that we are first creating a unary property *loves himself* and plugging that into *someone*, we are rather directly combining *someone* and *loves* in a new way. We might visualize this as letting the single grey circle in the diagram for *someone* fill two holes at once (perhaps by folding the diagram for loves in half so that the two holes can both be filled by a single grey circle). Similarly, we might posit a way of combining the unary quantifier *someone* with the proposition that *snow is white* to make the proposition that *something is such that snow is white*, without positing the intermediate step of creating the vacuous property of *being such that snow is white*. This generalizes our previous mode of combination: suppose R and S are higher-order relations, then we can make a new relation as follows:

$\lambda\bar{x}\bar{y}.R\bar{x}(\lambda\bar{z}.S\bar{w})$

as before $\bar{w}$ is a sequence of variables containing the variables in $\bar{y}$ and $\bar{z}$, with the variables in $\bar{y}$ each occurring exactly once and in order. But this time $\bar{w}$ may also contain multiple occurrences of z_i, or no occurrences of z_i for each variable z_i appearing in $\bar{z}$.

If we think of this direction of generalization as generalization along the dimension of width, the following direction of generalization can be thought of as a generalization of depth. Take a property of relations, like *being well-founded*, $W : (e \to e \to t) \to t$. It is common in logic to talk about a related property of being converse-well-founded that can be defined as $\lambda X.W(\lambda yz.Xzy)$. There is a corresponding notion for any higher-order property of relations: one can be converse-transitive, converse-symmetric, and so on. Converse-well-foundedness is not a converse of well-foundedness, it is what we might call a 'hereditary converse' of well-foundedness. We may formalize this with a (skippable) definition:

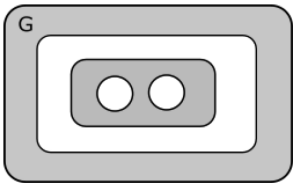
Definition 12.1 (Hereditary converse)**.** *First we inductively define the notion of one term being a hereditary converse of another.*

M is a hereditary converse of M for any term M.

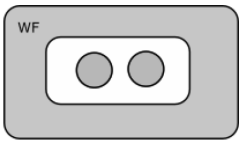
If $P_1, \ldots, P_n$ are hereditary converses of variables $y_1, \ldots, y_n$, and π is a permutation of $\{1, \ldots, n\}$, then $\lambda y_1 \ldots y_n.MP_{\pi 1} \ldots P_{\pi n}$ is a hereditary converse of M.

A term has finitely many hereditary converses, so one can define the relation 'X is a hereditary converse of Y' in the object language with a finite list $\lambda XY.(X = P_1 \vee \cdots \vee X = P_n)$ where $P_1, \ldots, P_n$ enumerate the hereditary converses of Y.

The positionalist suspicious of converses should be suspicious of hereditary converses too. I can make a single proposition in two ways: by plugging R into the argument place of converse-well-foundedness in one orientation, or plugging R into well-foundedness the opposite way (i.e. letting the two grey circles fill the holes of R the other way around), allowing us to decompose a single proposition in two different ways. Thus: one might posit multiple ways of plugging the property of being well-founded into a higher-order property G of type $((e \to e \to t) \to t) \to t$, yielding the propositions that well-foundedness is G and converse-well-foundness is G. Again this must be thought of as two ways of plugging a single well-foundedness relation into a higher-order property G, without positing a further property of converse-well-foundedness. We can visualize this as follows: the diagram for G looks like this:



there are two white circles inside a grey band. The diagram for well-foundedness itself contains two white circles:



which can filled by the two grey circles in two different ways. Again, we may think of the diagram for WF as written on a transparency which can be laid on top of the diagram for G in two different ways by turning it over. This presents us with another level of generalization, that can also be combined with the first. For instance, there is a notion that stands to ordinary reflexization as the notion of a hereditary converse stands to a converse. For example, in the full λ-language, for every property Q of unary properties (i.e. first-order quantifier), there is a property of binary relations, having a Q reflexization: $\lambda X.Q(\lambda y.Xyy) : (e \to e \to t) \to t$. We once we have allowed ourselves to plug binary relations into holes for unary properties, letting a grey circle fill two holes at once, we should be wary of positing these variant properties for the same reason: that a single proposition can be decomposed one way so as to have Q as the predicating constituent, and another way so as to have *having a Q reflexization* as the predicating constituent. Nonetheless, we could allow for a new way to combine Q with a higher-order property $G : ((e \to e \to t) \to t) \to t$ to get the proposition that having a Q reflexization is G. There is a similar hereditary generalisation of vacuous λ-abstraction, and these terms are problematic for parallel reasons. But, as above, the composite operation of applying this generalization to an entity and plugging it into a higher-order entity might still be legitimate. The possibilities become complex quickly.

Let me end our discussion, then, with a conjecture about what modes of combination might be acceptable to the most liberal kind of positionalist. Suppose R and S are relations. Then it is always possible to combine R and S to make the following entity, expressed with a λ-term:

$\bar{P}$-complication $\lambda \bar{x} \bar{y}.R\bar{x}(\lambda \bar{z}.S\bar{P})$

provided the result is well typed. Here $\bar{P} = P_1, \ldots, P_j$ is a sequence of λ-terms such that:

1. It contains the variables $y_1, \ldots, y_n$ in that order (i.e. $\bar{y}$ is a subsequence of $\bar{P}$).
2. The remaining terms are arbitrary pure λ-terms whose free variables all belong to the set of variables in the sequence $\bar{z}$.

$\bar{P}$-complication includes all of the modes of combination we have discussed already, and many others we have not discussed. These modes of combination are 'structural' in the sense that it appears to be consistent to maintain that every proposition has a unique outer relation into which various arguments are plugged in a now very general sense of 'plug', and that one can uniquely recover those arguments and the manner in which they were plugged from the structure of the proposition as well. I leave that question open, along with many other questions about the characterization of the structured view that every entity can be constructed by $\bar{P}$-complication. For instance, the result of taking some constants and closing under $\bar{P}$-complication and β-reduction is a general λ-language. It would be nice to find a diagrammatic representation of these terms along the lines suggested above, and it would also be illuminating to find a substructural type theory that characterized them exactly.

Endnotes

1. Russell (1918) 'You must not look about the real world for a object which you call "or", and say, "Now, look at this. This is 'or'." p39, Wittgenstein (1922) 5.4 'At this point it becomes manifest that there are no 'logical objects' or 'logical constants'.'
2. Russell writes "Take " Socrates is human." Here "is" is not a constituent of the proposition, but merely indicates the subject-predicate form'. Russell (1919) p200 . Note that, *pace* Russell, it's common in semantics to assign

the word 'is' a semantic value, specifically the application function $\lambda Xy.Xy : (e \to t) \to e \to t$ as a meaning; see for instance Heim and Kratzer (1998), Section 4.1.

3. This allowed Wittgenstein, for instance, to identify p and $\neg\neg p$ when he would have had to distinguish them if $\neg$ stood for something (Wittgenstein (1922), Section 5.44).
4. For instance the rule for λ-abstraction, Abstraction, $\frac{\Gamma,x:\sigma\vdash M:\tau}{\Gamma\vdash\lambda x.M:\sigma\to\tau}$ abstracts from right to left, but we could have chosen set things up in the other way, where the rule for λ-abstraction was instead $\frac{x:\sigma,\Gamma\vdash M:\tau}{\Gamma\vdash\lambda x.M:\sigma\to\tau}$.

Hints for exercises

[i] **Hint:** Firstly write it in prefix notation. Secondly, think about how you might build this using complication, reflexization and converses.

13

Application

Structure III

In this final chapter on structure we look at some ways to theorize about structure in the object language. In Section 13.1, we introduce some further theoretical primitives that can be added to the signature of higher-order logic in order to express structural ideas that cannot be expressed in the logical signature alone. In Section 13.2, we show how to formulate object language theories corresponding to the sorts of structural views considered in previous chapters. Section 13.3 points the reader to some further literature relating to the three chapters on structure.

13.1 Theoretical primitives

In the preceding chapters, we have considered a variety of views about the structure of reality, but we have generally described these views in fairly metaphorical ways—by way of analogy with the structure of languages or certain sorts of diagrams. The metaphors are helpful for guiding us when formulating higher-order theories of the structure of reality, but should not be taken as substitutes for such formulations. Indeed, the analogies with languages and diagrams encourage certain ways of talking that are not obviously in good standing. For instance, we can talk about a relational diagram being built from some other relational diagrams when the former can be obtained by putting the latter diagrams together in some combination. Similarly, we talk of an expression in a language being definable from some other expressions when those expressions can be put together according to the syncategorematic rules of the language to make the former. Since diagrams and expressions are first-order entities, the first-order quantifiers range over all diagrams and expressions whatever their type, and we can always make the quantification over the possible ways of combining diagrams and expressions explicit. But the notion analogous to definability for higher-order entities is not obviously expressible in the pure language of higher-order logic, because it involves quantifying over entities of arbitrary type. Now, according to some of the positions we have explored, there are only finitely many ways to combine a finite list of entities to make another entity of a given type. In this case we can express the idea that one thing can be defined from a finite list of other entities with a finite disjunction. This could be true, for instance, if your only mode of combination was complication. However, a finite expression of metaphysical definability may not in general be available.

To take another pair of examples, in Section 12.3 we introduced the notion of a pure entity that didn't have any constituents, and we have throughout talked about metaphysically simple, or 'fundamental', entities. But these are not obviously notions we can simply define

DOI: 10.4324/9781003039181-17

in the pure signature of higher-order logic: for instance the linguistic analogue of a simple entity of type τ in an applicative language is a constant of that type—an expression that cannot be written of the form MN for any terms $M : \sigma \to \tau$ and $N : \sigma$. Naive attempts to come up with an analogous higher-order definition of the simplicity of an entity of type τ will inevitably fail, since one needs to quantify over all pairs of entities of type $\sigma \to \tau$ and type σ for every possible σ all at once.

Nonetheless, all of these notions seem to be intelligible even if they cannot be flatfootedly defined in terms of the higher-order quantifiers. We could instead add these notions to a higher-order language and subject them to axioms that constrain their behaviour in the ways we'd expect given the analogy with diagrams and languages. In this section we consider doing exactly this, focusing on two possible further primitives mentioned above:

$A_1, \ldots, A_n \rhd C$ means C is 'metaphysically' definable from $A_1, \ldots, A_n$.

$\mathrm{Pure}_\sigma\ Q$ meaning that Q is a 'constituentless entity' (or, alterantively, being metaphysically definable from nothing).

We will end the section by mentioning some further primitives.

Let's begin with the notion of purity. Consider a λ-language $\mathcal{J}(\Sigma)$ whose signature contains, for each type σ, a predicate $\mathrm{Pure}_\sigma : \sigma \to t$, interpreted in the way outlined. What sort of object language principles should it be governed by? Of course, we have argued above that every combinator is pure, thus we should have an axiom stating that combinators are pure. What combinators we have depends on the language—below we use the expression '$\mathcal{J}$-combinator' for a closed term of $\mathcal{J}(\emptyset)$:

Pure Combinators $\mathrm{Pure}_\sigma\ Q$ whenever Q is a $\mathcal{J}$-combinator.

Presumably combinations of constituentless entities will also be constituentless. The most straightforward way to combine two entities is by application, thus we could posit that purity is closed under application

Pure Application $\forall Xy.(\mathrm{Pure}_{\sigma\to\tau}\ X \wedge \mathrm{Pure}_\sigma\ y \to \mathrm{Pure}_\tau(Xy))$

Observe that in the special case where $\mathcal{J}(\Sigma) = \mathcal{L}(\Sigma)$, the full λ-language, then Pure Combinators can be replaced with a more limited collection of its instances and Pure Application. Pure Combinators is derivable from the claim that $S^{\sigma\tau\rho}$ and $K^{\sigma\tau}$ are pure at every type, and Pure Application.

We have previously noted that application is an arbitrary special case of a wider class of ways of combining entities: complication. Provided $\lambda XY.(XY)^n_m$ is a $\mathcal{J}$-combinator, anything that can be made by complication can be made using combinators and application, since we can apply $\lambda XY.(XY)^n_m$ to R and S to make $(RS)^n_m$. However, if the complication combinator is not part of the language (but individual complications are) we would need a separate stipulation that purity is closed under complication:

Pure Complication $\forall Xy.(\mathrm{Pure}_\sigma\ X \wedge \mathrm{Pure}_\tau\ y \to \mathrm{Pure}_\rho(XY)^n_m)$

Here σ, τ and ρ must be replaced by types for which the above sentence is well-typed. Clearly we should impose further axioms stipulating the closure of purity under other modes considered in Chapter 12, when the λ-language is closed under those modes of combination but is lacking the corresponding combinator. For instance, converses, reflexizations and vacuous abstractions of pure entities are also pure.

Are there any further axioms governing purity? We might consider the view that combinators are the only constituentless entities. If we were working in a logically perfect language,

where the constants denote distinct metaphysically simple entities, then terms containing constants will not denote pure entities. So we could capture this thought with the following schema:

Only Pure Combinators $\neg \mathrm{Pure}_{\sigma}\, M$ whenever M is a closed term in β-normal form containing constants.

Note that in languages with vacuous λ-abstraction we need to stipulate that M cannot be β-reduced. For otherwise we would contradict Pure Combinators: for any combinator Q, an instance of Only Pure Combinators is $\neg \mathrm{Pure}(\lambda x.Q)c$ where c is a constant with the same type as x, and given β this implies $\neg \mathrm{Pure}\, Q$ contradicting Pure Combinators. (This stipulation is unnecessary in languages that ban vacuous λ-abstraction, such as the relevant λ-language.)

One might, however, want to explore views which allow for a wider class of constituentless entities. Above we explored the view, held by Russell and Wittgenstein, that logical words like 'if' and 'or' do not contribute constituents to the propositions they express, and thus should be treated as primitive logical modes of combination. Just as we made the move from positing a pure entity for each structural mode of combination, there is an analogous move positing a pure entity for each logical mode of combination. Thus we can consider axioms stating the purity of each logical operation:

Pure Conjunction $\mathrm{Pure}_{(t\to t)\to t} \wedge$

Pure Quantification $\mathrm{Pure}_{(\sigma\to t)\to t} \forall_{\sigma}$

$\vdots$

Once we have contemplated expanding our conception of purity to logical connectives, why stop there? Perhaps we could also include mereological primitives, geometrical primitives, or mathematical primitives as pure. We have been focused on the view that propositions, properties and relations can be decomposed into simple or fundamental properties and relations, but it's also common to suppose that individuals can be decomposed into more basic individuals and that relationships of relative fundamentality hold between medium sized objects and the objects of physics. According to one version of this idea, the mereological atoms are fundamental entities, and ordinary objects are less fundamental entities that are built from them by mereological fusion. Once we have acknowledged fusion as a mode of combination for individuals we could go further and posit a fusion operation which takes some individuals and creates another individual but does not itself contribute constituents to the resulting individual. Another common thought is that sets are less fundamental than their members, in which case set collection might be another candidate for being a pure operation.

Let's now turn to the notion of an entity C being 'definable' from some other entities $A_1 \ldots A_n$, which we notated $A_1, \ldots, A_n \rhd_{\sigma_1 \ldots \sigma_n \tau} C$. To distinguish it from the linguistic notion we will call this notion *metaphysical definability*. We can use the linguistic concept of definability as heuristic however: a closed term M involving constants $c_1, \ldots, c_n$ is defined from those constants. As mentioned previously the notion of purity lets us theorize directly about how entities are constructed from more basic entities in the object language. In this case M can be decomposed into a combinator—obtained by replacing the constants in M with variables and λ-abstracting them $\lambda x_1 \ldots x_n.M[x_1/c_1, \ldots, x_n/c_n]$—applied to the constants $c_1, \ldots, c_n$. The combinator $\lambda x_1 \ldots x_n.M[x_1/c_1, \ldots, x_n/c_n]$ represents the way in which $c_1, \ldots, c_n$ are combined to make M. Thus we might attempt to define metaphysical

definability from purity as follows:

$$\rhd_{\bar{\sigma}\tau} := \lambda \bar{x} y \exists_{\bar{\sigma}\to\tau} X(\mathrm{Pure}\, X \wedge X\bar{x} = y)$$

But the above definition of metaphysical definability in terms of purity is not available to metaphysical views that reject the existence of the relevant combinator, $\lambda x_1 \ldots x_n.M[x_1/c_1, \ldots, x_n/c_n]$. For views that reject the existence of pure entities altogether the above relation never holds between any entities, yet such views might still find the notion of metaphysical definability intelligible. It makes perfect sense to ask what entities can be built, say, from R, a and $\exists_e$, on the original ordered structured view, and there is a definite answer: only two things may be built $\exists_e(Ra)$ and $\exists_e \lambda x.Rxa$. Other structured views that admit the intelligibility of more λ-terms might accept further entities as answers. Thus structuralists rejecting the existence of pure entities ought to take metaphysical definability as primitive and not attempt to define it in terms of purity. Indeed, given metaphysical definability one can recover the notion of purity as follows:

$$\mathrm{Pure}_\sigma := \lambda x.\, \rhd_\sigma x$$

that is to say, to be pure is to be metaphysically definable from nothing.

To be sufficiently neutral about the existence of combinators and other pure entities we shall again work in a typed language $\mathcal{J}(\Sigma)$ in a suitable higher-order signature for that language. For each $\sigma_1, \ldots, \sigma_n$ and σ_{n+1} we can introduce constants expressing metaphysical definability:

$$\rhd_{\sigma_1 \ldots \sigma_{n+1}} : \sigma_1 \to \cdots \to \sigma_n \to \sigma_{n+1} \to t$$

Given terms $A_1 : \sigma_1, \ldots, A_n : \sigma_n$ and $B : \sigma_{n+1}$ we write (employing a form of infix notation): $A_1 \ldots A_n \rhd_{\sigma_1 \ldots \sigma_{n+1}} B$ to mean that B is metaphysically definable from $A_1, \ldots, A_n$. The type subscripts for $\rhd$ can be inferred from the types of $A_1, \ldots, A_n$ and B, and so we omit them when no ambiguity arises. It is often helpful to use the capital Greek letters, Γ and Δ, as short for sequences $A_1, \ldots, A_n$ of terms of type $\sigma_1, \ldots, \sigma_n$, and $\Gamma \rhd B$ for the corresponding metaphysical definability claims.

Here are the core principles governing metaphysical definability:

Reflexivity $A \rhd A$

Cut $(\Gamma \rhd A) \wedge (\Delta, A, \Delta' \rhd B) \to (\Delta, \Gamma, \Delta' \rhd B)$

Weakening $\Gamma, \Delta \rhd B \to \Gamma, A, \Delta \rhd B$

Exchange $\Gamma, A, B, \Delta \rhd C \to \Gamma, B, A, \Delta \rhd C$

Contraction $\Gamma, A, A, \Delta \rhd B \to \Gamma, A, \Delta \rhd B$

Reflexivity states the self-evident fact that every entity can be defined at least from itself. It should be emphasized that this is not incompatible with the possibility, raised in Section 12.3, that there are pure operations that can be defined from nothing (such as combinators): it just follows that pure operations can both be defined from nothing and from themselves. Cut captures a generalized form of transitivity for $\rhd$. If you can define A from some things, and you can define B from some other things along with A, then you can define B from those other things along with the things you need to define A. Weakening, Exchange and Contraction are built into the standard notion of definability: if an expression can be defined from a set of terms whose elements are represented by a sequence Γ, then that means you can

build the expression freely drawing elements from Γ. Thus if something can be defined from elements in Γ it can be defined from any superset of Γ (Weakening), you can use any elements in Γ as many times as you like (Contraction), and the order of the sequence is irrelevant (Exchange). We later will consider some different primitives—exhaustive definability, exact definability and ordered definability which may invalidate these principles.

Exercise 13.1 (Generalized Transitivity)**.** *Suppose that* $\Gamma_1 \rhd A_1, \ldots, \Gamma_n \rhd A_n$ *and* $A_1 \ldots A_n \rhd B$. *Show, using Cut, that* $\Gamma_1, \ldots, \Gamma_n \rhd B$.

In addition to the structural rules we can also consider further principles of metaphysical definability corresponding to each of the ways of combining entities to make new entities discussed previously. Converses, Reflexizations and Vacuity, of course, can only be formulated when $\mathcal{J}(\Sigma)$ contains the relevant λ-terms.

Application $(\Gamma \rhd M) \wedge (\Delta \rhd N) \rightarrow (\Gamma, \Delta \rhd MN)$

Complication $(\Gamma \rhd M) \wedge (\Delta \rhd N) \rightarrow (\Gamma, \Delta \rhd (MN)^n_m)$

Converses $(\Gamma \rhd M) \rightarrow (\Gamma \rhd \lambda \bar{z}xy\bar{w}.M\bar{z}yx\bar{w})$

Reflexizations $(\Gamma \rhd M) \rightarrow (\Gamma \rhd \lambda \bar{x}y\bar{z}.M\bar{x}yy\bar{z})$

Vacuity $(\Gamma \rhd M) \rightarrow (\Gamma \rhd \lambda \bar{x}y\bar{z}.M\bar{x}\bar{z})$

Conjunction $(\Gamma \rhd A) \wedge (\Delta \rhd B) \rightarrow (\Gamma, \Delta \rhd A \wedge B)$

Disjunction $(\Gamma \rhd A) \wedge (\Delta \rhd B) \rightarrow (\Gamma, \Delta \rhd A \vee B)$

Negation $(\Gamma \rhd A) \rightarrow (\Gamma \rhd \neg A)$

Universal $(\Gamma \rhd F) \rightarrow (\Gamma \rhd \forall_\sigma F)$

Existential $(\Gamma \rhd F) \rightarrow (\Gamma \rhd \exists_\sigma F)$

Thus, for instance, Application encodes the idea that you can build complex things by combining operations with their arguments via application. But application is surely not the only way to combine entities—indeed, it would be a very arbitrary place to stop, since it only allows one to plug entities into the first hole of a relation and not into other holes. The remaining principles formulate the idea that entities are closed under various other modes of combination considered in this chapter, and which can be adopted or rejected. Note that, for instance, Conjunction follows from Pure Conjunction, Application and the core principles of metaphysical definability, but is a strictly weaker principle: one could hold conjunctions of propositions are metaphysically definable from their conjuncts without maintaining that the operation of conjunction is metaphysically definable from nothing (indeed, one cannot even prove that the operation of conjunction exists if it is treated syncategorematically in $\mathcal{J}(\Sigma)$).

Comprehension Check 13.1. *Show that Conjunction follows from Pure Conjunction, Application and the core principles of metaphysical definability along with our definition of purity from metaphysical definability.*

If we are working in a logically perfect language, $\mathcal{J}(\Sigma)$, which contains only λ-terms that correspond to legitimate modes of combination, we could summarize the relevant rules by having a single schematic principle:

$x_1, \ldots, x_n \rhd A$ provided $FV(A) \subseteq \{x_1, \ldots, x_n\}$

since the syncategorematic rules of $\mathcal{J}(\Sigma)$ correspond to exactly the legitimate modes of combination.

Exercise 13.2. *Using Reflexivity, Cut and Application derive:*

a. $\neg, R, a, b \rhd \neg(Rab)$

b. $(\Gamma, Fa, \Delta \rhd B) \rightarrow (\Gamma, F, a, \Delta \rhd B)$

c. Assuming also β *show:* $(R \rhd \lambda xy.Ryx) \rightarrow (R, b, a \rhd Rab)$

Exercise 13.3. *Using Reflexivity, Complication and* $\beta\eta$ *show*

a. $\neg, \Box \rhd \lambda p.\neg(\Box p)$

b. $\wedge, F, G \rhd \lambda xy.(Fx \wedge Gy)$ *(Rewrite this in prefix notation first.)*

c. $A, B, C \rhd \lambda xyzw.Ax(By(Cz)w)$

d. $A, B, C \rhd \lambda xyzw.Ax(By)z(Cw)$

Exercise 13.4. *Show that* $\wedge, F, G \rhd \lambda x.(Fx \wedge Gx)$ *using Reflexivity, Cut, Complication and Reflexizations.*

As mentioned previously purity can be defined from $\rhd$:

Definition 13.1 (Purity)**.** *C is pure iff it can be built from nothing. We can introduce it as a predicate in the object language by definition:*

$$\mathrm{Pure}_\tau := \lambda X.\, \rhd_\tau X$$

As usual, we let context determine the subscripts.

The simplest way to ensure that there are pure entities is to posit the purity of the identity combinator:

Identity $\rhd \lambda X.X$

Exercise 13.5. *In this exercise you will see that Identity is sufficient to construct all the combinators, provided you can form reflexizations, converses and add vacuous argument places.*[i]

a. Show that the $W^{\sigma\tau}$ *combinator is pure from Identity and Reflexizations:* $\rhd W^{\sigma\tau}$

b. Show that the $C^{\sigma\tau\rho}$ *combinator is pure from Identity and Converses:* $\rhd C^{\sigma\tau\rho}$

c. Show that the $K^{\sigma\tau}$ *combinator is pure from Identity and Vacuity:* $\rhd K^{\sigma\tau}$

Exercise 13.6.

a. Show that if P is pure and $PA_1 \ldots A_n = B$ *then* $A_1, \ldots, A_n \rhd B$ *using Application.*

b. Work in the full λ*-language in a signature containing a well-foundedness property* $W : (e \rightarrow e \rightarrow t) \rightarrow t$*. Prove that converse-well-foundedness is metaphysically definable from well-foundedness from the assumption that all combinators are pure.*

Exercise 13.7. *For each of the following pairs of entities say which can be metaphysically defined from the other, explaining why. If they can both be metaphysically defined from each other say so. You may assume we are in the full* λ*-language, and you may assume Pure Combinators and Pure Application*

a. A binary relation and its reflexization.

b. A binary relation and its converse.

c. A proposition P and the vacuous property $\lambda x.P : e \rightarrow t$*.*

d. A proposition P and the vacuous operator property $\lambda X.P : (t \rightarrow t) \rightarrow t$*.*

So far we have drawn heavily on the linguistic model of metaphysical definability where we think of the metaphysically defined entity as being constructed piece by piece from other entities. However one does not have to adhere to that model when employing the notion. Suppose that we subscribe to the view that the fundamental entities are those given to us by physics, and that all other entities are metaphysically definable from the fundamental. Is it really possible to define *being a table* from fundamental properties like *being an electron*, *mass* and so on, using the sorts of operations we have discussed in this chapter? Perhaps it is, but it is far from obvious. It's natural, at any rate, to explore more permissive notions of metaphysical definability, which we give a model theoretic characterization later in the book. The following examples from geometry illustrates the distinction.

Example 13.1 (Euclidean geometry)**.** *Tarski showed that Euclidean geometry could be characterized by two primitives: a three-place relation of betweenness* $B : e \to e \to e \to t$*, where* $Bxyz$ *says* x, y *and* z *are points, and* x *lies on a straight line between* y *and* z*, and a four-place relation of congruence,* $C : e \to e \to e \to e \to t$*, where* $Cxyzw$ *says that* x, y, z *and* w *are points, and the line* xy *is the same length as* zw*. From the congruence primitive it is possible to define the higher-order property of being a property that applies to the points on a circle:*

$$\mathrm{Circ} := \lambda X.\exists xy \forall z(Xz \leftrightarrow Cxzxy)$$

Thus, assuming for simplicity the purity of the logical connectives and first-order quantifiers, being a circle is metaphysically definable from congruence: $C \rhd \mathrm{Circ}$ *(the pure operation witnessing this is the result of* λ*ing out the* C *from the above definition). Circles are very simple shapes. Having the same geometric shape as Great Britain, by contrast, is very complicated. Assuming its coastline is infinitely detailed, we could not come up with a finite definition of it in higher-order logic from congruence and betweenness. Nonetheless it may still be metaphysically definable from congruence and betweenness: our conception of purity and metaphysical definability was inspired by the linguistic notion of definability, but it can be useful to assume that it outstrips it.*

In Section 17.2, we will develop a model theoretic account of metaphysical definability in terms of 'metaphysical substitutions' (mentioned also in Section 8.3). These are certain functions that act on the 'domains' of each type and satisfy certain conditions for being well-behaved. The claim $x_1, \ldots, x_n \rhd y$ gets interpreted as y being fixed by every metaphysical substitution that fixes $x_1, \ldots, x_n$. This has straightforward meaning on the linguistic model: a substitution of a language that fixes expressions $A_1, \ldots, A_n$ will also fix any expression B that is made from $A_1, \ldots, A_n$ using only the syncategorematic rules of the language. In the geometric case, by contrast, a metaphysical substitution fixing the congruence and betweenness relations will essentially be a Euclidean map: a mapping on points of Euclidean space that preserves congruence and betweenness.[1] Euclidean maps preserve arbitrary shapes, such as the shape of Great Britain, corroborating the idea that arbitrary shape properties can be metaphysically defined from the metaphysicaly basic notions of congruence and betweenness.

Example 13.2 (Kantian geometry)**.** *Kant famously argued that there's a difference between a left hand in an otherwise empty world, and a right hand. Being a left-hand-shaped, however is not metaphysically definable from the Euclidean primitives of congruence and betweenness. The Kantian must assume a further primitive in addition to congruence and betweenness,* $Lxyzw$*, meaning that* x, y, z *and* w *are distinct points, and the point* x *lies on the left side of the plane defined by* yzw*.*[2] *Being a left hand is metaphysically definable from* C, B and L *but not* C

and B alone. We can see that being a left hand is not metaphysically definable from C and B using the above analysis in terms of metaphysical substitutions: a reflection about some plane maps points of space to points of space, and is a Euclidean map in the sense that it preserves congruence and betweenness, but it can map a left hand to a right hand. Thus not every metaphysical substitution fixing congruence and betweenness fixes being a left hand.

Our survey here has been far from exhaustive. There are many other primitives that could be useful. Below I briefly survey some further possible primitives.

$\mathrm{Fun}_\sigma\ A$ means that A is a fundamental or simple entity of type σ, and is not metaphysically definable from anything of any type strictly simpler than it.

$A \blacktriangleright C$ meaning that A is fundamental and C requires A—it is not possible to define C from some fundamental entities that do not include A.

$A_1, \ldots, A_n \lhd C$ meaning C is metaphysically definable from entities different from $A_1, \ldots, A_n$

In each of the above cases the naïve way to try to define these notions from metaphysical definability involves trying to quantify over entities from all types at once and is therefore illegitimate. Nonetheless, one might take them to be intelligible and so could be taken as primitives instead.

We have also focused on a fairly coarse-grained notion of definability. Like the linguistic notion of definability, C is metaphysically definable from $A_1, \ldots, A_n$ if we can build C from entities appearing in that list, using each entity as many times as we like, and not necessarily using all the entities. For theories of granularity in which one can make sense of the notion of constituent, constituent number or constituent order, more fine-grained primitives could be introduced. For instance, we could have the following:

$A_1, \ldots, A_n \rhd^* C$ meaning it is possible build C from $A_1, \ldots, A_n$ using each entity at least once (and possibly more than once).

$A_1, \ldots, A_n \rhd^! C$ meaning it is possible build C from $A_1, \ldots, A_n$ using each of $A_1, \ldots, A_n$ exactly once.

$A_1, \ldots, A_n \rhd^< C$ meaning it is possible to build C from $A_1, \ldots, A_n$ by successively combining adjacent elements of $A_1, \ldots, A_n$.

The first notion, $\rhd^*$, is exhaustive definability. While 'bachelor' may be defined from the words in the English dictionary, it is only exhaustively defined by the words 'married', 'and', 'not' and 'man'. Similarly, while you can build, in the inexhaustive sense, *John is tall* from *John*, *is tall*, and *Mary*, we have considered views on which it is only exhaustively built from the first two.[3] The principles of Reflexivity, Cut, Exchange and Contraction are sound for exhaustive definability but Weakening is not. Note too that the distinction between exhaustive and inexhaustive definability is not available to everyone. Some metaphysical views allow for 'forgetful' operations, like $\lambda x.F$ where x is not free in F. According to these views—views that accept Vacuity—the difference between exhaustive and inexhaustive definability is lost.

Exercise 13.8. *Show that if* $\Delta \rhd^* C$ *then* $\Delta, A \rhd^* C$ *using Vacuity, Reflexivity, Application and Cut for* $\rhd^*$.

The second primitive, $\rhd^!$, we will call exact definability, and is also potentially sensitive to the number of occurrences of a constituent.[4] If a lego spaceship can be built from a lego pack using all of the pieces, then the pack must contain the right number of instances of a single lego brick type to make the spaceship. Thus the sequence *John*, *loves*, *John* exactly defines *John loves John*, but the sequence *John*, *loves* might not (although it automatically exhaustively defines it). We have considered views in which complex entities are a bit like our earlier notion of lego constructs: just as you might need two instances of a single type of brick to build a given construct, you might need two *John* occurrences to build the above proposition. For exact definability, Reflexivity, Cut and Exchange are valid, whereas Weakening and Contraction may not be. Again, some views will not be able to distinguish this primitive from ordinary and exhaustive metaphysical definability, specifically views that posit reflexized properties like *loves oneself*.

Exercise 13.9. *Prove* $R, a \rhd^! Raa$ *using Application, Reflexization for* $\rhd^!$ *and* β *but* without *using Contraction.*

The last notion, ordered definability, also takes order into account. Intuitively you can order define B from $A_1, \ldots, A_n$ if it is possible to repeatedly combine adjacent elements of the sequence $A_1, \ldots, A_n$ to finally obtain B (that is, combining adjacent elements in $A_1, \ldots, A_n$ will yield a new sequence of elements which is one shorter; this can be repeated until we reach a sequence of length one). Thus the sequence $\neg, R, a, b$ order defines the proposition $(\neg((Ra)b))$. We illustrate how this is built with the following 'derivation':

$$\dfrac{\dfrac{\dfrac{\neg, R, a, b}{\neg, (Ra), b}}{\neg, ((Ra)b)}}{(\neg((Ra)b))}$$

Note that on this interpretation Reflexivity and Cut seem still to be sound, but Weakening, Contraction and Exchange are not. That Cut is sound requires justification: suppose I can make A by successively combining adjacent elements in Γ, and I can make B by successively combining adjacent elements of Δ, A, Δ'. Then using the first assumption I can successively combine adjacent elements in Δ, Γ, Δ' to get Δ, A, Δ' and then using the second assumption I can successively combine adjacent elements of these until I get B. We could also imagine even more restricted notions of definability taking order into account: one that requires us to construct the defined entity by repeatedly combining the leftmost pair of entities in a sequence, and repeating this until we reach a singleton sequence, and another primitive that only lets you combine the rightmost pair of entities. According to these primitives the sequence $\neg, R, a, b$ does not seem to define the proposition $(\neg((Ra)b))$, assuming the only mode of combination is complication. Note that Cut is no longer sound on these interpretations.

Again, metaphysicians who are happy with converse relations will not necessarily distinguish this primitive from the previous ones: the sequence *John*, *loves*, *Mary* lets one build *John*, *is loved by*, *Mary*, from which we can build *John is loved by Mary*.

Exercise 13.10. *Show* $R, b, a \rhd^< Rab$ *using Converses, Cut, Application and Reflexivity but* not *Exchange.*

$$\rhd_{\bar{\sigma}\tau} := \lambda\bar{x}y\exists_{\bar{\sigma}\to\tau}X(\mathrm{Pure}\, X \wedge X\bar{x} = y)$$

13.2 A general logical framework

In this section, we will provide a general framework for generating logical characterizations of structural views, including the views explored in Chapters 11 and 12. Although we will frequently prove metalinguistic claims about these theories—for instance, that they contain certain consequences, are consistent or inconsistent, and so on—these axiomatizations will generally involve infinitary rules, and are not designed to be used for *doing* higher-order metaphysics directly. The idea, rather, is that the theories describe a particular metaphysical worldview and these metalinguistic claims give us insight into what it would be like to adopt that worldview: for instance, is it a coherent picture? What sentences should someone accepting the picture adopt? We will also select certain principles (see Only Logical Circles, Separated Structure and Logical Possibility$^{\mathbf{L}}$ below) that belong to these theories which could be taken as bases for doing higher-order metaphysics directly.

Let's begin with the simplest diagrammatic structural view described in Section 11.3. The view was put informally as one in which the structure of propositions, properties and relations is accurately represented by relational diagrams. This claim can be split up into two thoughts: (i) that different relational diagrams correspond to different entities, and (ii) that every entity can be represented by a relational diagram (since every entity can be built from the simple entities using complication so that every entity).

In Section 11.4, we described a correspondence between the structural terms—those terms typable in the Curry system $\mathbf{ND}[]$—and relational diagrams, and we further argued that $\beta\eta$ was sound and complete with respect to diagrammatic reasoning, in the sense that two λ-terms represented the same diagram iff those terms were $\beta\eta$-equivalent. So to formulate this view as a logical theory it seems natural to work in the structural language, call it $\mathcal{S}(\Sigma)$, where the signature Σ contains all the logical constants including the quantifiers needed to simulate binding variables in non-structural ways.[5] The correspondence we described associated constants with simple diagrams. Thus we should assume that $\mathcal{S}(\Sigma)$ is a logically perfect language. The constants in Σ denote distinct simple entities, and every simple entity is expressed by some constant.[6] We can use the fact that $\beta\eta$ corresponds with sameness of relational diagrams to formulate the idea that distinct relational diagrams represent distinct entities:

$\beta\eta$**-Identity** $A =_\sigma B$ when $A, B : \sigma$ are closed terms that are $\beta\eta$-equivalent.

$\beta\eta$**-Distinctness** $A \neq_\sigma B$ when $A, B : \sigma$ are closed terms that are not $\beta\eta$-equivalent.

We can also characterize the idea that every entity corresponds to some relational diagram using the fact that relational diagrams correspond exactly to the structural terms in our logically perfect structural language, $\mathcal{S}(\Sigma)$. Suppose $F : \sigma \to t$ is a unary predicate, and Fd is true of every closed term $d : \sigma$ of $\mathcal{S}(\Sigma)$, then $\forall_\sigma F$ must be true, since every entity of type σ is denoted by a term. Thus we can capture the idea that a logically perfectly structural language denotes every entity using an infinitary rule like this. To state this more generally, we need to rephrase it into a form that is consistent with the general quantifiers required by the structural view to bind in non-structural ways (recall Section 9.4):

$\mathcal{S}(\Sigma)$**-Completeness** $F[d_1/a], F[d_2/a], \cdots \vdash \forall_{\boldsymbol{\sigma}} F\check{a}$

here $[d/a]$ refers to the result of substituting all the hatted elements of $a : \boldsymbol{\sigma}$ with d, and $d_1, d_2, \ldots$ range over all the closed terms of type σ in $\mathcal{S}(\Sigma)$. These axioms and the completeness rule can be added to the background logical principles suitable for the structural

language. Observe in the above that $\beta\eta$-Identity is redundant since we have $\beta\eta$, but I include it so this axiomatization may be generalized straightforwardly to different contexts.

This treatment of the particular structured view can be generalized to other structured views. For each of the structural views we've examined in Sections 11.3–12.3, we have described a general λ-language which we've claimed contains all and only expressions that denote entities that can be built using the modes of combination that are legitimate according to that structuralist. The view that extends the simple structural view by positing converses and reflexizations can use the structural language generated by $\mathbf{ND}[PC]$; the view that posits primitive logical modes of combination can work in the language generated by $\mathbf{ND}[L]$, and so on. Second, we have a notion of equivalence between expressions of the relevant language that tells us what it means for those expressions to be two instructions for building the same structured entity. We have seen that the equational theory of $\beta\eta$-equivalence is the proper logic for structured views based on the picture of propositions with holes that can be filled with other entities. We might, however, generalize further by allowing for views that operate with a stronger theory of equivalence, which can be represented by a logic $\mathbf{L}$, which may or may not extend classical higher-order logic. Finally, we also considered views that allowed for pure entities. Combinators are paradigm examples of pure entities, and such views would thus naturally be formulated in general λ-languages that have combinators, but we also explored the possibility that logical connectives, and possibly further operations, were pure. Thus we can also generalize by picking a pair of signatures Π and Σ, where Π intuitively is the signature containing pure constituentless entities, and Σ is a signature containing a unique constant for every metaphysically simple entity. We should also require the logic $\mathbf{L}$ to coher properly with the signature: $\mathbf{L}$ should be closed under the rule of substitution for constants in Σ, but not the pure constants in Π. Thus, we seem to have a structured view corresponding to every choice of:

1. Signature Σ representing the metaphysically simple entities, with exactly one constant for each simple entity.
2. Signature Π representing the pure constituentless entities.
3. General λ-language $\mathcal{J}(\Pi\cup\Sigma)$, containing all and only terms representing the entities that can be created using the simple entities using legitimate modes of combination.
4. Logic $\mathbf{L}$, closed under the rule of substitution for constants in Σ, representing when two terms correspond to the same entity in reality.

We will call the corresponding structured view $\mathcal{J}(\Pi \cup \Sigma)/\mathbf{L}$ structuralism since, putting in informally, it corresponds to the view that reality is structured like the language $\mathcal{J}(\Pi \cup \Sigma)$ quotiented by the identities in $\mathbf{L}$. We assume that all of the logical constants necessary for the language in question belong to $\Pi \cup \Sigma$, but we do not assume that any particular logical constant must be pure or impure. Thus the structured picture of Section 11.3, relative to the assumption that Σ enumerates all the metaphysically simple entities, is $\mathcal{S}(\Sigma)/\beta\eta$. Π is empty since there are no constituentless entities. $\mathcal{J}(\Pi\cup\Sigma)/\mathbf{L}$ structuralism can be axiomatized with the analogues of the above rules.

L-Identity $A =_\sigma B$ when $A, B : \sigma$ are closed terms and $A =_\sigma B \in \mathbf{L}$.

L-Distinctness $A \neq_\sigma B$ when $A, B : \sigma$ are closed terms and $A =_\sigma B \notin \mathbf{L}$.

$\mathcal{J}(\Pi \cup \Sigma)$**-Completeness** $F[d_1/a], F[d_2/a], \cdots \vdash \forall_{\boldsymbol{\sigma}} F\check{a}$.

Where $d_1, d_2, \ldots$ enumerate all the closed terms in $\mathcal{J}(\Pi \cup \Sigma)$.[7] Henceforth we will use $\mathcal{J}(\Pi \cup \Sigma)/\mathbf{L}$-structuralism to refer to the system governed by the above axioms and rules.

Definition 13.2 ($\mathcal{J}(\Pi \cup \Sigma)/\mathbf{L}$-structuralism)**.** *$\mathcal{J}(\Pi \cup \Sigma)/\mathbf{L}$-structuralism is the theory you get by taking the logical axioms of* H *(as formulated using the appropriate logical signature for $\mathcal{J}(\Pi \cup \Sigma)$), all instances of* **L***-Identity and* **L***-Distinctness, and closing under the rules of* H *plus the rule of $\mathcal{J}(\Pi \cup \Sigma)$-Completeness.*

So far we have only discussed structural views where **L** is the minimal theory consisting of identities between $\beta\eta$-equivalent terms, or in the case of the quasi-syntactic view the empty theory. We considered earlier Wittgenstein and Russell's position that logical words don't contribute constituents to the propositions expressed by sentences containing them. In favour of the view that $\neg$ does not stand for anything, Wittgenstein argues that, since p and $\neg\neg p$ are in fact the very same proposition, $\neg$ does not refer to a constituent. For they could not be the same proposition given the assumption that p on its own does not contain $\neg$, and that $\neg$ contributes a constituent to the latter proposition (Wittgenstein does not consider the idea that $\neg$ might stand for a constituentless entity; he is, perhaps, thinking of negation as simply another way of combining propositions like application). Frank Ramsey, another proponent of this viewpoint, discusses Wittgenstein's argument that $\neg$ does not stand for anything and suggests a notation in which to negate a formula you simply turn it upside-down, imposing the identity in virtue of the syntax itself.[8] A similar treatment of negation can of course also be combined with our earlier pictorial representations of propositions. Just as our relational diagrams automatically build in $\beta\eta$, languages and diagrammatic representations that adopt Ramsey's treatment of negation require the quotienting logics **L** to include identities involving logical words, such as $\neg\neg p =_t p$.

Remark 13.1 (Ramsey on Classicism)**.** Observe too that if we adopt Ramsey's proposal and adopt the usual symbols $\vee$ and $\wedge$ for disjunction and conjunction the deMorgan laws can also be built into the notation. In fact, Ramsey gives an interesting argument (echoing Wittgenstein (1922), Section 5.43) that we should identify all logically equivalent propositions:

> I find it very unsatisfactory to be left with no explanation of formal logic except that it is a collection of "necessary facts". The conclusion of a formal inference must, I feel, be in some sense contained in the premisses and not something new; I cannot believe that from one fact, e.g. that a thing is red, it should be possible to infer an infinite number of different facts, such as that it is not not-red, and that it is, both red and not not-red. These, I should say, are simply the same fact expressed by other words (p. 42).

The structuralist who distinguishes two logically equivalent propositions, A and B, must explain why the truth values of these two distinct propositions covary with one another: why do they necessarily have the same truth value? A very simple explanation is available on the assumption that they are the very same proposition.

The demand to explain the necessity of certain truths extends to all of the theorems of H, so Ramsey is effectively providing an argument for Classicism. Indeed, we could take Ramsey's argument to be that any necessary equivalence at all must be an identity. However, we saw in Chapter 7 that once you have Classicism, you can already have the claim that all broadly necessary equivalents are identical (see the derivation of Intensionalism in Section 7.4). Ramsey's idea that the conclusions of propositional entailments must contain the premises can also be cashed out in terms of propositional identities: given Classicism we

can similarly prove that for p to entail q, i.e. $\Box(p \to q)$, just is for $p \wedge q$ and p to be the very same proposition.

Let's now look at some consequences of these axiomatizations. We will begin by exploring a couple of principles that have been proposed in the higher-order metaphysics literature that can be roughly thought of as articulating, respectively, the idea that constituenthood is well-founded, and that entities can be uniquely decomposed into simple entities. Since we have aimed at a very high level of generality in the types of structuralisms we are treating, some of the principles we formulate below will not be well-formed terms of some of the languages we consider. In many cases these can be reformulated easily: for instance, when a quantifier binds variables in non-structural ways we can rephrase the principle using hatted quantifiers or the syncategoramatic quantifiers—all the structural views considered have enough resources to simulate these quantificational claims. On the other hand, sometimes universal quantifiers will bind variables appearing in predicating position which cannot easily be paraphrased, and the closest principle in the vicinity might be a schema.

Let's begin with the idea that an entity cannot be a proper constituent of itself. One might try to formulate this with schemas like the following:

$\forall_{\sigma\to\sigma} X \forall_\sigma y (y \neq_\sigma Xy)$

$\forall_{\sigma\to\tau\to\sigma} X \forall_\sigma y \forall_\tau z (y \neq_\sigma Xyz)$

The first is straightforward enough to motivate given the rejection of pure entities: applying an operation to an argument adds constituents. The second principle is motivated similarly: X and z must contribute constituents to Xyz. Note, however, that if you accept pure entities then neither of these principles can be motivated in the same way: a constituentless operation, such as $(\lambda p.p)$, does not add constituents and indeed in this case we have $\forall_t q.(\lambda p.p)q =_t q$. And even in the second case we can have counterexamples: for instance, $y = (\lambda wX.Xw)y(\lambda p.p)$. However, we *can* say that if $y = Xy$ then X must be constituentless, and similarly if $y = Xyz$ then X and z must be constituentless. Any given sort of structuralist could add purity primitives to their language giving them the expressive resources to state these sorts of things directly. $\mathcal{J}(\Pi \cup \Sigma)/\mathbf{L}$-structuralism takes closed terms in $\mathcal{J}(\Pi)$ to express all and only the pure entities, so the relevant axioms are the straightforward analogues of Pure Combinators and Only Pure Combinators:

Pure Π-terms $\mathrm{Pure}_\sigma\, P$ whenever P is a closed term of $\mathcal{J}(\Pi)$.

Only Pure Π-terms $\neg\,\mathrm{Pure}_\sigma\, M$ when M is closed term in $\beta\eta$-normal form containing at least one constant in Σ.

As always, these are being formulated in a logically perfect language. Note Pure Combinators and Only Pure Combinators are special cases of these schemas when $\Pi = \emptyset$.

The weakening of our first principle is $\forall_{\sigma\to\sigma} X \forall_\sigma y (y =_\sigma Xy \to \mathrm{Pure}_{\sigma\to\sigma}\, X)$. The weakening of the second is a principle that plays a central role in Dorr (2016):

Only Logical Circles $\forall_{\sigma\to\tau\to\sigma} X \forall_\sigma y \forall_\tau z (y =_\sigma Xyz \to \mathrm{Pure}_\tau\, z)$

Observe that in a structural view without any combinators ($\Pi = \emptyset$) Only Pure Π-terms implies $\neg\,\mathrm{Pure}\, z$ for all z, and so both of these principles collapse into our first formulations stating that y is distinct from Xy and from Xyz for all X, y and z of appropriate types.

Dorr formulates his principle in the relevant λ-language, consisting of terms typable in $\mathbf{ND}[IPC]$. This is important, for if we let in vacuous λ-abstraction Only Logical Circles would trivialize the notion of purity:

Exercise 13.11. *Working in a full higher-order λ-language $\mathcal{L}(\Pi \cup \Sigma)$ prove that everything of type σ is pure from Only Logical Circles.*

Exercise 13.12. *Assume that you have all the relevant λ-terms in your language, and you have Pure Combinators.*

a. Suppose that $y =_\sigma Xyz$. Only Logical Circles implies that z is pure. Show that X must also be pure by appealing to another instance of Only Logical Circles.

b. Prove the first principle about the well-foundedness of constituency from Only Logical Circles: if $y = Xy$ then X is pure.

Only Logical Circles is a consequence of many structural views which individuate using $\beta\eta$-equivalence.

Proposition 13.1. *Pure Combinators implies Only Logical Circles given $\mathcal{J}(\Sigma)/\beta\eta$-structuralism provided $\mathcal{J}$ is a sublanguage of the relevant λ-language and contains the sentence Only Logical Circles.*

Proof. By $\mathcal{J}(\Pi \cup \Sigma)$-completeness it suffices to show that for each of the closed terms M, N, and P, $N = MNP \rightarrow \mathrm{Pure}\, P$. When N and MNP are not $\beta\eta$-equivalent $\beta\eta$-Distinctness implies the falsity of the antecedent. When they are $\beta\eta$-equivalent, N and MNP have identical $\beta\eta$-normal forms. Let N' be the normal form of N, so that N' is identical to the normal form of $MN'P$. Observe that provided $x \in FV(A)$ then for any constant c, $A[B/x]$ has at least as many occurrences of c as $(\lambda x.A)B$. And clearly immediate η-reduction preserves number of constants: $\lambda x.Ax$ has exactly as many occurrences of c as A. Thus if P contains any constant c, the $\beta\eta$-normal form of $MN'P$ will have at least one more occurrence of c than N' does, and so cannot be identical to N' after all. So P contains no constants, and is a combinator. It follows that the consequent, $\mathrm{Pure}\, P$, is in the theory by Pure Combinators.

Note that in the case that quantification into predicating position is not allowed in $\mathcal{J}(\Pi \cup \Sigma)$ Only Logical Circles is not well-formed, but a similar argument establishes the schema:

$$\forall_\sigma y \forall_\tau z(y =_\sigma Myz \rightarrow \mathrm{Pure}_\tau\, z)$$

for all closed terms M. □

Let's consider now a principle from Bacon (2020) that restricts Predicate Structure and helps articulate, in a $\beta\eta$-friendly way, the idea that entities can be decomposed uniquely into metaphysically simple entities. As usual we assume it is being formulated in a given logically perfect language $\mathcal{J}(\Pi \cup \Sigma)$:

Separated Structure $Fa =_\tau Ga \rightarrow F =_{\sigma\rightarrow\tau} G$ when $a \in \Sigma^\sigma$, F and G are closed and do not contain a.

Separated Structure is in fact perfectly consistent with β. One type of counterexample to Predicate Structure and Predicate Argument Structure involved the multiple decompositions of *Mary loves Mary*: $Laa = (\lambda xy.Lyx)aa = (\lambda x.Lxx)a = (\lambda x.Laa)a$, yet $La, \lambda y.Lya, \lambda x.Lxx$ and $\lambda x.Laa$ are all distinct properties on account of potentially having different extensions. This contradicts Predicate Structure (and thus Predicate Argument Structure). None of these are counterexamples to Separated Structure, for any pair from the list of identities guaranteed by β, at least one side involves the constant a in the predicate. Unlike Predicate Structure the contants in the predicate and the argument must be completely separate. The other difference is that the argument must be a metaphysically simple

entity. This eliminates another class of counterexamples to Predicate Structure: applying an operator and applying it twice, i.e. $\lambda Xp.Xp$ and $\lambda Xp.X(Xp)$, are clearly different operations, yet when we feed these two operations the identity combinator as an argument we get the same result.

A logic **L** is closed under the ζ rule when the following holds:

If $Mx =_\sigma Nx \in \mathbf{L}$ and $x \notin FV(M) \cup FV(N)$ then $M =_{\sigma\to\tau} N \in \mathbf{L}$.

Separated Structure is also a consequence of our structural views provided **L** is closed under the ζ rule.

Proposition 13.2. *Suppose* **L** *is closed under the* ζ*-rule. Then Separated Structure follows from* $\mathcal{J}(\Sigma)/\mathbf{L}$*-structuralism.*

Proof. Let a be a constant in Σ and let F and G be closed terms not involving a. If $Fa =_\tau Ga$ doesn't belong to **L** then **L**-Distinctness implies $Fa =_\tau Ga \to F =_{\sigma\to\tau} G$, since it implies the falsity of the antecedent. In the other case $Fa =_\tau Ga$ is in **L**. By the rule of substitution for Σ constants we have $Fa[x/a] =_\tau Ga[x/a] \in \mathbf{L}$ and thus that $Fx =_\tau Gx \in \mathbf{L}$ (using the fact that a doesn't appear in F or G). Now $F =_{\sigma\to\tau} G$ is in **L** by the ζ-rule, and so is in $\mathcal{J}(\Sigma)/\mathbf{L}$-structuralism by **L**-Identity. □

Exercise 13.13. *Consider the following quantified variant of Separated Structure:*[9]

Quantified Separated Structure $\forall XY(\mathrm{Pure}\, X \wedge \mathrm{Pure}\, Y \wedge X\bar{c} = Y\bar{c} \to X = Y)$ *where* $\bar{c} = c_1, \ldots, c_n$ *are a sequence of non-repeating constants from* Σ.

This exercise explores this variant principle.

a. Show how every instance of Separated Structure may be proved from this principle and Pure Combinators.

b. Suppose **L** *is closed under the* ζ*-rule. Prove Quantified Separated Structure from Pure* Π*-terms, Only Pure* Π*-terms and* $\mathcal{J}(\Sigma)/\mathbf{L}$*-structuralism.*[ii]

Finally, when **L** extends Classicism $\mathcal{L}(\Pi\cup\Sigma)$-structuralism includes the schema we called Logical Possibility$^{\mathbf{L}}$, which states the possibility of each closed sentence A that is consistent in **L**.

Proposition 13.3. *Suppose* **L** *is a logic extending Classicism that is closed under the* ζ*-rule. Then* $\mathcal{L}(\Pi \cup \Sigma)/\mathbf{L}$*-structuralism contains all instances of Logical Possibility*$^{\mathbf{L}}$.

Exercise 13.14. *Assuming the conditions for Proposition 13.3 show that the schemas* **L***-Distinctness and Logical Possibility*$^{\mathbf{L}}$ *are equivalent.*

Exercise 13.15. *Using Separated Structure in the language* $\mathcal{L}(\Pi\cup\Sigma)$*, prove that if some metaphysically simple relation is the converse of another metaphysically simple relation the all relations are converses of each other. That is, assume* $R =_{e\to e\to t} \lambda xy.Syx$ *when* $R, S \in \Sigma^{e\to e\to t}$ *and show any pair of relations are converses of each other.*[iii]

Now that we have a wide class of theories representing possible structured views on the table, there are many technical questions to be settled concerning which of them are consistent. I will be far from comprehensive here; I will simply offer one limitative result, which may be generalized to several of the theories $\mathcal{J}(\Pi\cup\Sigma)/\mathbf{L}$, and one consistency result which can also be generalized straightforwardly to a few other theories.

Let's begin with the inconsistency result, which targets $\mathcal{L}(\Pi \cup \Sigma)/\beta\eta$-structuralism. This view is in some sense maximally liberal about what λ-terms are meaningful—all of them, including combinators—and maximally conservative about which identities we let in—it is as fine-grained as possible given $\beta\eta$-equivalence. We briefly discussed this view in Section 11.1 where we noted that the pictorial view we have been developing for more limited λ-languages breaks down, so there is already some evidence that this is a view without a proper picture behind it. The inconsistency, unsurprisingly, is just a slightly tweaked version of the Russell-Myhill argument.

Theorem 13.1 (Russell-Myhill)**.** *$\mathcal{L}(\Pi \cup \Sigma)/\beta\eta$-structuralism is inconsistent.*

Proof. Since we have required that $\Pi \cup \Sigma$ contain the logical constants, we have some constants of type $(t \to t) \to t$, for instance $\forall_t$. Now consider the Russell-Myhill term:

$$M := \lambda p.\forall_{t \to t} X(p =_t \forall_t X \to \neg Xp)$$

First we prove $\neg M(\forall_t M)$. Suppose for contradiction that $M(\forall_t M)$. Expanding out the definition of the first M, and instantiating X for $\forall_t M$, we get $\forall_t M =_t \forall_t M \to \neg M(\forall_t M)$, and thus $\neg M(\forall_t M)$ contradicting the assumption.

Thus $\neg M(\forall_t M)$. To obtain a contradiction we now prove $M(\forall_t M)$, or in other words, $\forall_{t \to t} X(\forall_t M =_t \forall_t X \to \neg X(\forall_t M))$. By $\mathcal{L}(\Pi \cup \Sigma)$-completeness it suffices to show that for every closed term N, $\forall_t M =_t \forall_t N \to \neg N(\forall_t M)$. Our theory implies the falsity of the antecedent whenever $\forall_t M$ and $\forall_t N$ are not $\beta\eta$-equivalent, so it suffices to prove $\neg N(\forall_t M)$ when $\forall_t M$ and $\forall_t N$ are $\beta\eta$-equivalent. When $\forall_t M$ and $\forall_t N$ are $\beta\eta$-equivalent they must have an identical normal form by Proposition 3.2. Since we can only $\beta\eta$-reduce inside M and N, clearly this normal form must be of the form $\forall_t T$ for some term T which can be obtained by reducing M until no further reduction is possible, and can also be obtained by reducing N until no further reduction is possible. Thus M and N are $\beta\eta$-equivalent to T and thus to each other. Thus it suffices to simply establish $\neg M(\forall_t M)$, which we did above. So we have a contradiction. □

Observe that the above result is easily adapted to other λ-languages provided they have the resources to formulate the paradoxical Russell-Myhill term M. For instance, all the languages we've given names to that contain any combinators—the relevant, linear, affine and ordered languages—all either contain the term M, or something logically equivalent to it involving quantifiers that simulate binding variables in multiple positions or variables abstracted out of order.

What views are left intact by this result? As we mentioned above, the targeted view is extremal along a couple of dimensions: it accepts the meaningfulness of all λ-terms, but rejects all identities apart from those required by $\beta\eta$. Thus many structural views that restrict or relax along either of these dimensions are left open:

1. General λ-languages that don't have combinators.
2. More liberal λ-languages (say, with combinators) but a stronger notion of equivalence than $\beta\eta$.

The simplest λ-languages to describe that don't have combinators are those given by the substructural Curry typing systems that don't contain Identity. These include the systems $\mathbf{ND}[PCW]$, $\mathbf{ND}[PC]$, $\mathbf{ND}[PW]$, $\mathbf{ND}[CW]$, $\mathbf{ND}[P]$, $\mathbf{ND}[W]$, $\mathbf{ND}[C]$, and $\mathbf{ND}[]$ with the

hatted quantifiers required for expressing quantificational claims, and the eight variants of these systems that treat the quantifiers and connectives as syncategorematic: $\mathbf{ND}[PCWL]$, $\mathbf{ND}[PCL]$, $\mathbf{ND}[PWL]$, $\mathbf{ND}[CWL]$, $\mathbf{ND}[PL]$, $\mathbf{ND}[WL]$, $\mathbf{ND}[CL]$ and $\mathbf{ND}[L]$. In fact we can show all of the structured theories based on the latter sort of language to be consistent. I will show $\mathbf{ND}[L]$. Let $\mathcal{K}(\Sigma)$ denote the language consisting of terms that are typable in $\mathbf{ND}[L]$.

Theorem 13.2. *$\mathcal{K}(\Sigma)/\beta\eta$-structuralism is consistent.*

Proof. To prove this we use the fact (see Proposition 10.2) that every term in $\mathbf{ND}[L]$ has a unique derivation. The idea is to assign an extension to every term M by induction on the length of the proof in $\mathbf{ND}[L]$. We will assign each constant $R : \sigma_1 \to \cdots \to \sigma_k \to t$ an extension that consists of a set of tuples $(N_1, \ldots, N_k)$ of closed terms $N_1 : \sigma_1, \ldots, N_k : \sigma_k$ that are closed under $\beta\eta$-equivalence: if $(N_1, \ldots, N_n)$ is in the set, and N_i and M_i are $\beta\eta$-equivalent for $i = 1, \ldots, n$ then $(M_1, \ldots, M_n)$ is in the set. One then needs to check the trivial condition that this closure condition also holds of the extension complex terms. (An alternative method would be to let the extensions be tuples of equivalence classes of closed terms under the equivalence relation of $\beta\eta$-equivalence.) The extensions are assigned as follows:

- Constants: $[\![\vdash c : \overline{\sigma} \to t]\!]$ an arbitrary set of tuples of terms $(N_1, \ldots, N_n)$ where $N_i : \sigma_i$ $i = 1, \ldots, n$ that are closed under $\beta\eta$-equivalence.
- Concretion: $[\![\Gamma, x : \sigma \vdash Mx : \tau]\!] = [\![\Gamma \vdash M : \sigma \to \tau]\!]$.
- Abstraction: $[\![\Gamma \vdash \lambda x.M : \sigma \to \tau]\!] = [\![\Gamma, x : \sigma \vdash M : \tau]\!]$.
- Application: $[\![\overline{x} : \overline{\sigma}, \overline{y} : \overline{\tau} \vdash MN : \overline{\rho} \to t]\!] = \{\overline{a}\overline{b}\overline{c} \mid \overline{a}N[\overline{b}/\overline{y}]\overline{c} \in [\![\overline{x} : \overline{\sigma} \vdash M : \pi \to \overline{\rho} \to t]\!]\}$ (where $M : \pi \to \overline{\rho} \to t$ $N : \pi$).
- Conjunction: $[\![\Gamma, \Delta \vdash (A \vee B) : t]\!] = \{\overline{a}\overline{b} \mid \overline{a} \in [\![\Gamma \vdash A : t]\!]$ or $\overline{b} \in [\![\Delta \vdash B : t]\!]\}$.
- Negation: $[\![\Gamma \vdash \neg A : t]\!] = [\![\Gamma \vdash A : t]\!]^c$.
- Universal quantification: $[\![\Gamma \setminus \{y_1 \ldots y_n\} \vdash \forall_\sigma x.A[x/y_1 \ldots x/y_n] : t]\!] = \{\overline{a} \mid \overline{a'} \in [\![\Gamma \vdash A : t]\!]$ for every closed term b of type σ and sequence $\overline{a'} \sim_{\Gamma, \overline{y}, b} \overline{a}\}$.

  Here $\overline{a'} \sim_{\Gamma, \overline{y}, b} \overline{a}$ means $\overline{a'}$ and Γ are sequences of the same length, and for each position in these sequences where Γ contains one of the variables in $\overline{y}$, $\overline{a'}$ has b, and $\overline{a}$ is the result of removing these entries from $\overline{a'}$

□

Above we have mainly focused on structural views modeled on the picture of a language quotiented by the theory of $\beta\eta$-equivalence. However, there is no reason why a structuralist couldn't accept further identities. For instance, resuming an earlier line of thought, Wittgenstein accepted identities like $p =_t \neg\neg p$ on the grounds that the negation sign itself does not refer to a propositional constituent, but just signifies the reversal of the sense of a proposition. Indeed, Ramsey's argument against positing distinctions between logically equivalent propositions justifies all identities belonging to Classicism, so a particularly natural theory to quotient by is Classicism, or extensions of Classicism. Many of the above theorems apply even to coarse-grained theories like Classicism: for instance, Classicism is closed under the ζ rule, and so the corresponding structural view contains Separated Structure.[10] We mentioned above that when a logic $\mathbf{L}$ extends Classicism we can derive

every instance of Possibility$^{\mathbf{L}}$ from the corresponding form of structuralism. However, it is presently unknown whether any of these Classicist forms of structuralism are in fact consistent. Thus I end the chapter with what I take to be an important conjecture:

Conjecture 13.1. *$\mathcal{L}(\Pi \cup \Sigma)/\mathbf{L}$-structuralism is consistent in some logic* **L** *extending Classicism, and some signature* Π *and* Σ *where* Π *contains the logical constants* Λ.

13.3 Further reading

The Russell-Myhill paradox was first identified by Bertrand Russell in appendix B of his book "Principles of Mathematics" in 1902, and was later rediscovered by John Myhill in the context of Church's logic of sense and denotation.[11] As we noted in Section 1.3, Russell believed that quantification over properties and relations was really a figure of speech to be replaced in more careful philosophical theorizing by by higher-order generalizations. According to this viewpoint, the paradox Russell is most famous for, involving the property of non-instantiation ("Russell's paradox"), simply dissolves because it cannot be regimented: it is not grammatical for a predicate to be its own subject. By contrast, we saw in Section 11.1 that the Appendix B paradox can be formulated in higher-order logic. Russell apparently took this to suggest that a more complex type system was needed: the ramified theory of types. The ramified theory of types is a much more stringent extension of the commonsense grammatical principles on which we have formalized typed languages in this book. For instance, applying a predicate to an argument always raises its ramified typed so that it is not grammatical to apply an operator to a sentence twice, as in a sentence like $\Box(\Box A)$. The reader interested in learning more about the ramified theory of types, in relation to the Russell-Myhill paradoxes might consult Church (1976) and Hodes (2015). For further discussion of ramification in relation to other paradoxes see Kaplan (1995), Kripke (2011), Tucker and Thomason (2011), and Bacon et al. (2016). Klement (2003) contains a useful overview of the history.

More recently, metaphysicians have become interested in the Russell-Myhill paradox due to its implications for metaphysical theorizing. Uzquiano (2015), Dorr (2016), Fairchild (2017), Goodman (2017) and Fritz (forthcoming a) use the paradox to conclude that reality is not structured in various ways. Fritz (forthcoming a), for instance, uses the Russell-Myhill paradox to draw limitations on the sort of structure needed to make sense of grounding discussed in Chapter 6, and establishes that certain widely used principles of ground are inconsistent.

The topic of converse relations is also one that would seem to be fertile ground for possible applications of higher-order logic. Good places to start on the philosophical side include Williamson (1985), Fine (2000), and Dorr (2004). The view 'positionalism', discussed in Section 12.4, comes from Fine (2000) and has spawned an expansive secondary literature (see, for instance, Dixon (2018), Dixon (forthcoming), Donnelly (2016), and Litland (forthcoming)). Readers interested in formal models of positionalism should consult Leo (2008) and Leo (2010). The use of tools from the λ-calculus, particularly general λ-languages and substructural type systems, seems to hold great potential for applications to this topic, although at present it is somewhat underexplored. The material in Section 12.4 may be seen as a first step in the direction.

The relational diagrams explored in Chapters 11 and 12, and their relation to substructural Curry type systems, expands on the authors earlier work in Bacon (2023).

Endnotes

1. These maps can reflect, rotate, translate, or enlarge/shrink regions. Note the possibility of enlargements means that Euclidean maps are not isometries.
2. That is, when you y at the palm of your left hand, z at the knuckle and w at the fingertip, your thumb will lie in the same side as the plane as x.
3. The ordinary notion of definability in logic tends to be inexhaustive: a predicate is definable from a *set* (as opposed to sequence) of predicates when it's possible to write out a definition of the former using only predicates in the latter.
4. I learned of the usefulness of this notion from Cian Dorr; various principles governing the notion are explored in Dorr (handout).
5. You must also include the full signature of logical constants instead of relying on metalinguistic abbreviations both for the reasons discussed in Section 4.1, and also because the relevant abbreviations are not always available in this restricted language. $\lambda xy.\forall_{e\to t}Z(Zx \to Zy)$ and $=_e$, for instance, are different due to being differently structured, and moreover the former term isn't structural.
6. Here we quickly come up against the limits of what can be said in higher-orderese. The idea that every simple entity is expressed by some constant seems to require quantification at all types at once. This type neutral quantification is eliminable if there were simple entities in at most finitely many different types since one can replace the type neutral quantification with a finite conjunction of generalizations at those different types. But otherwise we should reject this characterization as helpful nonsense—a ladder to be kicked away, as it were—and simply take the notion of a logically perfect language as primitive.
7. If there are uncountably many terms the terms cannot be enumerated, but the rule still stands: the premises simply consist of all the terms of the form $F[d/a]$ wher d is a closed term.
8. Ramsey writes "'not" cannot be a name [. . .] for if it were, "not-not-p" would have to be about the object not and so different in meaning from "p"' Ramsey (1927), pp. 42–43. Similarly Wittgenstein (1922), Section 5.44: 'affirmation can be produced by double negation: in such a case does it follow that in some sense negation is contained in affirmation? [. . .] The proposition '$\neg\neg p$' is not about negation, as if negation were an object'.
9. Bacon (2020).
10. Indeed, any extension of Classicism that is closed under the rule of necessitation for broad necessity will also be closed under the ζ rule, due to the Modalized Functionality principle. Thus, for instance, the result of adding any number of axioms of the form $\Box A$ to Classicism will be closed under the ζ rule.
11. See Russell (1903) and Myhill (1958) respectively.

Hints for exercises

[i] **Hint:** Use η on a suitable instance of the identity combinator.

[ii] **Hint:** By $\mathcal{J}(\Sigma)$-completeness it suffices to show that $\mathrm{Pure}\, P \wedge \mathrm{Pure}\, Q \wedge P\bar{c} = Q\bar{c} \to P = Q$ for every closed term P and Q, and constants $\bar{c} = c_1, \ldots, c_n$ from Σ.

[iii] **Hint:** Use Separated Structure to show $\lambda XY.X = \lambda XYzw.Ywz$, and use this identity to show that any pair of relations T and U are converses of each other.

Part IV

Higher-order model theory

14

Applicative structures

Not all typed languages are logical languages: functional programming languages, for instance, are often modelled as typed λ-languages, yet the terms of these languages cannot be used to make assertions. In this chapter, we will introduce a general class of structures for modelling typed languages, logical or otherwise, called *applicative structures*. The structures have instrumental value and constitute a stepping stone towards our goal of modelling logical languages. However, their study also provides some insight into the inner workings of the meanings of λ-expressions: they provide us with a generalization of the notion of a function.

In Section 14.1, we introduce the notion of an applicative structure, a type of structure that can be used to interpret applicative languages. In Section 14.2 we see how to interpret arbitrary expressions of the full λ-language and combinatory languages in certain applicative structures—those that are 'functional' and 'have combinators'. In Section 14.3 we generalize the notion of an interpretation to non-functional structures and also treat general λ-languages. The remaining sections (Sections 14.4–14.7) explore further algebraic properties and concepts pertaining to the class of applicative structures and interpretations, including congruences, quotients, products, homomorphisms and initial structures. These further topics may be skipped by the reader wishing to move on to models of higher-order logic.

Convention 14.1. *This chapter assumes some familiarity with sets of functions. When $\ldots x \ldots$ is an expression denoting a mathematical object that depends on the value of x, we will use the notation $x \mapsto \ldots x \ldots$ to denote the function that maps an arbitrary element x to $\ldots x \ldots$ (this notation shares many formal features with the λ-notation). If A and B are sets, we will use the notation $A \to B$ to denote the set of all functions with domain A and codomain B.*

Convention 14.2. *In this part of the book, we will encounter mathematical objects with increasing amounts of structure allowing us to model different aspects of typed languages. We will generally use the terms* structure, interpretation *and* model *to indicate how much information these mathematical objects have. We use the word 'structure' to indicate an entity which provides domains of possible interpretations that terms of a typed language could take, and gives meaning to the fundamental operation of application that we find even in the simplest of the languages we have encountered, applicative languages. Interpretations are structures augmented with a function $[\![\cdot]\!]$ which* interprets *a given typed language: assigns to each term of the language an interpretation in the underlying structure. And finally, we reserve the word* model *for an interpretation of a typed language that additionally allows us to evaluate its sentences for truth and falsity.*

DOI: 10.4324/9781003039181-19

14.1 Applicative structures

First, our central definition.

Definition 14.1 (Applicative structure)**.** *An applicative structure,* $\mathbf{A} = (A^{\cdot}, \mathrm{App}^{\cdot})$ *is a family of sets* A^{σ} *and functions* $\mathrm{App}^{\sigma\tau}$ *indexed by types* σ *and* τ *such that:*

- A^{σ} *is a set for each* σ*.*
- $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^{\sigma} \to A^{\tau}$*.*

Informally, A^{σ} should be understood as a set containing potential denotations for terms of type σ. A^{e} thus contains potential meanings for names, $A^{t\to t}$ the potential meanings for operator expressions, and so on. Later, when we model higher-order languages, these sets will additionally serve the role of being the domains of quantification for the quantifiers $\forall_{\sigma}$ and $\exists_{\sigma}$.

$A^{\sigma\to\tau}$ should be thought of as the structures' internal notion of an operation from A^{σ} to A^{τ}. As we'll see, a function from A^{σ} to A^{τ} is a reasonable model of such an operation, and we will consider applicative structures in which $A^{\sigma\to\tau}$ consists of functions from A^{σ} to A^{τ}. But we obtain a significant amount of generality by not making that identification. $\mathrm{App}^{\sigma\tau}$ is the structures' internal notion of operation application, analogous to the operation of function application which takes a function and its argument and produces a value. For instance, if $a \in A^{e}$ serves as an interpretation of a name *Alice* in an applicative structure $\mathbf{A} = (A^{\cdot}, \mathrm{App}^{\cdot})$, and $f \in A^{e\to t}$ the interpretation of the predicate *is tall*, then $\mathrm{App}^{et}(f, a)$ is the interpretation of the sentence *Alice is tall*. There is a different application operation for each type so that, for instance, just as we can apply the interpretation of a predicate to the interpretation of a name, we can also apply the interpretation $q \in A^{(e\to t)\to t}$ of a quantifier expression, such as *someone*, to a predicate, $\mathrm{App}^{(e\to t)t}(q, f)$, yielding the interpretation of *someone is tall*.

We will illustrate the notion of an applicative structure with four concrete examples.

Example 14.1 (Full Henkin structure)**.** *Let* A^{e} *and* A^{t} *be arbitrary sets, and let:*

- $A^{\sigma\to\tau} = A^{\sigma} \to A^{\tau}$ *(the set of all functions with domain* A^{σ} *and codomain* A^{τ}*.*

assuming A^{σ} *and* A^{τ} *have already been defined inductively. Since* $A^{\sigma\to\tau}$ *consists of functions between* A^{σ} *and* A^{τ}*,* $\mathrm{App}^{\sigma\tau}$ *may be defined in terms of function application:*

- $\mathrm{App}^{\sigma\tau}(f, a) = f(a)$*.*

Applicative structures in which $A^{\sigma\to\tau}$ consists of a set of functions from A^{σ} to A^{τ} (not necessarily all of them) have a special name:[1]

Definition 14.2 (Henkin structure)**.** *Any applicative structure in which* $A^{\sigma\to\tau} \subseteq A^{\sigma} \to A^{\tau}$*, and in which* $\mathrm{App}^{\sigma\tau}$ *is defined by function application is called a Henkin structure.*

However, there are many applicative structures that are not Henkin structures. Another important example includes:

Example 14.2 (Term structures)**.** *Let* Σ *be a signature, and* $\mathcal{J}(\Sigma)$ *a typed language over that signature. Then the* closed term structure *is defined as follows:*

- $A^\sigma = \mathcal{J}^\sigma(\Sigma)$, *the set of closed terms of type* σ.
- $\mathrm{App}^{\sigma\tau}(M, N) = (MN)$ *when* $M \in A^{\sigma\to\tau}$ *and* $N \in A^\sigma$.

The open term structure *is defined analogously, except that* A^σ *is identified with the set of open or closed terms of type* σ.

In general, we shall refer to the term structure as $\mathcal{J}(\Sigma)$*, using the same notation we used for a language, and* $\mathcal{J}^\sigma(\Sigma)$ *for the domain* A^σ.

Notice in this example that when $M \in \mathcal{J}^{\sigma\to\tau}(\Sigma)$ and $N \in \mathcal{J}^\sigma(\Sigma)$, M has type $\sigma \to \tau$ and N type σ, so the term (MN) is well-formed and has type τ. Thus it belongs, as required, to $A^\tau = \mathcal{J}^\tau(\Sigma)$. A term M is not a function, and thus App cannot be defined in terms of function application: here it is the operation of prefixing a term to its argument and enclosing in parentheses, as dictated by our term formation rules.

In the following, we will introduce the idea that operations in A^σ may involve 'modes of presentation', individuating operations with functional type more finely than by their applicative behaviour. In this model, we suppose, for instance, that a model of a property is an ordered pair of a function mapping individuals to propositions and a mode of presentation of that function. (For instance, *is a lawyer* might consist of the ordered pair consisting of a function mapping a to the proposition that a is a lawyer, and the mode of presentation 'lawyer', whereas *is an attorney* might consist of the same function with the mode of presentation 'attorney'.) We won't attribute any particular philosophical significance to this example, but it will serve to illustrate important differences between Henkin structures, explored in Exercises 14.1–14.3.[2]

Example 14.3. *For a fixed signature* Σ*, we shall consider* $\mathcal{L}^\sigma(\Sigma)$ *as a set of modes of presentations of entities of type* σ*. Let* X *and* Y *be sets, representing unadorned individuals and propositions respectively. Then*

- $A^e = X \times \mathcal{L}^e(\Sigma)$, $A^t = Y \times \mathcal{L}^t(\Sigma)$.
- $A^{\sigma\to\tau} = (A^\sigma \to A^\tau) \times \mathcal{L}^{\sigma\to\tau}(\Sigma)$.
- $\mathrm{App}^{\sigma\tau}((f, M), (a, N)) = f(a, N)$.

Although it is beyond the scope of this book to investigate these issues further, structures like the above with further restrictions are useful for proving cut-elimination theorems (see Chapter 5 of Brown (2007)).

Definition 14.3 (Possible values structure)**.** *An applicative structure* $\mathbf{A} = (A^\cdot, \mathrm{App}^\cdot)$ *is a possible values structure when:*

- $A^e \subseteq X \times \mathcal{L}^e(\Sigma)$, $A^t \subseteq Y \times \mathcal{L}^t(\Sigma)$ *for some set* X *and* Y.
- $A^{\sigma\to\tau} \subseteq \{(f, M) \mid f : A^\sigma \to A^\tau, M : \sigma \to \tau, \pi_2(f(a, N)) = (MN)$ *for every* $(a, N) \in A^\sigma\}$.

It is common among structured proposition theorists to represent structured entities as ordered tuples of simple constituents. For instance, the proposition that *Sean is tall* might be with the ordered pair *(Sean, tallness)*, treating *Sean* and *tallness* is simple unstructured entities. The simplest way to capture this idea in the present formalism is to identify the result of applying a predicate f to an argument, a, as the ordered pair (f, a), as illustrated in the following example:

Example 14.4. *For each type σ let S^σ be a (possibly empty) set, which we shall informally think of the simple, or unstructured, entities of type σ. We may define an applicative structure by letting $A^\cdot$ be the smallest type-indexed family of sets satisfying the following constraints:*

- $S^\sigma \subseteq A^\sigma$.
- *If* $f \in A^{\sigma\to\tau}$ *and* $a \in A^\sigma$ *then the ordered pair* $(f, a) \in A^\tau$.

and defining application by ordered pair formation:

- $\mathrm{App}^{\sigma\tau}(f, a) = (f, a)$.

To illustrate the abstract and general nature of applicative structures, let us turn to a final example of a structure.

Example 14.5. *For every type σ and τ:*

- $A^\sigma = \mathbb{N}$.
- $\mathrm{App}^{\sigma\tau}(n, m) = n + m$.

(Actually, this example isn't entirely divorced from applications. We might think of this applicative structure as what you get by abstracting away all features of entities except for the number of constituents they contain. Note that when you apply a property with n constituents to an argument with m constituents the result should have $n + m$ constituents, so the definition of application makes sense.)

Let us now turn to some important distinctions, which correspond to two special properties of the full Henkin structures—applicative structures where $A^{\sigma\to\tau}$ is the set of all functions from A^σ to A^τ. In these structures the $\sigma \to \tau$ operations are plenitudinous: there's an operation corresponding to any applicative behaviour. At the same time, functions are governed by a principal of functionality, meaning that there is at most one operation for any applicative behaviour. The internal operations, $A^{\sigma\to\tau}$, of type $\sigma \to \tau$ can fail to behave like the function space from $A^\sigma \to A^\tau$ in each of these two ways. Firstly, it may fail to be *full*, in the sense that there are applicative behaviours that no element of $A^{\sigma\to\tau}$ exhibits, and it may fail to be *functional*, in the sense that two elements of $A^{\sigma\to\tau}$ have the same applicative behaviour.

The notion of an 'applicative behaviour' is clearly not precise, but since functions are plenitudinous and functional, we can simply identify an applicative behaviour with a function. We introduce the following definition, relating elements of $A^{\sigma\to\tau}$ to applicative behaviours in this sense.

Definition 14.4 (Applicative behaviour)**.** *If* $\mathbf{A}$ *is an applicative structure, and* $d \in A^{\sigma\to\tau}$*, the* applicative behaviour *of d is the function $f : A^\sigma \to A^\tau$ such that:*

$$f(a) = \mathrm{App}^{\sigma\tau}(d, a) = f(a) \text{ for every } a \in A^\sigma.$$

We can also talk about the applicative behaviours an element when it is considered as an n-ary operation $d \in A^{\sigma_1\to\cdots\to\sigma_n\to\tau}$. This is the function $f : A^{\sigma_1} \to \cdots \to A^{\sigma_n} \to A^\tau$, defined by:

$f(a_1)(a_2)\ldots(a_n) = \mathrm{App}(\mathrm{App}(\ldots\mathrm{App}(\mathrm{App}(\tilde{f}, a_1), a_2), \ldots), a_n)$ *for every* $a_1 \in A^{\sigma_1}, \ldots, a_n \in A^{\sigma_n}$.

An applicative behaviour, f, is realized *iff some element of the structure has that applicative behaviour.*

A function $f : A^\sigma \to A^\tau$ may be the applicative behaviour of several elements none. In the above, and going forward, we shall often omit superscripts from App, as they are usually clear from context.[3]

Any function $f : A \times B \to C$, can be "curried" to form a function $\bar{f} : A \to (B \to C)$ by setting $\bar{f}(a)(b) = f(a, b)$ for every $a \in A, b \in B$. By currying the application operation $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$ we obtain a function $\overline{\mathrm{App}}^{\sigma\tau} : A^{\sigma\to\tau} \to (A^\sigma \to A^\tau)$. This function, when applied to an arbitrary element $d \in A^{\sigma\to\tau}$, returns its applicative behaviour.

Comprehension Check 14.1. *Show that* $\overline{\mathrm{App}}^{\sigma\tau}$ *maps each element to its applicative behaviour.*

Definition 14.5 (Full structures)**.** *An applicative structure,* **A***, is full iff for every* σ *and* τ*, every applicative behaviour* $f : A^\sigma \to A^\tau$ *is realized by at least one element in* $A^{\sigma\to\tau}$

- *For any function* $f : A^\sigma \to A^\tau$ *there is at least one element* $d \in A^{\sigma\to\tau}$ *such that* $\mathrm{App}^{\sigma\tau}(d, a) = f(a)$ *for every* $a \in A^\sigma$.

.

Exercise 14.1. *Which of the above structures—Examples 14.1, 14.2, 14.3 and 14.5 are full? In cases where they are not full, exhibit an applicative behaviour* $f : A^t \to A^t$ *that is not realized by any element* $d \in A^{t\to t}$.

In a full applicative structure the applicative behaviour function $\overline{\mathrm{App}}$ surjectively maps elements of $A^{\sigma\to\tau}$ to applicative behaviours in $A^\sigma \to A^\tau$.

Comprehension Check 14.2. *Show that an applicative structure* **A** *is full iff* $\overline{\mathrm{App}}^{\sigma\tau} : A^{\sigma\to\tau} \to A^\sigma \to A^\tau$ *(curried* $\mathrm{App}^{\sigma\tau}$*) is surjective for each* σ *and* τ.

Functions are governed by a principal of functionality: if $f, g : A \to B$, and $f(a) = g(a)$ for every $a \in A$ then $f = g$. This principal also delineates a special class of applicative structures:

Definition 14.6 (Functional structures)**.** *An applicative A is functional iff for every* σ *and* τ*, every applicative behaviour* $f : A^\sigma \to A^\tau$ *is realized by at most one element of* $A^{\sigma\to\tau}$*:*

- *If* $g, h \in A^{\sigma\to\tau}$ *and* $\mathrm{App}^{\sigma\tau}(g, a) = \mathrm{App}^{\sigma\tau}(h, a)$ *for every* $a \in A^\sigma$ *then* $g = h$.

Exercise 14.2. *Which of the above structures are functional. In each case explain your answer. In cases where they are not functional, exhibit two element* $f, g \in A^{t\to t}$ *that are distinct but have the same applicative behaviour.*

As before we may think of $\overline{\mathrm{App}}$ as associating each operation $\tilde{f} \in A^{\sigma\to\tau}$ with an applicative behaviour, a function from A^σ to A^τ. In a functional structure, the association with applicative behaviours is injective, meaning that distinct $\sigma \to \tau$ operations are associated with different applicative behaviours.

Comprehension Check 14.3. *Show that an applicative structure is functional iff* $\overline{\mathrm{App}}^{\sigma\tau} : A^{\sigma\to\tau} \to A^\sigma \to A^\tau$ *is injective for every* σ *and* τ.

Our final definition concerns a special sort of applicative structure that has the ability to interpret the S and K combinators.

Definition 14.7 (Having combinators)**.** *An applicative structure* $\mathbf{A}$ has combinators *iff for each* σ, τ *and* ρ *there is an element* $\tilde{s} \in A^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$ *such that for any* $g \in A^{\sigma\to\tau\to\rho}$, $h \in A^{\sigma\to\tau}$ *and* $a \in A^{\sigma}$*:*

$$\mathrm{App}(\mathrm{App}(\mathrm{App}(\tilde{s}, g), h), a) = \mathrm{App}(\mathrm{App}(g, a), \mathrm{App}(h, a))$$

and for every σ *and* τ *there is an element* $\tilde{k} \in A^{\sigma\to\tau\to\sigma}$ *such that for any* $a \in A^{\sigma}$ *and* $b \in A^{\tau}$*:*

$$\mathrm{App}(\mathrm{App}(\tilde{k}, a), b) = a$$

A more concise definition may be given as follows. Let $k : A^{\sigma} \to A^{\tau} \to A^{\sigma}$ *be the function* $a \mapsto (b \mapsto a)$*. Let* $s : A^{\sigma\to\tau\to\rho} \to A^{\sigma\to\tau} \to A^{\sigma} \to A^{\rho}$ *be the function* $g \mapsto (h \mapsto (a \mapsto g(a)(h(a))$*. Then an applicative structure has combinators iff:*

- *The applicative behaviour* $k : A^{\sigma} \to A^{\tau} \to A^{\sigma}$ *is realized by an element of* $A^{\sigma\to\tau\to\sigma}$ *for every* σ *and* τ*.*
- *The applicative behaviour* $s : A^{\sigma\to\tau\to\rho} \to A^{\sigma\to\tau} \to A^{\sigma} \to A^{\rho}$ *is realized by an element of the appropriate domain for every* σ, τ *and* ρ*.*

Exercise 14.3. *Which of the above structures have combinators.*

Exercise 14.4. *Show that every full structure has combinators.*

Exercise 14.5. *Show that if an applicative structure has combinators then the curried ternary function* $c : A^{\sigma\to\tau\to\rho} \to A^{\tau} \to A^{\sigma} \to A^{\rho}$ *is an applicative behaviour that is realized, where* c *is defined as* $g \mapsto (a \mapsto (b \mapsto g(b)(a)))$*.)*

Similarly show that $i : A^{\sigma} \to A^{\sigma}$, $a \mapsto a$ *is realized.*[i]

Exercise 14.6.

a. *For each of the combinators* C, K, S, I, B' *and* W *write out an equation that an element of an applicative structure would have to satisfy in order to contain an interpretation of that combinator. For instance, for* B *we need an element* b *in* $A^{(\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$ *satisfying the equation*

$$\mathrm{App}(\mathrm{App}(\mathrm{App}(b, f), g), a) = \mathrm{App}(f, \mathrm{App}(g, a))$$

for every $f \in A^{\tau\to\rho}$, $g \in A^{\sigma\to\tau}$ *and* $a \in A^{\sigma}$*.*

b. *Consider the applicative structure in Example 14.5, where* $A^{\sigma} = \mathbb{N}$ *for every sigma, and* $\mathrm{App}^{\sigma\tau}(n, m) = n + m$*. Which of the above combinators can this structure interpret? (That is, which equations are satisfied by elements of this structure?)*

14.2 Functional interpretations

To interpret a typed language, $\mathcal{J}(\Sigma)$ in an applicative structure $\mathbf{A}$ we must define a mapping that takes a term M of type σ to an element, $[\![M]\!] \in A^{\sigma}$, that represents the terms denotation.

If the applicative structure is functional then there is only one degree of freedom in this process: how one interprets the constants in Σ. Once these are fixed, the interpretations of the remaining terms are fixed compositionally—the interpretations of smaller terms (possibly with respect to variables assignments) fix the interpretations of the larger terms. Consequently, a model of a typed language over a signature Σ can be determined by assigning interpretations to the constants.

In this section, we will treat the interpretation of the full λ-language $\mathcal{L}(\Sigma)$ in functional applicative structures. Curiously, when we move to non-functional applicative structures there is another degree of freedom: how we interpret the λ-terms. Consequently, an analogous principle of compositionality doesn't straightforwardly hold for interpretations over non-functional applicative structures. We will hold off treating that case until the next section, where we consider where we formulate the notion of a model in terms of a constraint. This will also prove helpful in generalizing the models to an arbitrary general λ-language $\mathcal{J}(\Sigma)$.

Definition 14.8 (Interpretation)**.** *Let Σ be a signature. An* interpretation $\mathbf{I}$ *for this signature is a pair $(\mathbf{A}, [\![\cdot]\!]_I)$ where*

- $\mathbf{A}$ *is an applicative structure.*
- *For each type σ, $[\![\cdot]\!]$ determines a function $\Sigma^\sigma \to A^\sigma$, mapping $c \in \Sigma^\sigma$ to a denotation $[\![c]\!]$.*

$\mathbf{I}$ *is full, functional or has combinators iff the underlying applicative structure $\mathbf{A}$ is full, functional or has combinators respectively.*

Example 14.6. *If A is the term structure of Example 14.2, for a signature Σ, then it may be turned into an interpretation for the signature Σ by setting $[\![c]\!] = c$ for every $c \in \Sigma$.*

Interpretations can be used to interpret various typed languages. Recall that an applicative typed language (without variables) over a signature Σ consists of just the expressions you can make from the signature Σ by applying terms $M : \sigma \to \tau$ to terms $N : \sigma$ (these languages do not contain λ). For any applicative typed language, one can extend $[\![\cdot]\!]$ so that it takes arbitrary terms of typed language of type σ to denotations in A^σ.

Definition 14.9 (Interpretations of typed applicative languages)**.** *If $\mathcal{A}(\Sigma)$ is a typed applicative language and $M = (A, [\![\cdot]\!])$ is an interpretation, we can extend the interpretation function to arbitrary terms $M : \sigma$ as follows:*

- $[\![(MN)]\!] = \mathrm{App}^{\sigma\tau}([\![M]\!], [\![N]\!])$ *when $M : \sigma \to \tau$, $N : \sigma$.*

This definition suffices because the only way to make new terms from old in an applicative language is by application. If a structure has combinators, and is functional, it can be extended to model the combinatory language $CL(\{S, K\}, \Sigma)$.

Definition 14.10 (Functional combinatory interpretations)**.** *If $CL(\{S, K\}, \Sigma)$ is a combinatory language and $M = (A, [\![\cdot]\!])$ is a functional interpretation of the signature $\Sigma \cup \{S^{\sigma\tau\rho}, K^{\sigma\tau}\}$ with combinators, we call it a* combinatory structure*. We can extend the interpretation function to arbitrary terms $M : \sigma$ as follows:*

- $[\![(MN)]\!] = \mathrm{App}^{\sigma\tau}([\![M]\!], [\![N]\!])$ *when $M : \sigma \to \tau$, $N : \sigma$.*

- $[\![S]\!]$ *uniquely realizes the applicative behaviour* $s : A^{\sigma\to\tau\to\rho} \to A^{\sigma\to\tau} \to A^{\sigma} \to A^{\rho}$ *of Definition 14.7.*
- $[\![K]\!]$ *is the unique element with the applicative behaviour* $k : A^{\sigma} \to A^{\tau} \to A^{\sigma}$ *of Definition 14.7.*

One might have thought that we don't really need to assume that M is functional in order to interpret combinatory languages. Functionality guarantees that is at most one element with applicative behaviours s and k respectively; if we are in a non-functional applicative structure they may be realized by several elements of the structure. We could just let $[\![S]\!]$ be any element that realizes s, and $[\![K]\!]$ any element that realizes k. Unfortunately doing so does not respect the Synonymy Thesis of Chapter 3: synonymous terms could fail to receive identical denotations. We will encounter concrete examples shortly.

The interpretation of λ-terms in an applicative structure is more complex due to the presence of variables. A term M involving free variables will receive a denotation in an applicative structure only relative to a variable assignment, g, telling us how to interpret the variables appearing in M.

Definition 14.11 (Variable assignment)**.** *A variable assignment on an applicative structure* $\mathbf{A}$*, is a type-indexed family of functions* g *where* $g^{\sigma} : Var^{\sigma} \to A^{\sigma}$*. In general, we will omit superscripts from* g*.*

If $x \in Var^{\sigma}$ *and* $a \in A^{\sigma}$ *we define another variable assignment* $g[a/x]$ *as follows:*

$g[a/x](y) = g(y)$ *when* $y \neq x$ *and* $= a$ *when* $y = x$*.*

This operation can be applied repeatedly. When $x_1 \ldots x_n$ *are all distinct variables we will shorten* $g[a_1/x_1]\ldots[a_n/x_n]$ *to* $g[a_1/x_1 \ldots a_n/x_n]$*. When* $\bar{x}$ *and* $\bar{a}$ *stand for the sequences* $x_1 \ldots x_n$ *and* $a_1 \ldots a_n$ *respectively, we can also write* $g[\bar{a}/\bar{x}]$ *for short.*

Definition 14.12 (Functional $\mathcal{L}(\Sigma)$-interpretation)**.** *If* $\mathcal{L}(\Sigma)$ *is the full* λ*-language over signature* Σ *and* $M = (A, [\![\cdot]\!])$ *is a functional interpretation of* Σ *with combinators, we call it a functional* $\mathcal{L}(\Sigma)$*-interpretation. In this case, we can extend the interpretation function to assign values* $[\![M]\!]^g \in A^{\sigma}$ *whenever* $M : \sigma$ *and* g *is a variable assignment:*

- $[\![x]\!]^g = g(x)$.
- $[\![c]\!]^g = [\![c]\!]$ *when* $c \in \Sigma$.
- $[\![(MN)]\!]^g = \mathrm{App}^{\sigma\tau}([\![M]\!]^g, [\![N]\!]^g)$ *when* $M : \sigma \to \tau$ *and* $N : \sigma$.
- $[\![\lambda x.M]\!]^g = d$*, where* d *uniquely realizes the applicative behaviour* $A^{\sigma} \to A^{\tau}$*:*

  $a \mapsto [\![M]\!]^{g[a/x]}$

  when $x : \sigma, M : \tau$*.* $[\![\lambda x.M]\!]^g$ *is undefined if there isn't a unique element that realizes this applicative behaviour.*

The last clause leaves open the possibility that $[\![\cdot]\!]$ is undefined on some arguments, however we will prove shortly that this possibility never arises, and that it is always defined.

The following exercises will help you develop some familiarity with how λ-terms are interpreted in the full Henkin structure of Example 14.1.

Exercise 14.7. *Consider the full Henkin structure where $A^e = \mathbb{N}$, $A^t = \{0,1\}$ and $A^{\sigma\to\tau} = A^\sigma \to A^\tau$. Suppose $0 : e$, $suc : e \to e$, $plus : e \to e \to e$, and $\leq : e \to e \to t$ where $[\![0]\!] = 0$, $[\![suc]\!] = n \mapsto n+1$, $[\![plus]\!] = n \mapsto (m \mapsto n+m)$ and $[\![\leq]\!] = n \mapsto m \mapsto (1$ if $n \leq m$ and 0 otherwise$)$. Using the compositional clauses for the interpretation function, calculate:*

a. $[\![(\leq (suc\, 0))0]\!]$

b. $[\![\lambda x.x]\!]$ *where* $x : e$

c. $[\![\lambda x \lambda y.(\leq y)x]\!]$

d. $[\![(plus\, y)(suc(suc\, x))]\!]^g$ *for an arbitrary assignment* g.

e. $[\![\lambda x \lambda y.(plus\, y)(suc(suc\, x))]\!]$

Note that our definition of $[\![M]\!]^g$ in a functional interpretation without combinators still makes sense, it's just that $[\![M]\!]^g$ will sometimes be undefined. This is explored in the following exercise.

Exercise 14.8. *Consider the interpretation of $\wedge : t \to t \to t$, and $F, G : e \to t$ in the $\mathcal{L}(\Sigma)$-interpretation of Example 14.5 where $[\![F]\!] = 3$, $[\![G]\!] = 7$, $[\![\wedge]\!] = 10$. Show $[\![M]\!]^g$ is defined (for arbitrary g) for the following terms M, and say what they are if they are defined.*

a. $[\![Fx]\!]^g$ *for arbitrary* g.

b. $[\![\lambda x.Fx]\!]$

c. $[\![\lambda x.\lambda y.(Fy \wedge Gx)]\!]$

d. $[\![\lambda x.(Fx \wedge Gx)]\!]$

e. $[\![\lambda X.\lambda y.\lambda z.Xzy]\!]$

It turns out that, in a functional $\mathcal{L}(\Sigma)$-interpretation with combinators, $[\![M]\!]^g$ is always defined. That there is a unique element that realizes the applicative behaviour $a \mapsto [\![M]\!]^{g[a/x]}$, if there is one at all, follows straightforwardly from functionality. The fact that the $\mathcal{L}(\Sigma)$-interpretation has combinators furthermore ensures there always is an element that realizes $a \mapsto [\![M]\!]^{g[a/x]}$. To establish this we need a lemma:

Lemma 14.1 (Substitution lemma)**.** *Suppose that $[\![M]\!]^g$ and $[\![N]\!]^g$ are defined for every assignment g. Then:*

$$[\![M]\!]^{g[[\![N]\!]^g/y]} = [\![M[N/y]]\!]^g$$

provided that N is free for y in M.

The proof of the substitution lemma is a good way to familiarize yourself with Definition 14.12.

Exercise 14.9. *In this exercise, you will complete all of the steps for an induction establishing the substitution lemma: $[\![M]\!]^{g[[\![N]\!]^g/y]} = [\![M[N/y]]\!]^g$. The induction is on the complexity of the term M.*

a. Show that the substitution lemma holds vacuously when M is a constant or variable distinct from y.

b. Show that $[\![y]\!]^{g[[\![N]\!]^g/y]} = [\![N]\!]^g = [\![y[N/y]]\!]^g$

c. Suppose that $[\![P]\!]^{g[[\![N]\!]^g/y]} = [\![P[N/y]]\!]^g$ and $[\![Q]\!]^{g[[\![N]\!]^g/y]} = [\![Q[N/y]]\!]^g$, and show that $[\![(PQ)]\!]^{g[[\![N]\!]^g/y]} = [\![(PQ)[N/y]]\!]^g$.

d. Suppose that $[\![M]\!]^{g[[\![N]\!]^g/y]} = [\![M[N/y]]\!]^g$ and show that $[\![\lambda x.M]\!]^{g[[\![N]\!]^g/y]} = [\![(\lambda x.M)[N/y]]\!]^g$

It's easy to see, using the substitution lemma, that if x is not free in M then $[\![M]\!]^g = [\![M]\!]^{g[a/x]}$ for arbitrary x. More generally, we can prove that $[\![M]\!]^g = [\![M]\!]^h$ for any g and h agreeing on the free variables in M:

Proposition 14.1. *Suppose that $FV(M) \subseteq \{x_1, \ldots, x_n\}$. For all assignments g and h, if $g(x_i) = h(x_i)$ for all x_i, then $[\![M]\!]^g = [\![M]\!]^h$.*

Proof. We firstly prove that when $FV(M) \subseteq \{x_1, \ldots, x_n\}$, $[\![M]\!]^h = [\![M]\!]^{g[\overline{h(x)}/\overline{x}]}$ for all g and h. And this can be easily done by induction on the structure of M.[4] Then note that given the assumption about g and h, $[\![M]\!]^{g[\overline{h(x)}/\overline{x}]} = [\![M]\!]^{g[\overline{g(x)}/\overline{x}]} = [\![M]\!]^g$. □

Convention 14.3. *When M is closed we write $[\![M]\!]$ to mean $[\![M]\!]^g$ for any g.*

We shall show something seemingly stronger: that the function $a \mapsto [\![M]\!]^{g[a/x]} : A^\sigma \to A^\tau$ is well-defined for any variable $x : \sigma$ and has a unique element realizing it in $A^{\sigma\to\tau}$ for any term $M : \tau$. The well-definedness of this function just means that it has a value for each argument. Thus, in particular, it means that $[\![M]\!]^{g[a/x]} \in A^\tau$ is well-defined for every $a \in A^\sigma$, establishing that $[\![M]\!]^g$ is well-defined as required.

Theorem 14.1. *If $\mathbf{A} = (A, [\![\cdot]\!])$ is a functional $\mathcal{L}(\Sigma)$-interpretation (so that $\mathbf{A}$ is a functional applicative structure with combinators), then for every term M of $\mathcal{L}^\tau(\Sigma)$ and any variables $x_1, \ldots, x_n$ of types $\sigma_1, \ldots, \sigma_n$, the function $a_1 \mapsto \cdots \mapsto a_n \mapsto [\![M]\!]^{g[a_1/x_1,\ldots,a_n/x_n]} : A^{\sigma_1} \to \cdots \to A^{\sigma_n} \to A^\tau$ is well-defined and there is a unique element of $A^{\bar\sigma\to\tau}$ that realizes it.*

Proof. In what follows we illustrate each of the steps with the $n = 1$ case first and then show how to generalize the argument for $n > 1$.

For each constant $c \in \Sigma^\tau$ and variable $y : \tau$ distinct from x, $a \mapsto [\![c]\!]^{g[a/x]}$ and $a \mapsto [\![y]\!]^{g[a/x]}$ are constant functions and are realized by $\mathrm{App}(\tilde{k}^{\tau\sigma}, [\![c]\!])$ and $\mathrm{App}(\tilde{k}^{\tau\sigma}, g(y))$ in the structure respectively. In the case $y = x$, $a \mapsto [\![x]\!]^{g[a/x]}$ is just $a \mapsto a$ and is uniquely realized by $\tilde{i}^\sigma$ (defined from $\tilde{s}$ and $\tilde{k}$ as in Exercise 14.5). To show the more general case, that $a_1 \mapsto \cdots \mapsto a_n \mapsto [\![c]\!]^{g[\bar{a}/\bar{x}]}$ is realized, we simply apply $\tilde{k}$ repeatedly to the denotation of c to obtain the desired element. Elements realizing the projection functions $a_1 \mapsto \cdots \mapsto a_n \mapsto a_j$ where $1 \leq j \leq n$ can similarly be made using combinations of $\tilde{s}$ and $\tilde{k}$. Projection of the final argument, which realizes $a_1 \mapsto \cdots \mapsto a_n \mapsto a_n$, is made by applying $n - 1$ instances of a k operation (of the appropriate type) to something that realizes the identity function i^{σ_n}. Every other projection function can be obtained from this as a generalized converse, transposing the jth argument with the final argument. Generalized converses can be made from s and k as we know from Section 3.5.)

Let $M : \tau \to \rho$ and $N : \tau$. Suppose for induction that $f (= a \mapsto [\![M]\!]^{g[a/x]}) : A^\sigma \to A^{\tau\to\rho}$ and $h (= a \mapsto [\![N]\!]^{g[a/x]}) : A^\sigma \to A^\tau$ are well-defined and are realized by $\tilde{f} \in A^{\sigma\to\tau\to\rho}$ and $\tilde{h} \in A^{\sigma\to\tau}$ respectively. For any $a \in A^\sigma$,

$$\mathrm{App}(\mathrm{App}(\mathrm{App}(\tilde{s}^{\sigma\tau\rho}, \tilde{f}), \tilde{h}), a) = \mathrm{App}(\mathrm{App}(\tilde{f}, a), \mathrm{App}(\tilde{h}, a))$$

by the characteristic equation for $\tilde{s}$. Since $\tilde{f}$ realizes $a \mapsto [\![M]\!]^{g[a/x]} : A^\sigma \to A^{\tau\to\rho}$ and $\tilde{h}$ realizes $a \mapsto [\![N]\!]^{g[a/x]} : A^\sigma \to A^\tau$, $\mathrm{App}(\tilde{f}, a) = [\![M]\!]^{g[a/x]}$ and $\mathrm{App}(\tilde{h}, a) = [\![N]\!]^{g[a/x]}$. Thus:

$$\mathrm{App}(\mathrm{App}(\tilde{f}, a), \mathrm{App}(\tilde{h}, a)) = \mathrm{App}([\![M]\!]^{g[a/x]}, [\![N]\!]^{g[a/x]}) = [\![MN]\!]^{g[a/x]}$$

that is, $\mathrm{App}(\mathrm{App}(\tilde{s}^{\sigma\tau\rho}, \tilde{f}), \tilde{h})$ applied to a yields $[\![MN]\!]^{g[a/x]}$ so that it realizes the function $a \mapsto [\![MN]\!]^{g[a/x]}$. It is unique because the structure is functional.

For the general case where $n > 1$ we must use a generalization of the $s^{\sigma\tau\rho}$ function, $s^{\bar{\sigma}\tau\rho}$ defined as $f \mapsto h \mapsto \bar{a} \mapsto f(\bar{a})h(\bar{a})$. Here we are extending our the bar notation from λ to $\mapsto$ in the obvious way, so that, e.g. $\bar{a} \mapsto$ is short for $a_1 \mapsto \cdots \mapsto a_n \mapsto$. $s^{\bar{\sigma}\tau\rho}$ can be defined from s and k using the techniques introduced in Section 3.5.

In the final case, we treat a general n. We want to show that $a_1 \mapsto \cdots \mapsto a_n \mapsto [\![\lambda x.M]\!]^{g[\bar{a}/\bar{y}]}$ is uniquely realized where $M : \tau$. By the inductive hypothesis, the $n+1$-ary function $f = a_1 \mapsto \cdots \mapsto a_n \mapsto b \mapsto [\![M]\!]^{g[\bar{a}/\bar{y}][b/x]}$ is well-defined[5] and is uniquely realized by $\tilde{f} \in A^{\bar{\sigma} \to \tau}$. $\tilde{f}$ must also realize the function $a_1 \mapsto \cdots \mapsto a_n \mapsto [\![\lambda x.M]\!]^{g[\bar{a}/\bar{y}]}$ since they have the same applicative behaviour, and it is unique by functionality. □

Theorem 14.2. *If M and N are $\beta\eta$-equivalent terms and $(\mathbf{A}, [\![\cdot]\!])$ a functional $\mathcal{L}(\Sigma)$-interpretation, then for every variable assignment g, $[\![M]\!]^g = [\![N]\!]^g$.*

Proof. Recall that M is said to be immediately β-reducible to N if we can get N by replacing one term of the form $(\lambda x.P)Q$ in M with $P[Q/x]$ provided Q is free for x in P; and M is said to be immediately η-reducible to N if we can get N by replacing one term of the form $\lambda x.Px$ in M with P provided x is not free in P. Now, it suffices to show that if M is N or is immediately β/η-reducible to N, then $[\![M]\!]^g = [\![N]\!]^g$ for all g.

By induction on the structure of M. The base cases are trivial. When $M = PQ$, we have two cases:

(i) $N = P'Q'$ where P is identical or immediately β/η-reducible to P' and Q is identical or immediately β/η-reducible to Q'. Then we have:

$$\begin{aligned}[\![PQ]\!]^g &= \mathrm{App}([\![P]\!]^g, [\![Q]\!]^g) \\ &= \mathrm{App}([\![P']\!]^g, [\![Q']\!]^g) \text{ (by inductive hypothesis)} \\ &= [\![P'Q']\!]^g\end{aligned}$$

(ii) $M = (\lambda x.P)Q$ and $N = P[Q/x]$ where Q is free for x in P. Then we have:

$$\begin{aligned}[\![(\lambda x.P)Q]\!]^g &= \mathrm{App}([\![\lambda x.P]\!]^g, [\![Q]\!]^g) \\ &= (a \mapsto [\![P]\!]^{g[a/x]})([\![Q]\!]^g) \\ &= [\![P]\!]^{g[[\![Q]\!]^g/x]} \\ &= [\![P[Q/x]]\!]^g \text{ (by Lemma 14.1)}\end{aligned}$$

Suppose $M = \lambda x.P$. Then $[\![\lambda x.P]\!]^g = \tilde{f}$, which should realize the applicative behaviour $a \mapsto [\![P]\!]^{g[a/x]}$. Let's discuss two cases in turn:

(i) $N = \lambda x.P'$ where P is identical or immediately $\beta\eta$-reducible to P'. Then $a \mapsto [\![P]\!]^{g[a/x]}$ and $a \mapsto [\![P']\!]^{g[a/x]}$ are the same function according to our inductive hypothesis. Since the underlying structure is functional, $[\![\lambda x.P]\!]^g = [\![\lambda x.P']\!]^g$.

(ii) $P = Nx$ for some x not free in N. Then, for all a of the correct type, we have:

$$\begin{aligned}\mathrm{App}(\tilde{f}, a) &= (a \mapsto [\![Nx]\!]^{g[a/x]})(a) = [\![Nx]\!]^{g[a/x]} \\ &= \mathrm{App}([\![N]\!]^{g[a/x]}, [\![x]\!]^{g[a/x]}) \\ &= \mathrm{App}([\![N]\!]^{g}, a) \text{ (by Lemma 14.1)}\end{aligned}$$

Because the underlying structure is functional, $\tilde{f} = [\![N]\!]^{g}$. □

We end with an application of this result. Recall that the Church numerals for type t are combinators of type $(t \to t) \to t \to t$ of the form $\lambda X.\lambda p.X^n p$ where $X : t \to t$, $p : t$, $X^0 p = p$ and $X^{n+1} p = X(X^n p)$.

Exercise 14.10. *By Theorem 14.2 we know that when M and N are $\beta\eta$-equivalent and closed, $[\![M]\!] = [\![N]\!]$ for any $\mathcal{L}(\Sigma)$-interpretation. Use this fact to show that no two Church numerals are $\beta\eta$-equivalent.*[ii]

14.3 The environment model condition

In a functional applicative structure with combinators, an interpretation of a typed language, $\mathcal{L}(\Sigma)$ can be determined by specifying the interpretations of the constants in Σ. Once the interpretations of the constants are fixed, the interpretation of an arbitrary term (relative to a variable assignment) is uniquely determined and can be computed compositionally. Specifically, the interpretation of a predication like (MN) is determined by the structures application operation App, and the interpretation of a λ-term, $(\lambda x.M)$ is the unique element of the structure with a certain applicative behaviour determined by the term M.

If the applicative structure has combinators but isn't functional, we potentially have many choices concerning which element of the structure to interpret $(\lambda x.M)$ with: there might be several elements realizing the applicative behaviour $a \mapsto [\![M]\!]^{g[a/x]}$. As a result, a straightforwardly compositional interpretation of the λ-terms is not in general possible, and the interpretations of arbitrary terms may not supervene on the interpretations of the constants in the signature. Interpretations in non-functional structures are therefore often presented as a constraint on a function from terms of a language into an applicative structure, rather than a definition extending the basic terms. As such, an interpretation in a non-functional applicative structure no longer simply consists of an assignment of a meaning to each *constant* of type σ to an element of A^σ, but a global assignment to arbitrary terms of type σ to elements of A^σ, subject to further constraints.

A quick glance at Theorem 14.1 reveals that we could extend the same process of assigning terms meanings up to the clause for λ-terms, where we could pick instead arbitrarily pick one element realizing the relevant applicative behaviour. That is, letting $M : \tau$ and $x : \sigma$:

$[\![\lambda x.M]\!]^{g}$ = $\tilde{f}$ where $\tilde{f}$ is an arbitrarily chosen element that realizes the applicative behaviour:

$$a \mapsto [\![M]\!]^{g[a/x]} : A^\sigma \to A^\tau$$

Unfortunately this naïve way of picking the meanings of terms will fail to secure the analogue of Theorem 14.2 stating that β-equivalent or η-equivalent terms have the same denotations in a structure. Let's begin with β. Two terms are immediately β-equivalent if one

can be gotten from the other by substituting a term of the form $(\lambda x.M)N$ for $M[N/x]$ in the other. Curiously, provided this substitution doesn't happen underneath the scope of a λ, the above method for assigning meanings to λ-terms counts them as equivalent. For instance:

$$[\![(\lambda x.M)N]\!]^g = \mathrm{App}([\![\lambda x.M]\!]^g, [\![N]\!]^g)$$

but because $[\![\lambda x.M]\!]^g$ realizes the function $a \mapsto [\![M]\!]g[a/x]$,

$$\mathrm{App}([\![\lambda x.M]\!]^g, [\![N]\!]^g) = [\![M]\!]^{g[[\![N]\!]^g/x]}$$

and finally, by Lemma 14.1 (which continues to hold in the present context)

$$[\![M]\!]^{g[[\![N]\!]^g/x]} = [\![M[N/x]]\!]^g$$

However, the above process fails to identify these terms when they appear under λs. Consider the terms $\lambda Z.Z((\lambda x.M)N)$ and $\lambda Z.Z(M[N/x])$. To interpret the former, using our proposed method of choosing arbitrarily, we would pick something that realizes the function:

$$d \mapsto [\![Z((\lambda x.M)N)]\!]^{g[d/x]}$$

Similarly, to interpret the form we pick an arbitrary element realizing the function

$$d \mapsto [\![Z(M[N/x])]\!]^{g[d/x]}$$

Of course, since the immediate β-equivalents defining the above two functions no longer appear under λs, these two functions are in fact identical. But because we are making arbitrary choices independently for each term, there's no guarantee that we pick the same realizing element in either case.

Exercise 14.11. *Explain how the above procedure for picking denotations could fail to secure η by considering what the method might deliver as the interpretations of F and $\lambda x.Fx$, where F is a constant.*

One might try to fix the issue preventing us from securing β by making sure that such choices are coordinated between terms. For instance, one could pick a partial function $[\![\lambda]\!] : (A^\sigma \to A^\tau) \to A^{\sigma\to\tau}$ which given a function $f : A^\sigma \to A^\tau$ picks out a particular element realizing f in $A^{\sigma\to\tau}$ if one exists. But the problem is deeper than that. Recall that function composition is associative, which means if you have function $h : A \to B, g : B \to C$ and $f : C \to D$ then:

$$f \circ (g \circ h) = (f \circ g) \circ h$$

It's natural to think that functional expressions in typed languages are subject to similar laws. The complex adverb *not definitely necessarily* can be equally well analysed in terms of composing *not definitely* with *necessarily* or *not* with *definitely necessarily*.

While this identity can be justified by the principal of functionality, it is strictly weaker than it, and is guaranteed by the principle of β-equivalence, which can be used to show that $BX(BYZ)$ and $B(BXY)Z$ are synonymous (provided B is given suitable type superscripts). We have encountered many similar synonymy claims, such as $SKK = I$ (again, with suitable

type superscripts). But it's easy to see how one could go wrong, with the latter identity, if the identity applicative behaviour $i : A^\sigma \to A^\sigma$ is realized by several elements. The combinator expressions S and the versions of K will end up expressing whatever elements are chosen to realize the s and k functions. Thus SKK will end up picking out a particular element to realize identity function $A^\sigma \to A^\sigma, \tilde{j}$. Thus it's clear that if we picked a different element, $\tilde{i}$, i.e. something distinct from $\tilde{j}$, to be the designated element realizing the identity function we would not secure the synonymy of SKK and I. The latter is a λ-term, $\lambda x.x$, and its interpretation is calculated directly by using the designated element realizing the identity function, whereas $(SK)K$ is computed by applying two already determined meanings to each other. Similar points apply to the associativity of composition, and other identities secured by β.

The solution here is to replace the explicit recursive definition of $[\![\cdot]\!]$ from an interpretation of a signature with a constraint on a function defined on *all* terms of the language, which includes, among other things, assigning the same value to $\beta\eta$-equivalent terms. The more general notion of an interpretation that results solves another problem we have not yet addressed: to specify the relevant notion of interpretation for an arbitrary λ-language $\mathcal{J}(\Sigma)$. In the case of the full λ-language in functional applicative structures, we were able to provide a recursive definition of $[\![\cdot]\!]$ given the assumption that the structure *had combinators*. This condition essentially ensured that there were enough entities to interpret all of the λ-terms. Since general λ-languages are sublanguages of the full λ-language they can always be interpreted in functional structures with combinators. However, intuitively they should have a wider class of interpretations: λ-languages that do not contain any combinator expressions, for instance, should also have models that don't have combinators.

All of this brings us to our definition of an interpretation of a general λ-language in a non-functional applicative structure.[6]

Definition 14.13 (The environment model condition)**.** *A general $\mathcal{J}(\Sigma)$-interpretation of a general λ-language $\mathcal{J}(\Sigma)$ is a pair $(\mathbf{A}, [\![\cdot]\!])$ where $\mathbf{A} = (A^{\cdot}, \mathrm{App}^{\cdot})$ is an applicative structure and $[\![\cdot]\!]$ determines a function taking an* arbitrary *term M in $\mathcal{J}^\sigma(\Sigma)$ and an assignment g, to an element $[\![M]\!]^g \in A^\sigma$, subject to the constraints:*

- $[\![x]\!]^g = g(x)$.
- $[\![(MN)]\!]^g = \mathrm{App}^{\sigma\tau}([\![M]\!]^g, [\![N]\!]^g)$ *where* $M : \sigma \to \tau, N : \sigma$.
- $[\![M]\!]^g = [\![N]\!]^h$ *when* $M \sim_{\beta\eta} N$ *and h and g agree on* $FV(M) \cap FV(N)$.

When $(A, [\![\cdot]\!])$ satisfies these constraints we say it satisfies the environment model condition *with respect to the language $\mathcal{J}(\Sigma)$.*

Observe that a general $\mathcal{J}(\Sigma)$-interpretation for $\mathcal{J}(\Sigma)$ doesn't merely tell us the interpretation of the constants: $[\![\cdot]\!]$ is defined on all term-assignment pairs at once, and is subject to various constraints. For a given assignment of interpretations to the primitive constants in Σ, there may be several complete interpretation functions extending it that satisfy the constraints in our definition. This corroborates our earlier assertion that the interpretations are no longer compositional, in the sense that the interpretation of the complex terms relative to a variable assignment supervenes on the interpretations of the constants and assignments of values to variables.

By the same token, there mightn't be any interpretation function satisfying the environment model condition that extends a particular assignment of denotations to the constants. Clearly an applicative structure without combinators cannot satisfy the environment

model condition with respect to a λ-language that does contain combinators (there might be nothing in the structure to interpret the S and K combinators). But even in an applicative structure with combinators, it is not obvious that an interpretation extending any given assignment of denotations to the constants can be given.

Secondly, notice that for functional interpretations we were able to prove Theorem 14.2, telling us that $\beta\eta$-equivalent terms had the same denotations. This is no longer a derived theorem in the context of non-functional interpretations, but something that has been built in by hand. It means that checking a given interpretation even *is* a general $\mathcal{J}(\Sigma)$-interpretation is a non-trivial task. As a result, it makes these models rather ineffectual, at least with respect to the use that models are often put to: namely to establish the underivability of something. For instance, if I wanted to show that a pair of terms M and N aren't $\beta\eta$-equivalent, a natural strategy is to find a $\mathcal{J}(\Sigma)$-interpretation such that $[\![M]\!] \neq [\![N]\!]$. By Theorem 14.2, it follows that M and N cannot be $\beta\eta$-equivalent. This strategy is fine if we are using the functional $\mathcal{L}(\Sigma)$-interpretations of Section 14.2 because all you need to do is specify the interpretations of the constants in a functional applicative structure with combinators. We don't need to check further whether the result is a functional $\mathcal{L}(\Sigma)$-interpretation or not, that has been taken care of already by Theorem 14.1. On the other hand, to build a general $\mathcal{L}(\Sigma)$-interpretation, one of the things we need to check is that $[\![P]\!] = [\![Q]\!]$ for every pair of closed $\beta\eta$-equivalent terms P and Q, and also check that $[\![M]\!] \neq [\![N]\!]$. But this is effectively the same as showing that the identity of M and N cannot be derived from the laws of $\beta\eta$-equivalence: there is not enough distance between the model-theoretic fact and the proof-theoretic fact that the model theory is being employed to settle.

Remark 14.1. It is worth noting that mathematicians and computer scientists sometimes choose to formulate the notion of an interpretation for a λ-language in terms of the environment model condition, even in the functional case.[7] For functional interpretations, however, the constraint is satisfied uniquely exactly if the underlying applicative structure has combinators, and is equivalent to the compositional definition of denotation given in Definition 14.12.

Remark 14.2. While the environment model condition gives us a suitably general notion of model that is applicable to an arbitrary general λ-language, the problem of defining an appropriate notion of model for particular general λ-language is often much simpler. For instance, we saw that the relevant, linear and affine languages have combinatory bases $\{S, B, C, I\}$, $\{B, C, I\}$ and $\{B, C, I, K\}$, respectively. Thus one can modify our condition of a model *having combinators* in the obvious way to obtain the notion of a functional model for these languages in a straightforward way.

Although we have emphasized the shortcomings of non-functional $\mathcal{J}(\Sigma)$-interpretations here, it should be noted that, in practice, many non-functional applicative structures carry extra structure beyond the structures notion of application, App, and in these cases the extra structure singles out a preferred interpretation of λ-terms, and more standard compositional definitions are possible. This basic shortcoming is one we will attempt to overcome with the applicative structures introduced in Chapters 17 and 18: they tend to have more structure which allows one to provide a compositional semantics for λ-languages. Finally, category theory provides a completely general setting in which to reason about non-functional models of the full λ-language, and we will touch on this briefly in Chapter 17.

We end by noting that when A is functional and has combinators the two notions of $\mathcal{L}(\Sigma)$-interpretation coincide.

Proposition 14.2. *Suppose* $\mathbf{I} = (\mathbf{A}, [\![\cdot]\!])$ *is a functional* $\mathcal{L}(\Sigma)$*-interpretation with combinators, with* $[\![\cdot]\!]^g$ *extended to arbitrary terms compositionally as in Theorem 14.1. Then* $[\![\cdot]\!]$ *satisfies:*

1. $[\![x]\!]^g = g(x)$.
2. $[\![(MN)]\!]^g = \mathrm{App}^{\sigma\tau}([\![M]\!]^g, [\![N]\!]^g)$ *where* $M : \sigma \to \tau, N : \sigma$.
3. $[\![M]\!]^g = [\![N]\!]^h$ *when* $M =_{\beta\eta} N$ *and* h *and* g *agree on* $FV(M) \cap FV(N)$.

Conversely, suppose $[\![\cdot]\!]'$ *is a typed collection of functions from* $\mathcal{L}^\sigma(\Sigma)$ *to* A^σ *that satisfies the analogue of the above three conditions and is such that* $[\![c]\!]' = [\![c]\!]$ *for every constant in* Σ. *Then* $[\![\cdot]\!]' = [\![\cdot]\!]$.

Proof. The first two conditions are easily established, so we show the last. Since M and N are $\beta\eta$-equivalent, it can be shown that $FV(N^*) \subseteq FV(M) \cap FV(N)$ where N^* is the $\beta\eta$-normal form of both M and N. Suppose without loss of generality that g and h assgin different values to $x_1, \ldots, x_n \in FV(N)$. Then by Proposition 14.1, $[\![N]\!]^h = [\![N]\!]^{g[\overline{h(x)}/\overline{x}]}$. So by Theorem 14.2, $[\![N]\!]^h = [\![N^*]\!]^{g[\overline{h(x)}/\overline{x}]}$. But observe that $[\![N^*]\!]^{g[\overline{h(x)}/\overline{x}]} = [\![N^*]\!]^g$ according to Proposition 14.1. Thus, by Theorem 14.2 again we finally get $[\![N]\!]^h = [\![N]\!]^g = [\![M]\!]^g$.

Now suppose that $[\![\cdot]\!]'$ is as above. We prove by induction on the structure of M that $[\![M]\!]^g = [\![M]\!]'^g$. The first condition ensures that it holds for variables, and by our assumption, it holds for constants as well. The second condition ensures it holds for terms of the form MN, so it remains to treat terms of the form $\lambda x.M$. Suppose for the inductive hypothesis that $[\![M]\!]^g = [\![M]\!]'^g$ for every assignment g. We use the functionality of the structure to show the interpretations of the λ-terms coincide: $\mathrm{App}([\![\lambda x.M]\!]^g, a) = [\![(\lambda x.M)x]\!]^{g[a/x]}$ (by the second condition) $= [\![M]\!]^{g[a/x]}$ (by the third condition, since $(\lambda x.M)x =_{\beta\eta} M$) $= [\![M]\!]'^{g[a/x]}$ (by the inductive hypothesis) $= [\![(\lambda x.M)x]\!]'^{g[a/x]}$ (by the third condition) $= \mathrm{App}([\![(\lambda x.M)]\!]'^g, a)$ (by the second condition). □

14.4 Congruences and quotients

In this section, we will consider several operations that allow us to construct new applicative structures from old ones.

Definition 14.14 (Congruence)**.** *A congruence* $\sim$ *on an applicative structure* $\mathbf{A}$ *is a type-indexed collection of equivalence relations,* $\sim^\sigma$ *subject the constraint that:*

- *Whenever* $f, g \in A^{\sigma\to\tau}$ *and* $a, b \in A^\sigma$*, if* $f \sim^{\sigma\to\tau} g$ *and* $a \sim^\sigma b$ *then* $\mathrm{App}(f, a) \sim^\tau \mathrm{App}(g, b)$.

A partial congruence is a typed indexed collection of partial equivalence relations subject to the same constraint.

In other words, applying equivalent operations to equivalent arguments yields equivalent results.

Congruences are familiar notions from other areas of algebra, where they are generally equivalence relations that are preserved by the basic operations of the algebra. For applicative structures, however, there are many occasions where we have reason to use a generalization of the notion of an equivalence relation which behaves like an equivalence relation only on a subset of a given domain. More precisely,

Definition 14.15 (Partial equivalence relation)**.** *A partial equivalence relation* $\sim$ *on a set* X *is a transitive and symmetric (but not necessarily reflexive) relation on* A.

Because it is not reflexive, not every element $a \in X$ is related to itself: some elements of X are left out. However, if a is related to anything, it is related to itself.

Exercise 14.12. *Suppose $\sim$ is a partial equivalence on X. Show that if $a \sim b$ for any $b \in X$ then $a \sim a$. Show that every partial equivalence relation on X is the same as an equivalence relation on a subset of X.*[iii]

Given an equivalence or partial equivalence relation, $\sim$, on a set X and an element $a \in X$, write $[a]_\sim$ for the set $\{b \mid a \sim b\}$. (If $\sim$ is a partial equivalence relation $[a]_\sim$ could be empty, and in particular not even contain a!)

Given a congruence $\sim$ on an applicative structure $\mathbf{A}$ we can form another applicative structure whose elements are equivalence classes of elements of the original applicative structure.

Definition 14.16 (Quotient)**.** *If $\mathbf{A} = (A^{\cdot}, \mathrm{App}^{\cdot})$ is an applicative structure and $\sim$ a congruence (or partial congruence) on $\mathbf{A}$, then $\mathbf{A}/\sim = ((A/\sim)^{\cdot}, \mathrm{App}^{\cdot}_{\sim})$ is the applicative structure:*

- $(A/\sim)^\sigma = \{[a]_{\sim^\sigma} \mid a \in A^\sigma, a \sim^\sigma a\}$
- $\mathrm{App}^{\sigma\tau}_{\sim}([f]_{\sim^{\sigma\to\tau}}, [a]_{\sim^\sigma}) = [\mathrm{App}^{\sigma\tau}(f,a)]_{\sim^\tau}$

In general, the type superscripts on $\sim$ can be inferred from the entities flanking it, and will be omitted. Similarly, we will often write $[a]$ instead of $[a]_{\sim^\sigma}$ when the congruence is clear from context.

Our definition of App is potentially sensitive to the arbitrary representative, f and a, we chose for the equivalence classes $[f]$ and $[a]$. What if $[f'] = [f]$ and $[a'] = [a]$ but $[\mathrm{App}(f',a')] \neq [\mathrm{App}(f,a)]$? Then our definition would be inconsistent because it would deliver both $[\mathrm{App}(f,a)]$ and $[\mathrm{App}(f',a')]$ as the result of a single application. The fact that $\sim$ is a congruence prevents this from happening.

Exercise 14.13. *Show that if $[f] = [f']$ and $[a] = [a']$ then $[\mathrm{App}(f,a)] = [\mathrm{App}(f',a')]$.*

A quotient of an applicative structure by a congruence is effectively the result of coarsening the structure by identifying elements that are distinct in the larger structure. The quotient of an applicative structure by a partial congruence coarsens the structure, but also trims it down: some elements can be thrown away, namely those not in the domain of $\sim$ (i.e. when $a \not\sim a$).

Recall that in Exercise 14.3 you showed that: the term structure $\mathcal{L}(\Sigma)$ did not have combinators. It did not contain any element realizing k such that $\mathrm{App}(\mathrm{App}(\tilde{k}, M), N) = M$ for arbitrary M and N of the appropriate types, since $\tilde{k}$ would be a term, P, such that $(PM)N = M$, and this is not true for any term. But notice the K combinator (of the appropriate type) gets close. Although it does not have the property that $(KM)N = M$ for every M and N, it does have the property that $(KM)N \sim_{\beta\eta} M$ for every M and N. Indeed, we have the following

Proposition 14.3. *The relation $\sim^\sigma_{\beta\eta}$, the relation of $\beta\eta$-equivalence between terms of type σ, is a congruence on the term applicative structure $\mathcal{L}(\Sigma)$.*

Exercise 14.14. *Prove this proposition.*

We may then take the quotient of the term structure by $\beta\eta$-equivalence, $\mathcal{L}(\Sigma)/\sim_{\beta\eta}$.

Exercise 14.15. *In Exercise 14.3 we saw that $\mathcal{L}(\Sigma)$, considered as an applicative structure, does not have combinators. Prove that the applicative structure $\mathcal{L}(\Sigma)/\sim_{\beta\eta}$ has combinators.*

Exercise 14.16. *Show that the quotient of the open term structure $\mathcal{L}(\Sigma)/\sim_{\beta\eta}$ is functional. Is the closed-term structure functional for every signature Σ?*

Finally note that we can define a $\mathcal{L}(\Sigma)$-interpretation from the quotient term structure by interpreting $[\![c]\!] = [c]$. We call this the open term structure for the language $\mathcal{L}(\Sigma)$.

Definition 14.17 (Open term structure)**.** *The open term structure for the language $\mathcal{L}(\Sigma)$ is the $\mathcal{L}(\Sigma)$-interpretation defined by setting*

- $A := \mathcal{L}(\Sigma)/\sim_{\beta\eta}$
- $[\![c]\!] := [c]_{\sim_{\beta\eta}}$

14.5 Homomorphisms

Definition 14.18 (Homomorphism)**.** *If $\mathbf{A} = (A, App_A)$ and $\mathbf{B} = (B, App_B)$ are two applicative structures, a homomorphism $h : \mathbf{A} \to \mathbf{B}$ is a type-indexed collection of functions h^σ such that:*

- $h^\sigma : A^\sigma \to B^\sigma$
- $h^\tau(App_A^{\sigma\tau}(f,a)) = App_B^{\sigma\tau}(h^{\sigma\to\tau}(f), h^\sigma(a))$ *whenever $f \in A^{\sigma\to\tau}$ and $a \in A^\sigma$*

We will in general omit superscripts from h, as they are determined uniquely by the supplied arguments.

Suppose $\mathbf{I} = (\mathbf{A}, [\![\cdot]\!]_I)$ and $\mathbf{J} = (\mathbf{B}, [\![\cdot]\!]_J)$ are interpretations for the same signature Σ. A homomorphism of interpretations $h : \mathbf{I} \to \mathbf{J}$ is a homomorphism of the applicative structures $\mathbf{A}$ and $\mathbf{B}$ such that:

- $h([\![c]\!]_I) = [\![c]\!]_J$ *for every $c \in \Sigma$*

If $\mathbf{I}$ and $\mathbf{J}$ are $\mathcal{J}(\Sigma)$-interpretations or combinatory structures (functional or non-functional), a combinatory *homomorphism is a homomorphism such that:*

- $h([\![K^{\sigma\tau}]\!]_I) = [\![K^{\sigma\tau}]\!]_J$
- $h([\![S^{\sigma\tau\rho}]\!]_I) = [\![S^{\sigma\tau\rho}]\!]_J$

for every type σ, τ and ρ. A λ-homomorphism is a homomorphism such that

- $h([\![M]\!]_I^g) = [\![M]\!]_J^{h\circ g}$

for every term M of $\mathcal{L}(\emptyset)$. We call it $\mathcal{J}$ homomorphism iff the preservation condition holds for every term of $\mathcal{J}(\emptyset)$.

Convention 14.4. *If $\mathbf{A}$ and $\mathbf{B}$ are applicative structures, $\mathcal{J}(\Sigma)$-interpretations, etc. and h is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ we write $h : \mathbf{A} \to \mathbf{B}$.*

Note that we previously used this notation to indicate a function between two sets. They are not the same since h is not a function but a type-indexed collection of functions, but the analogy is suggestive enough that it seems helpful to overload notation in this instance.

In order to familiarize yourselves with homomorphisms it's worth trying the following exercise:

Exercise 14.17. *Suppose $h : \mathbf{A} \to \mathbf{B}$ and $g : \mathbf{B} \to \mathbf{C}$ are homomorphisms. Define a homomorphism $g \circ h$ by setting $(g \circ h)^\sigma = g^\sigma \circ h^\sigma$. Show that $g \circ h$ is a homomorphism.*

Exercise 14.18. *Suppose $\sim$ is a congruence of applicative structures or $\mathcal{J}(\Sigma)$-interpretations. Show that the mapping $\pi : \mathbf{A} \to \mathbf{A}/\sim$ defined by $\pi(a) = [a]_\sim$ is a homomorphism.*

In the study of applicative structures, $\mathcal{J}(\Sigma)$-interpretations, etc. it is often necessary to consider a generalization of a homomorphism that is only partially defined.[8]

Definition 14.19 (Partial homomorphism)**.** *A partial homomorphism of applicative structures $h : \mathbf{A} \to \mathbf{B}$ is a type-indexed collection of partial functions $h^\sigma : A^\sigma \rightharpoonup B^\sigma$ such that whenever $h(f)$ and $h(a)$ are defined, $h(\mathrm{App}(f, a))$ is defined and equal to $\mathrm{App}(h(f), h(a))$.*

A partial homomorphism of interpretations, $\mathcal{J}(\Sigma)$-interpretations or combinatory interpretations over Σ furthermore $\mathbf{I}$ and $\mathbf{J}$ satisfies the condition that $h(\llbracket c \rrbracket_I)$ is defined and equal to $\llbracket c \rrbracket_J$ for every $c \in \Sigma$.

Exercise 14.19. *Show that partial homomorphisms may also be composed.*

Exercise 14.20. *Suppose $\sim$ is a partial congruence of applicative structures or $\mathcal{J}(\Sigma)$-interpretations. Show that the mapping $\pi : \mathbf{A} \to \mathbf{A}/\sim$ defined by $\pi(a) = [a]_\sim$ is a partial homomorphism.*

Every combinatory homomorphism is a λ-homomorphism. More precisely:

Theorem 14.3. *If $\mathbf{I}$ is a $\mathcal{L}(\Sigma)$-interpretation for a λ-language $\mathcal{L}(\Sigma)$, and $h : \mathbf{I} \to \mathbf{J}$ a homomorphism of $\mathcal{L}(\Sigma)$-interpretations that preserves combinators, and the interpretation of constants: $h(\llbracket c \rrbracket_I) = \llbracket c \rrbracket_J$ for each $c \in \Sigma$. Then $h(\llbracket M \rrbracket_I^g) = \llbracket M \rrbracket_J^{h \circ g}$ for arbitrary terms M and assignments g.*

Proof. Suppose that h is a combinatory homomorphism as given. Given an arbitrary term M of $\mathcal{L}(\Sigma)$ it suffices to show that $h^\sigma(\llbracket M \rrbracket_I^g) = \llbracket M \rrbracket_J^{h \circ g}$. First note that M is $\beta\eta$-equivalent to a term M' that is created from variables, constants in Σ and instances of $K^{\sigma\tau}$ and $S^{\sigma\tau\rho}$ using only application (namely $M' := (M^\dagger)^*$ from Section 3.5). Because M and M' are $\beta\eta$-equivalent and because $\mathbf{I}$ is a $\mathcal{L}(\Sigma)$-interpretation (functional or non-functional) $\llbracket M \rrbracket_I^g = \llbracket M' \rrbracket_I^g$. A straightforward proof by induction on the structure of terms shows that for any term N made from variables, constants, K and S by application, $h(\llbracket N \rrbracket_I^g) = \llbracket N \rrbracket_J^{h \circ g}$ (since h preserves the interpretations of constants, S and K, and preserves application; and clearly $h(\llbracket x \rrbracket_I^g) = \llbracket x \rrbracket_J^{h \circ g}$ for variables). Thus $h(\llbracket M' \rrbracket_I^g) = \llbracket M' \rrbracket_J^{h \circ g}$, and finally $\llbracket M' \rrbracket_J^{h \circ g} = \llbracket M \rrbracket_J^{h \circ g}$ because M and M' are $\beta\eta$-equivalent. □

Like congruences, homomorphisms are a staple of algebra and are connected to them in various ways. Every homomorphism determines a congruence:

Definition 14.20. *If $h : \mathbf{A} \to \mathbf{B}$ is a (partial) homomorphism of applicative structures, $\mathcal{J}(\Sigma)$-interpretations, etc then the* kernel *of h is the (partial) congruence relation on $\mathbf{A}$ defined as:*

- *$a \sim_h^\sigma b$ iff $h^\sigma(a)$ and $h^\sigma(b)$ are defined and $h^\sigma(a) = h^\sigma(b)$.*

Exercise 14.21. *Prove that the kernel of h is a congruence when $h : \mathbf{A} \to \mathbf{B}$ is a homomorphism, and a partial congruence when h is a partial homomorphism.*

We end this section with a couple of more exercises manipulating homomorphisms, which also serves to introduce another construction on applicative structures, products:

Definition 14.21 (Product of applicative structures)**.** *If* $\mathbf{A} = (A^{\cdot}, App^{\cdot}_{A})$ *and* $\mathbf{B} = (B^{\cdot}, App^{\cdot}_{B})$ *are applicative structures, their product,* $\mathbf{A} \times \mathbf{B} = ((A \times B)^{\cdot}, \mathrm{App}^{\cdot}_{A\times B})$*, may be defined, where*

- $(A \times B)^{\sigma} = A^{\sigma} \times B^{\sigma}$
- $\mathrm{App}^{\sigma\tau}_{A\times B}((f,g),(a,b)) = (\mathrm{App}^{\sigma\tau}_{A}(f,a), \mathrm{App}^{\sigma\tau}_{B}(g,b))$

The product of two $\mathcal{J}(\Sigma)$*-interpretations* $\mathbf{I}$ *and* $\mathbf{J}$ *is defined as above for the applicative structure with* $[\![c]\!]_{I\times J} = ([\![c]\!]_{I}, [\![c]\!]_{J})$*.*

Exercise 14.22. *Define operations* $\pi^{\sigma}_{A} : (A \times B)^{\sigma} \to A^{\sigma}$ *and* π^{σ}_{B} *where* $\pi^{\sigma}_{A}(a,b) = a$ *and* $\pi_{B}(a,b) = b$*. Show that they are homomorphisms from* $\mathbf{A} \times \mathbf{B}$ *to* $\mathbf{A}$ *and to* $\mathbf{B}$ *respectively.*

Exercise 14.23. *Suppose* $\mathbf{C}$ *is an applicative structure and* $h : \mathbf{C} \to \mathbf{A}$ *and* $h' : \mathbf{C} \to \mathbf{B}$ *are homomorphisms. Define a homomorphism,* $g : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$ *such that* $\pi_{A} \circ g = h$ *and* $\pi_{B} \circ g = h'$*. Show that if* $g' : \mathbf{C} \to \mathbf{A} \times \mathbf{B}$*,* $\pi_{A} \circ g' = h$ *and* $\pi_{B} \circ g' = h'$ *then* $g = g'$*.*

14.6 Isomorphisms

Finally, we turn to a special sort of homomorphism which tells us that two structures are effectively the same.

Definition 14.22 (Isomorphism)**.** *A homomorphism* $h : \mathbf{A} \to \mathbf{B}$ *(where* $\mathbf{A}$ *and* $\mathbf{B}$ *are applicative structures or* $\mathcal{J}(\Sigma)$*-interpretations) is an isomorphism if and only if one of the following conditions holds:*

1. h^{σ} *is a bijection for each* σ*.*
2. *There is a homomorphism* $g : \mathbf{B} \to \mathbf{A}$ *such that* $g \circ h = id_{\mathbf{A}}$ *and* $h \circ g = id_{\mathbf{B}}$*, where* $id_{\mathbf{A}} : \mathbf{A} \to \mathbf{A}$ *and* $id_{\mathbf{B}} : \mathbf{B} \to \mathbf{B}$ *are the evident identity homomorphisms.*

It is easily verified that conditions 1 and 2 are equivalent to one another (and thus equivalent to the disjunction).

Exercise 14.24. *Show that the applicative structure of Example 14.4, in which application is identified with ordered pair formation, is isomorphic to the term structure of the applicative language* $\mathcal{A}(S)$ *treating* S *as though it were a signature.*

To familiarize yourself with isomorphisms, it's worth trying the following slightly more advanced exercise.

Exercise 14.25 (The First Isomorphism Theorem)**.** *Suppose that* $h : \mathbf{A} \to \mathbf{B}$ *is a homomorphism. Define an applicative structure,* $h(\mathbf{A}) = (h(A)^{\cdot}, \mathrm{App}^{\cdot}_{h(A)})$ *where* $(h(A))^{\sigma} = \{h^{\sigma}(a) \in B^{\sigma} \mid a \in A^{\sigma}\}$ *and* $App_{h(A)}(f,a) = App_{B}(f,a)$*.*

a. *Show that* $\mathbf{A}/\sim_{h}$ *is isomorphic to* $h(\mathbf{A})$ *via an isomorphism* $j : \mathbf{A}/\sim_{h} \to h(\mathbf{A})$*.*
b. *Show* $h = j \circ \pi$ *where* $\pi : \mathbf{A} \to \mathbf{A}/\sim_{h}$ *is the homomorphism from Exercise 14.18.*
c. *Prove the analogous claims for partial homomorphisms* h *and the corresponding partial kernel.*

Henkin structures are archetypal functional structures: the relation between an element $f : A^{\sigma\to\tau}$ and its applicative behaviour, a function $A^\sigma \to A^\tau$, is just identity, as elements of $A^{\sigma\to\tau}$ just are functions. One might have thought that any functional structure is essentially the same (i.e. is isomorphic to) a Henkin structure. For instance, any element $f \in A^{t\to t}$ determines a unique function $A^t \to A^t$, namely $a \mapsto \mathrm{App}(f, a)$. If the structure is additionally functional, then this association is injective, so that $A^{t\to t}$ is isomorphic to a subset of the function $A^t \to A^t$.

Exercise 14.26.

a. In the natural number structure $\mathbf{A}$ *of Example 14.5, find the applicative behaviour—i.e. a function* $A^t \to A^t$ *(*$= \mathbb{N} \to \mathbb{N}$*)—corresponding to the element* $4 \in A^{t\to t}$.

b. Suppose $\square \in \Sigma^{t\to t}$*, and consider the term applicative structure* $\mathcal{L}(\Sigma)$*. Find the applicative behaviour of the operation* $\square$*, a function* $\mathcal{L}^t(\Sigma) \to \mathcal{L}^t(\Sigma)$.

c. More generally describe a bijection between $\mathbb{N} = A^{t\to t}$ *and a set of functions from* A^t *to* A^t*, and a bijection between* $\mathcal{L}(\Sigma)^{t\to t}$ *and a set of functions from* $\mathcal{L}^t(\Sigma)$ *to* $\mathcal{L}^t(\Sigma)$*. (Make clear the role of the assumption that both structures are functional plays in this proof.)*

One strategy for showing that a functional applicative structure, $\mathbf{A}$, is isomorphic to a Henkin structure is to use the 'applicative behaviour' correspondence, explored in the previous exercise, to generate a bijection between $\mathbf{A}$ and a Henkin structure $\mathbf{B}$, such that $B^e = A^e$, $B^t = A^t$, where $B^{t\to t}$ is the set of functions $B^t \to B^t$ that are the applicative behaviours of elements of $A^{t\to t}$. But this process cannot be straightforwardly extended to higher types. Take the term applicative structure and suppose there is an operation $F \in \Sigma^{(t\to t)\to t\to t}$. Its applicative behaviour is a function $\mathcal{L}^{t\to t} \to \mathcal{L}^{t\to t}$. But an element $f \in B^{(t\to t)\to t\to t}$ in a Henkin structure is a function from $B^{t\to t} \to B^{t\to t}$. $B^{t\to t}$ is a set of functions and $\mathcal{L}^{t\to t}$ is a set of terms, so the applicative behaviours at sufficiently complex types become different from elements of Henkin structures. This is an instance of a more general phenomenon in the theory of applicative structures which we will turn to later in our discussion of logical relations (Chapter 16).

The upshot is that, given a functional applicative structure, we have to define the desired isomorphism inductively in tandem with the definition of the Henkin structure that it will be isomorphic to.

Theorem 14.4. *Every functional applicative structure,* $\mathbf{A}$*, is isomorphic to a Henkin structure.*

Proof. Suppose that $\mathbf{A}$ is a functional applicative structure. We build a Henkin structure $\mathbf{B}$, and bijections $j^\sigma : A^\sigma \to B^\sigma$ between their domains, inductively as follows.

For the base case, we defined $B^e = A^e$ and $B^t = A^t$, and we defined $j^e : A^e \to B^e$ and $j^t : A^t \to B^t$ to be the identity functions.

Suppose B^σ and B^τ are defined, and their are bijections $j^\sigma : A^\sigma \to B^\sigma$ and $j^\tau : A^\tau \to B^\tau$. We may define a function $j^{\sigma\to\tau}$ that maps an element $f \in A^{\sigma\to\tau}$ to a function in $B^\sigma \to B^\tau$, the function $b \mapsto j^\tau(\mathrm{App}(f, (j^\sigma)^{-1}(b)))$. $B^{\sigma\to\tau}$ may be identified with the image of this function. $j^{\sigma\to\tau}$ is clearly surjective onto $B^{\sigma\to\tau}$. It is injective too: if $j^{\sigma\to\tau}(f) = j^{\sigma\to\tau}(g)$ then for any $b \in B^\sigma$, $j^\tau(\mathrm{App}(f, (j^\sigma)^{-1}(b))) = j^\tau(\mathrm{App}(g, (j^\sigma)^{-1}(b)))$. Because j^τ is injective, this can only happen if for every such b, $\mathrm{App}(f, (j^\sigma)^{-1}(b)) = \mathrm{App}(g, (j^\sigma)^{-1}(b))$. Because j^σ is surjective, every element of $a \in A^\sigma$ is of the form $j^{\sigma-1}(b)$ for some $b \in B^\sigma$. So it follows that $\mathrm{App}(f, a) = \mathrm{App}(g, a)$ for every $a \in A^\sigma$. Finally, by the functionality of A, it follows that $f = g$.

It is easily verified that j is a homomorphism and, because it is bijective at each type, an isomorphism. □

14.7 Initial structures

The term structure of a language, and its quotient by $\beta\eta$-equivalence are interesting structures from a metaphysical perspective. The former provides us a model (in some sense of model) of what a structured theory of propositions, properties and relations ought to look like.

Remark 14.3. Of course, we know that this theory cannot have a model in which the other principles of H, or H with extensional β are also true, since we know that theory to be inconsistent. But models based on applicative structures are not yet models of logical languages anyway. They are at best models of an expressively weak equational theory that purports to capture the structured vision of reality.

The term structure quotiented by $\beta\eta$-equivalence similarly might be thought to capture a slightly coarser-grained conception of reality that only makes identifications between propositions, properties and relations that are forced by β and η-equivalence (recall the structural theories discussed in Section 13.2).

In fact they also enjoy a very special property in terms of their homomorphism-theoretic relations to other structures. This provides us with an interesting perspective on the term structures and, potentially, another way of articulating what a structural model of reality should be like in abstract terms.

By a *category* of structures and homomorphisms we shall informally mean a class of structures related by homomorphisms of a certain sort. In the following we will focus on two categories of structures: (i) the category of interpretations (for a fixed signature) and arbitrary homomorphisms between them, and (ii) the category of functional $\mathcal{L}(\Sigma)$-interpretation with combinators and homomorphisms preserving combinators. The subsequent discussion can be also be generalized to non-functional structures, although we will not do so.

Definition 14.23 (Initial structure)**.**

1. *An interpretation of signature* Σ, $\mathbf{I}$ *is* initial *in the category of interpretations and homomorphisms iff for every other interpretation of the relevant sort* $\mathbf{J}$ *there is a unique homomorphism* $h : \mathbf{I} \to \mathbf{J}$.
2. *A* $\mathcal{L}(\Sigma)$*-interpretation in the signature* Σ *with combinators is initial in the category of functional* $\mathcal{L}(\Sigma)$*-interpretations with combinators and homomorphisms preserving combinators iff there's a unique model homomorphism preserving combinators from it to any other functional* $\mathcal{L}(\Sigma)$*-interpretation with combinators.*

Exercise 14.27. *Show that if* $\mathbf{I}$ *and* $\mathbf{J}$ *are initial, there is an isomorphism between them.*[iv]

First we consider typed applicative languages over the signature Σ—languages that do not contain λ or combinators, and whose terms are just built by application. We begin by showing that the term structure $\mathcal{A}(\Sigma)$ is an initial structure.

Theorem 14.5. *The term $\mathcal{A}(\Sigma)$-interpretation, of an applicative typed language $\mathcal{A}(\Sigma)$, is an initial model of signature Σ, and with unique homomorphism $[\![\cdot]\!]_I : \mathcal{A}(\Sigma) \to \mathbf{I}$, the interpretation function of* $\mathbf{I}$.

Proof. Firstly note that $[\![\cdot]\!]_I : \mathcal{A}(\Sigma) \to \mathbf{I}$ is a homomorphism, as $[\![\mathrm{App}(M,N)]\!]_I = [\![(MN)]\!]_I$ by the definition of App in the term model, and $[\![(MN)]\!]_I = App_I([\![M]\!]_I, [\![N]\!]_I)$ by the semantic clause for terms of the form (MN).

Suppose that $h : \mathcal{A}(\Sigma) \to \mathbf{I}$ is another homomorphism of models. We will prove, by induction, that h and $[\![\cdot]\!]_I$ agree on all terms, and are therefore identical. The conditions for homomorphisms ensure that $h([\![c]\!]_{\mathcal{A}(\Sigma)}) = [\![c]\!]_I$, and since $[\![c]\!]_{\mathcal{A}(\Sigma)} = c$ we have that h agrees with $[\![\cdot]\!]_I$ on the constants. Suppose for induction that $h(M) = [\![M]\!]_I$ and $h(N) = [\![N]\!]_I$. Then $h(MN) = \mathrm{App}(h(M), h(N)) = \mathrm{App}([\![M]\!]_I, [\![N]\!]_I) = [\![MN]\!]_I$. This completes the proof. □

By Theorem 14.2, we know that if $M =_{\eta\beta} N$ then $[\![M]\!]^g = [\![N]\!]^g$. It is thus possible to extend $[\![\cdot]\!]$ to act on the quotiented term model $\mathcal{L}(\Sigma)/=_{\beta\eta}$, yielding a function:

$$[\![[M]]\!]^g_{=_{\beta\eta}} = [\![M]\!]^g$$

This definition does not depend on M since for any $N \in [M]_{=_{\beta\eta}}$, $[\![M]\!]^g = [\![N]\!]^g$.

If you did Exercise 14.16 you will have noticed that when Σ is empty the quotiented closed-term structure is not functional. For instance, there are infinitely many $\beta\eta$-inequivalent elements of type $(t \to t) \to t \to t$, of the form $\lambda X \lambda p.X(X(\ldots Xp)\ldots)$ but only one element of type $t \to t$, $\lambda p.p$. It's easily observed that

Theorem 14.6. *The $\beta\eta$-term interpretation, of a λ-language $\mathcal{L}(\Sigma)$, $\mathcal{L}(\Sigma)/=_{\beta\eta}$ is an initial functional interpretation of signature Σ, and the unique combinator preserving homomorphism $h : \mathcal{L}(\Sigma)/=_{\beta\eta} \to \mathbf{I}$ is the quotiented interpretation function of* $\mathbf{I}$, $[\![\cdot]\!]^I_{=_{\beta\eta}}$.

Exercise 14.28. *Show that $[\![\cdot]\!]^I_{=_{\beta\eta}} : \mathcal{L}(\Sigma)/=_{\beta\eta} \to \mathbf{I}$ is a combinator preserving homomorphism.*

Exercise 14.29. *Show that any other homomorphism $h : \mathcal{L}(\Sigma)/=_{\beta\eta} \to \mathbf{I}$ preserving combinators must be identical to $[\![\cdot]\!]^I_{=_{\beta\eta}}$.*

Endnotes

1. Named after Leon Henkin (Henkin, 1950).
2. See Brown (2007) for an overview; they derive from the V-complexes introduced in Andrews (1971) for proving cut-elimination. For a philosophical application of a similar construction see the appendix to Dorr (2016).
3. Provided the domains A^σ are pairwise disjoint, the superscripts are uniquely determined by the types of their arguments, so that $\mathrm{App}(d,a)$ is short for $\mathrm{App}^{\sigma\tau}(d,a)$ when $d \in A^{\sigma\to\tau}$ and $a \in A^\sigma$.
4. Consider the unique non-trivial case in which $M = \lambda y.N$. Since y is not free in $\lambda y.N$, $g[a/y][\overline{h(x)}/\overline{x}]$ is well-defined for all a of the correct type. Also note that $g[a/y][\overline{h(x)}/\overline{x}]$ and $g[\overline{h(x)}/\overline{x}][a/y]$ are the same function. So $a \mapsto [\![N]\!]^{h[a/y]}$ and $a \mapsto [\![N]\!]^{g[\overline{h(x)}/\overline{x}][a/y]}$ are the same function according to I.H. Since the underlying structure is functional. $[\![\lambda y.N]\!]^h = [\![\lambda y.N]\!]^{g[\overline{h(x)}/\overline{x}]}$.
5. If x is some y_i, $g[\overline{a}/\overline{y}][b/x] = g[a_1/y_1, \ldots, a_{i-1}/y_{i-1}, b/y_i, a_{i+1}/y_{i+1}, \ldots, a_n/y_n]$; otherwise, $g[\overline{a}/\overline{y}][b/x] = g[\overline{a}/\overline{y}, b/x]$.
6. Definitions of a model of this form trace back to Meyer (1982).
7. See, for instance, Mitchell (1996), or Hindley and Seldin (2008).
8. It is worth noting that the definition beneath is distinct from the notion defined in Friedman (1975b) which was also required to be surjective at each type.

Hints for exercises

[i] **Hint:** Consider the discussion in Sections 3.5 and 3.6.
[ii] **Hint:** Think about the full Henkin structure with $A^t = \mathbb{N}$ as a $\mathcal{L}(\Sigma)$-interpretation of the empty signature. Consider applying the interpretations of different Church numerals to the successor function.
[iii] **Hint:** Consider the subset $\{a \sim a \mid a \in X\}$ of X.
[iv] **Hint:** Use the first definition of an isomorphism, and think about what the unique homomorphism from $\mathbf{I} \to \mathbf{I}$ must be.

15

Models of higher-order languages

Throughout this book, we have treated the language of higher-order logic, $\mathcal{L}(\Lambda)$, as an interpreted language. Its sentences can be evaluated as true or false *simpliciter*, and we have used them to make claims, conjectures and so forth. Sometimes, however, it is useful to take the language of higher-order logic and assign it non-intended interpretations constructed from mathematical objects—i.e. *models*—for which we can introduce, within the language of set-theory, properties like *being true in* $\mathbf{M}$ for each model $\mathbf{M}$. Sometimes when we are using a sentence of higher-order logic to make a conjecture or claim, we would like some guarantee that the sentence is consistent, in the sense that one cannot prove everything from the standard axioms. If a higher-order logic is *sound* with respect to a class of models then consistency can be secured by finding a model in which the sentence is true. If the logic is *complete* with respect to that class of models, then one can know that there will exist such a model if the sentence is indeed consistent. In Section 15.1, a general class of models for this purpose are described, and in Sections 15.2 and 15.3 the soundness and completeness results are proved. Section 15.4 resolves a putative tension between the inconsistency of the quasi-syntactic structured view of reality (Section 11.1) and the existence of very fine-grained models of higher-order languages. In the final two sections, we return to the study of the interpreted language of higher-order logic. Section 15.5 attends to the question of how to theorize about the intended interpretation of the language of higher-order logic. The question of soundness and completeness for uninterpreted languages is fairly uninteresting: according to the results in this chapter every higher-order logic is sound and complete with respect to *some* class of models. Nonetheless we might ask whether it is possible to provide a recursive axiomatization of the *logical truths*—a notion that applies only to an interpreted language and can be defined from truth simpliciter. In Section 15.6, we employ Gödel's incompleteness results to show that the logical truths of higher-order logic cannot be recursively axiomatized.

15.1 General models of higher-order logic

In Chapter 14, we introduced the notion of an applicative structure, and showed how, when the structure had combinators, we could define an interpretation function assigning denotations to each term of a λ-language, either by a constraint or compositionally, depending on whether the underlying applicative structure is functional.

Models of this sort are often studied in the context of expressively weak fragments of higher-order logic called *equational theories*: sets of equations of the form $M = N$ where M and N have the same type.[1] But in order to introduce a notion of a model for a higher-order theory, such as H, we need to be able say when an arbitrary sentence—term of type

DOI: 10.4324/9781003039181-20

t—is true or false. So we need to augment our notion of model with a new gadget telling us when propositions—elements of A^t—are true or false, which we can achieve via a function $v : A^t \to \{0, 1\}$.

Since we are especially interested in models of H, we will restrict attention to models that are subject to further constraints ensuring that the interpretations of the quantifiers and connectives behave correctly.[2]

Definition 15.1 (General model of higher-order logic)**.** *Suppose that $\Sigma \supseteq \Lambda$ is a logical signature. A general model of $\mathcal{J}(\Sigma)$ is a triple $(\mathbf{A}, [\![\cdot]\!], v)$ where*

- *$(\mathbf{A}, [\![\cdot]\!])$ is a (functional or non-functional) $\mathcal{J}(\Sigma)$-interpretation in an applicative structure $\mathbf{A}$.*
- *$v : A^t \to \{0, 1\}$.*

subject to the further constraint that

$v([\![\top]\!]) = 1$
$v([\![\bot]\!]) = 0$
$v(\mathrm{App}(\mathrm{App}([\![\vee]\!], p), q)) = \max(v(p), v(q))$ *for every* $p, q \in A^t$
$v(\mathrm{App}(\mathrm{App}([\![\wedge]\!], p), q)) = \min(v(p), v(q))$ *for every* $p, q \in A^t$
$v(\mathrm{App}(\mathrm{App}([\![\to]\!], p), q)) = \max(1 - v(p), v(q))$ *for every* $p, q \in A^t$
$v(\mathrm{App}(\mathrm{App}([\![\leftrightarrow]\!], p), q)) = |v(p) + v(q) - 1|$ *for every* $p, q \in A^t$
$v(\mathrm{App}([\![\neg]\!], p)) = 1 - v(p)$ *for every* $p \in A^t$
$v(\mathrm{App}([\![\forall_\sigma]\!]), f)) = \min_{a \in A^\sigma} v(\mathrm{App}(f, a))$ *for every* $f \in A^{\sigma \to t}$
$v(\mathrm{App}([\![\exists_\sigma]\!]), f)) = \max_{a \in A^\sigma} v(\mathrm{App}(f, a))$ *for every* $f \in A^{\sigma \to t}$
$v(\mathrm{App}(\mathrm{App}([\![=_\sigma]\!], a), b)) = v(\mathrm{App}(\mathrm{App}([\![\lambda xy.\forall_{\sigma \to t}.(Zx \leftrightarrow Zy)]\!], a), b)$ *for all* $a, b \in A^\sigma$

When Σ is a signature containing only some of the logical constants, we consider a higher-order model of Σ to be a model satisfying the above constraints corresponding to the logical constants in the signature, with the additional constraint that:

There is a false proposition: $v(p) = 0$ *for some* $p \in A^t$

We have also considered languages with extended logical signatures in this book (which are especially relevant when we consider a general λ-language $\mathcal{J}(\Sigma)$). When considering languages with further logical constants we need to extend the conditions on v accordingly. Below I describe how to treat the hatted quantifiers from Section 9.4. Let us write $\mathrm{App}(f, \bar{a})$ for $\mathrm{App}(\ldots \mathrm{App}(\mathrm{App}(f, a_1), a_2), \ldots), a_n)$. Recall that $\boldsymbol{\sigma}$ refers to a hatted sequence of types, where a hat appears over instances of only one kind of type, and $\check{\boldsymbol{\sigma}}$ is the result of deleting all the hatted types from $\boldsymbol{\sigma}$. Given $\bar{a}$ belong to domains of type $\check{\boldsymbol{\sigma}}$ and $\bar{b}$ in domains of type $\boldsymbol{\sigma}$, and suppose τ is the hatted type in $\boldsymbol{\sigma}$: we write $\bar{a} \preceq \bar{b}$ to mean that there is some element $d \in A^\tau$ such that $\bar{a}$ can be filled out to make $\bar{b}$ by inserting d into $\bar{a}$ at the positions of the sequence that are hatted in $\boldsymbol{\sigma}$. Then the model conditions of the hatted quantifiers may be stated simply:

$v(\mathrm{App}(\mathrm{App}([\![\forall_{\boldsymbol{\sigma}}]\!]), f), \bar{a}) = \min_{\bar{a} \preceq \bar{b} \in A^{\boldsymbol{\sigma}}} v(\mathrm{App}(f, \bar{b}))$ for every $f \in A^{\boldsymbol{\sigma} \to t}$

$v(\mathrm{App}(\mathrm{App}([\![\exists_{\boldsymbol{\sigma}}]\!]),f),\bar{a}) = \max_{\bar{a}\preceq\bar{b}\in A^{\boldsymbol{\sigma}}} v(\mathrm{App}(f,\bar{b}))$ for every $f \in A^{\boldsymbol{\sigma}\to t}$

When the quantifiers are treated syncategorematically (Section 10.3) we can no longer formulate the constraint as a constraint on the interpretations of the logical constants, it must be a global constraint on the whole language:

$v(\mathrm{App}([\![\forall_\sigma x.A]\!]^g) = \min_{a\in A^\sigma} v([\![A]\!]^{g[a/x]})$
$v(\mathrm{App}([\![\exists_\sigma x.A]\!]^g) = \max_{a\in A^\sigma} v([\![A]\!]^{g[a/x]})$

The list of conditions above constrain the logical constants so as to have appropriate logical behavior. Because we have adopted a signature of logical constants which is highly redundant, at least with respect to extensional behaviour[3], it can be instructive to consider what the clauses mean for a smaller signature that doesn't have redundancy, such as $\to$ and $\forall_\sigma$, from which the other logical constants can be defined up to extension. The clause for $[\![\to]\!]$ is equivalent to

$v(\mathrm{App}(\mathrm{App}([\![\to]\!],p),q)) = 1$ iff $v(p) = 0$ or $v(q) = 1$

in other words, the usual clause for the truth-functional conditional. The clause for $[\![\forall_\sigma]\!]$ is equivalent to the following:

$v(\mathrm{App}([\![\forall_\sigma]\!]),f)) = 1$ iff $v(\mathrm{App}(f,a))$ for every $a \in A^\sigma$ (and $= 0$ otherwise)

That is to say, the interpretation of the universal quantifier applies to an element $f \in A^{\sigma\to\tau}$, interpreting a predicate of type σ things, just in case f is satisfied by every element of the domain A^σ.

When considering limited signatures, like $\to$ and $\forall_\sigma$, the further constraint that v is surjective becomes necessary, otherwise one can find models in which $v([\![\forall_t p.p]\!]) = 1$. It is redundant in many signatures, such as any signature containing $\bot$.

To develop an intuition for general models let us consider two particular examples (in the minimal logical signature consisting of $\to$ and $\forall_\sigma$ for each σ).

Example 15.1 (Fregean models)**.** *Let* $\mathbf{A}$ *be a full Henkin structure in which* $A^t = \{0,1\}$*. That is:*

$A^t = \{0,1\}$
A^e *any non-empty set.*
$A^{\sigma\to\tau} = A^\sigma \to A^\tau$

It may be extended to a general model as follows:

$v : \{0,1\} \to \{0,1\}$ *is the identity function.*
$[\![\to]\!] = x \mapsto (y \mapsto \max(1-x,y))$
$[\![\forall_\sigma]\!] = f \mapsto \min_{a\in A^\sigma}(f(a))$

In Fregean models there are only two propositions: the true and the false (represented by 1 and 0 respectively). v just tells us that the true is true and the false is false. $\to$ is interpreted as

the evident truth function. $\forall_\sigma$ maps a function $f : A^\sigma \to \{0, 1\}$ to 1 if f maps every argument to 1, and 0 otherwise. Evidently a Fregean model is a model, as the conditions for $\to$ and $\forall_\sigma$ are immediate: for instance $v(\mathrm{App}([\![\forall_\sigma]\!]), f)) = \min_{a \in A^\sigma}(f(a)) = \min_{a \in A^\sigma} v(\mathrm{App}(f, a))$.

Example 15.2 (Full possible worlds models)**.** *Let W be a fixed non-empty set—the 'worlds'—and $@ \in W$—the 'actual' or 'designated' world.*

Let **A** *be the full Henkin structure in which $A^t = P(W)$. That is:*

$A^t = P(W)$
A^e *any non-empty set.*
$A^{\sigma\to\tau} = A^\sigma \to A^\tau$

It may be extended to a general model as follows:

$v : P(W) \to \{0, 1\}$ *where* $v(p) = 1$ *iff* $@ \in p$.
$[\![\to]\!] = p \mapsto (q \mapsto (W \setminus p) \cup q)$
$[\![\forall_\sigma]\!] = f \mapsto \bigcap_{a \in A^\sigma}(f(a))$

This example represents a fairly common way of modeling propositions, properties and relations typified by philosophers like Robert Stalnaker and David Lewis. According to the possible worlds picture a proposition is identified with a set of possible worlds, or equivalently, a function from worlds to truth values. Properties and relations are given a similarly intensional interpretation. Usually an $e \to t$ property is represented by a function from possible worlds to extensions (subsets of A^e). In the above model, a property is modeled instead as a function from individuals to sets of possible worlds. For instance, *is tall* will pick out the function that maps an individual a to the set of worlds where a is tall. One can translate back and forth between the two representations as follows.

- If $f : W \to P(A^e)$ define $g : A^e \to P(W)$ by $g(a) = \{w \mid a \in f(w)\}$
- For $g : A^e \to P(W)$ define $f : W \to P(A^e)$ by $f(w) = \{a \mid w \in g(a)\}$

Notice that our model is simplistic in several respects. For one thing, we have chosen to include all functions from individuals to sets of worlds as candidate properties: we have chosen a *full* Henkin structure. But there are metaphysical pictures in which only some such functions correspond to genuine properties. For instance: there is a function, f, that maps Alice to the set of worlds where Bob is tall, and Bob to the set of worlds containing electrons. But one might maintain that no property has this applicative behaviour, for it conflicts with two natural principles:

1. Applying a property to Alice ought to result in a proposition that says something *about Alice*. Intuitively the set of worlds where Bob is tall is not about Alice, whereas by contrast the set of worlds where Alice is tall is. More generally, it will not say something about an unrelated person, such as Bob, unless Bob was involved in the property being ascribed.
2. A property ought to ascribe the same property to every argument it is applied to: no property ascribes tallness to Alice when applied to Alice, and baldness to Bob when applied to Bob.

While it is a subtle matter whether these two constraints can be satisfied (or even given precise statements) within a possible world framework, they certainly cannot be satisfied if fullness is imposed.[4]

Second, we have assumed that there is no contingent existence. One can concoct more complicated models of higher-order logic in which things exist contingently, although we will bracket those complications for now.[5]

Exercise 15.1. *Show that a full possible worlds model is a model. I.e. show*

a. $v((\mathrm{App}([\![\rightarrow]\!], p), q) = \max(1 - v(p), v(q))$ *for every* $p, q \in A^t$

b. $v(\mathrm{App}([\![\forall_\sigma]\!]), f)) = 1$ *iff* $v(\mathrm{App}(f, a))$ *for every* $a \in A^\sigma$ *(and* $= 0$ *otherwise)*

15.2 Soundness

A model tells us which elements of A^t are true or false, and thus tells us indirectly which *sentences* are true (and which formulae are satisfied by variable assignments).

Definition 15.2. *A formula* $A : t$ *is satisfied by an assignment* g *in a model* $\mathbf{M}$*, written* $\mathbf{M}, g \models A$ *iff* $v([\![A]\!]^g) = 1$.

A formula $A : t$ *is* true *in a model* $\mathbf{M}$*, written* $\mathbf{M} \models A$ *iff* $\mathbf{M}, g \models A$*, for every assignment* g.

Observe that we have defined the truth of an open formula in a model to be equivalent to the truth of its universal closure.

We can see now that the propositional tautologies, the quantifier axiom UI, and instances of β and η are true in any given model.

Proposition 15.1. $\mathbf{M}, g \models \forall_\sigma F \rightarrow Fa$ *for any higher-order model* $\mathbf{M}$ *and assignment* g.

Proof. We must show that $v([\![\forall_\sigma F \rightarrow Fa]\!]^g) = 1$. Suppose that $v([\![\forall_\sigma F]\!]^g) = 1$. It suffices to establish that $v([\![Fa]\!]^g) = 1$. By the clause for application we have that $v(\mathrm{App}([\![\forall_\sigma]\!]^g, [\![F]\!]^g)) = 1$. So by the constraint on $[\![\forall_\sigma]\!]$ it follows that $v(\mathrm{App}([\![F]\!]^g, d)) = 1$ for every $d \in A^\sigma$, and thus in particular that $v(\mathrm{App}([\![F]\!]^g, [\![a]\!]^g)) = 1$. So $v([\![Fa]\!]^g) = 1$, as required. □

We see now A^σ playing two roles: as in Chapter 14, it is the set from which the denotations of constants and other expressions are selected, and additionally it serves as the set over which the quantifiers of type σ range. As the above argument illustrates, having these two roles played by the same set ensures that the principle of universal instantiation is true in any model. Someone interested in rejecting universal instantiation in response to the Russell-Myhill paradox (e.g. a ramifier) might attempt to tease apart the two roles.

Proposition 15.2. $\mathbf{M}, g \models A$ *whenever* A *is an instance of a tautology.*

This follows the usual proof of the soundness of the propositional calculus with respect to truth functions.

Proposition 15.3. $\mathbf{M}, g \models A \rightarrow B$ *whenever* A *and* B *are* $\beta\eta$*-equivalent.*

We know that $[\![A]\!]^g = [\![B]\!]^g$ whenever A and B are $\beta\eta$-equivalent. So $v([\![A]\!]^g) = 1$ only if $v([\![B]\!])^g = 1$, and thus $v([\![A \rightarrow B]\!]^g) = 1$.

An instructive exercise.

Exercise 15.2. *Suppose* $\mathbf{M} = (\mathbf{A}, [\![\cdot]\!], v)$*. Show that* $\mathbf{M}, g \models \forall x.C$ *if and only if* $\mathbf{M}, g[a/x] \models C$ *for every* $a \in A^\sigma$*.*[i]

Theorem 15.1 (Soundness)**.** *Suppose $\mathcal{C}$ is a class of higher-order models. Then the set $T = \{A \mid \mathbf{M} \models A$, for all $\mathbf{M} \in \mathcal{C}\}$ is a higher-order theory.*

Proof. A higher-order theory contains all tautologies, instances of universal instantiation and $\beta\eta$, and is closed under Modus Ponens and Gen. The previous propositions establish that tautologies, universal instantiation and $\beta\eta$ are true in all models, and thus belong to T.

It remains to show T is closed under Modus Ponens and Gen. If $A \to B \in T$ and $A \in T$, then (i) $v([\![A \to B]\!]^g) = 1$ and (ii) $v([\![A]\!]^g) = 1$ for each model $\mathbf{M} \in \mathcal{C}$. Given any particular model and assignment it follows that $v([\![B]\!]^g) = 1$ since, by (i), $v([\![A]\!]^g) = 0$ or $v([\![B]\!]^g) = 1$. But the former option cannot obtain by (ii), so $v([\![B]\!]^g) = 1$ as required.

Suppose now that $A \to B \in T$ and $x \notin FV(A)$. So in any particular model, $v([\![A \to B]\!]^g) = 1$ for every assignment g. It suffices to show that $v([\![A \to \forall x.B]\!]^g) = 1$. Suppose $v([\![A]\!]^g) = 1$. Since $x \notin FV(A)$, $[\![A]\!]^{g[a/x]} = [\![A]\!]^g$ for arbitrary a so $v([\![A]\!]^{g[a/x]}) = 1$. Since $A \to B$ is true in $\mathbf{M}$ for every assignment it follows that $v([\![B]\!]^{g[a/x]}) = 1$ for arbitrary a, so by Exercise 15.2 $v([\![\forall x.B]\!]^g) = 1$. □

15.3 Completeness

In this section, we will prove the completeness of general models for our notion of a theory. If T is a consistent theory then T has a general model $\mathbf{M}$, in which all the members of T are true. The proof follows a familiar strategy, due to Henkin[6], for proving completeness theorems: first, you extend T to a "negation complete" and "witness complete" set, and second you construct a term structure using the resulting theory. The second stage is in effect the operation of quotienting the term structure we encountered in Section 14.4. The mapping of a term to its equivalence class provides us with an interpretation function, and the negation complete and witness complete extension of T determines the valuation, allowing us to turn the term structure into a model.

Recall that a theory is any set of sentences containing (i) the propositional tautologies, (ii) the quantifier axiom UI, (iii) β and η, and (iv) is closed under the rules of Modus Ponens and Gen. Recall that the notion of a theory is formulated in the signature Λ^- which has only $\to$ and $\forall_\sigma$ as primitives—symbols like $\neg$ and $\exists_\sigma$ in the below are treated as abbreviations. Entirely analogous strategies for proving completeness are available for logics suitable for the richer logical signatures considered in Section 5.2.

A set of sentences X proves A when A belongs to any theory containing X. X is consistent when it does not prove $\bot$. We state without proof some standard proof-theoretic properties, that are familiar from first-order logic. First, as we noted in Section 5.1, X proves A iff there is a proof of A from X: a finite sequence of formulas where each is either an element of X, an instance of an axiom of H, or follows from previous formulas by one of the rules of H. The second is the deduction theorem:

Theorem 15.2 (The deduction theorem)**.** *If there is a proof of B from $X \cup \{A\}$ then there is a proof of $A \to B$ from X.*

The deduction theorem is established, roughly, by showing that one can transform a proof of B from $X \cup \{A\}$ into a proof of $A \to B$ from X by prefixing $A \to$ to each step of the original proof.[7]

Corollary 15.1. *If $X \cup \{\neg A\}$ is inconsistent then X proves A.*

Recall that $\neg A$ is defined as $A \to \bot$ in this signature, so the deduction theorem tells us that if $X \cup \{\neg A\}$ proves $\bot$ then X proves $\neg\neg A$, and thus also A.

Definition 15.3 (Negation completeness)**.** *A set $T \subseteq \mathcal{L}^t(\Sigma)$ of formalae in a logical language ($\Lambda \subseteq \Sigma$) is negation complete iff, for every formula A, either $A \in T$ or $\neg A \in T$.*

Definition 15.4 (Witness completeness)**.** *A set $T \subseteq \mathcal{L}^t(\Sigma)$ of formulae in a logical language ($\Lambda \subseteq \Sigma$) is witness complete iff, for every predicate $F : \sigma \to t$, either $\neg\exists_\sigma x.Fx \in T$ or $Fa \in T$ for some term $a : \sigma$.*

We say that a set is σ-witness complete iff the above holds for a particular type σ.

Proposition 15.4. *Let Σ be a signature and Σ^+ the signature obtained by adding infinitely many new constants to Σ^σ for each type σ.*

Every consistent theory T of $\mathcal{L}(\Sigma)$ can be extended to a consistent witness complete theory $T^+ \supseteq T$ in $\mathcal{L}(\Sigma^+)$.

Proof. We'll find a σ-witness complete set for a fixed σ. The construction can be repeated for each σ to achieve the desired set.

We enumerate the predicates $\sigma \to t$ of $\mathcal{L}(\Sigma^+)$ $F_0, F_1, \ldots$ and enumerate the new constants of type σ $a_0, a_1, \ldots$ in such a way that a_{n+1} doesn't appear in $F_0, \ldots, F_n$.[8] Define a sequence of sets as follows:

$$T_0 = T$$
$$T_{n+1} = \begin{cases} T_n \cup \{F_n a_n\} & \text{if } T_n \cup \{F_n a_n\} \text{ is consistent} \\ T_n \cup \{\neg\exists_\sigma x.F_n x\} & \text{otherwise} \end{cases}$$

It's easy to see, in general, that if X is consistent and a doesn't appear in X then either $X \cup \{Fa\}$ or $X \cup \{\neg\exists x.Fx\}$ is consistent: if neither were then by Corollary 15.1 X would prove $\neg Fa$ and $\exists x.Fx$. But if there's a proof of $\neg Fa$ from X, one can make a proof of $\neg Fx$ from X by substituting a for a variable x (because X doesn't involve a, then proof will not depend on any specific properties of the term a.) And so by using Gen, we derive $\forall_\sigma x\neg Fx$ from X. This contradicts the assumption that X is consistent, since X also entails $\exists x.Fx$.

Note that a_n is a new constant that doesn't appear in T or in T_m for $m < n$, so by the above reasoning $T_n \cup \{F_n a_n\}$ is consistent or $T_n \cup \{\neg\exists_\sigma x.F_n x\}$ is.

Let T^+ be the derivable consequences of $\bigcup_n T_n$. If T^+ were inconsistent there would be a proof of $\bot$ from a finite subset X of T^+, which would have to be contained in some T_n for a particular n. This contradicts the assumption that T_n is consistent.

Moreover T^+ is witness complete. If $\neg\exists_\sigma x.F_n x \notin T^+$ then $T_{n+1} = T_n \cup \{F_n a_n\} \subseteq T^+$, so $F_n a_n \in T^+$. That is, F_n has a witness. □

Exercise 15.3. *Suppose T is a consistent set of sentences of $\mathcal{L}(\Sigma)$, and enumerate the sentences this language $A_0, A_1, \ldots$. Define*

- $T_0 = T$
- $T_{n+1} = \begin{cases} T_n \cup \{A_n\} & \textit{if } T_n \cup \{A_n\} \textit{ is consistent} \\ T_n \cup \{\neg A_n\} & \textit{otherwise} \end{cases}$

a. Show that T_{n+1} is consistent if T_n is, and conclude that each T_n is consistent.

b. Show that $T^+ = \bigcup_n T_n$ *is consistent.*

c. Show that T^+ *is negation complete.*

Negation complete consistent theories are important because they are closely related to models of the propositional calculus. In particular, if T is a consistent negation complete theory then the characteristic function of T—the function $v(A) = 1$ if $A \in T$ and $v(A) = 0$ when $A \notin T$—is a model of the propositional calculus. This is spelled out in the next exercise.

Exercise 15.4. *Suppose that T is a consistent negation complete set, and v is the characteristic function of T, as defined above. Show:*

a. $v(A \wedge B) = min(v(A), v(B))$.

b. $v(A \vee B) = max(v(A), v(B))$.

c. $v(\neg A) = 1 - v(A)$.

Witness complete sets ensure a similar thing for quantifiers. Suppose T is a witness complete theory and v its characteristic function. Then $v(\exists_\sigma F) = 1$ if and only if $v(Fa) = 1$ for some term a, mimicking the quantifier condition for a general model. The left to right direction follows from witness completeness: $\exists_\sigma F \in T$ only if $Fa \in T$ for some term $a : \sigma$. The right to left direction from the fact that T is a theory: if $Fa \in T$ then its consequence $\exists_\sigma F$ is also in T.

So it looks like the following ought to be true. Whenever T is a consistent witness complete and negation complete theory over a language $\mathcal{L}(\Sigma)$, and v its characteristic function, then the term structure equipped with v as its valuation, $(\mathcal{L}(\Sigma), [\![\cdot]\!], v)$, is a general model. There is one wrinkle however: the term structure is not an applicative structure with combinators, and so according to our definitions not even a $\mathcal{L}(\Sigma)$-interpretation. However, this wrinkle is easily smoothed over: in Chapter 14 we saw how to quotient a term structure so that $\beta\eta$-equivalent terms got identified. In the present context, it's possible to quotient by an even stronger notion of equivalence: $M \sim N$ whenever $M =_\sigma N \in T$. Certainly $T \vdash M =_\sigma N$ whenever M and N are $\beta\eta$-equivalent.

Remark 15.1. This quotienting procedure is a technicality that could be avoided if we had relaxed our notion of an interpretation, and thus a model, of a λ-language slightly. In the environment model condition, we imposed the constraint $[\![M]\!]^g = [\![N]\!]^h$ when $M \sim_{\beta\eta} N$ and h and g agree on $FV(M) \cap FV(N)$, but we could have replaced this identity with the condition that $[\![M]\!]^g$ and $[\![N]\!]^h$ are instead Leibniz equivalent.

A similar modification can be in our definition of a functional interpretation. Say that $k \in A^{\sigma\to\tau\to\sigma}$ is a quasi-K-combinator if $\mathrm{App}(\mathrm{App}(k, a), b)), a)$ is merely Leibniz equivalent to a. A similar notion of a quasi-S-combinator can be defined: an element s which the model thinks has the applicative behaviour of characteristic of the S combinator. The weaker condition of containing quasi-combinators suffice for interpreting λ-terms and for the truth of the $\beta\eta$ principle of H, and the requirement that a model only have quasi-combinators would remove the need to quotient in the above argument.

With this in place, we may finally state and prove the model existence theorem.

Theorem 15.3. *If T is a consistent theory of* $\mathcal{L}(\Sigma)$*, then T has a general model* $\mathbf{M}$ *in which every sentence of T is true.*

Proof. By putting the previous proposition and exercise together we can extend T to a consistent negation and witness complete set in the extended language $\mathcal{L}(\Sigma^+)$ (with infinitely

many witnessing constants added) by first extending T to a consistent witness complete set, and then to a consistent negation complete set T^+.

Define an equivalence relation $\sim_\sigma$ on terms of $\mathcal{L}(\Sigma^+)$ by $M \sim N$ iff $M =_\sigma N \in T^+$. It is easily seen to be a congruence, as T^+ is closed under Leibniz's law. We build a model $(\mathbf{A}, [\![\cdot]\!], v)$ of T^+ as follows:

$\mathbf{A}$ is the quotient structure $\mathcal{L}(\Sigma^+)/\sim$. I.e

$$A^\sigma = \{[M]_\sim \mid M \in \mathcal{L}^\sigma(\Sigma^+)\}$$
$$\mathrm{App}^{\sigma\tau}([M]_\sim, [N]_\sim) = [MN]_\sim$$

$[\![M]\!]^g = [M[N_1/x_1, \ldots, N_n/x_n]]_\sim$ for some $N_1, \ldots, N_n$, where $a_i = g(x_i)$ and $N_i \in a_i$

$$v([A]_\sim) = \begin{cases} 1 & \text{if } A \in T^+ \\ 0 & \text{otherwise} \end{cases}$$

It is routine to check that $(\mathbf{A}, [\![\cdot]\!])$ is a $\mathcal{L}(\Sigma)$-interpretation. (We have checked most of this in the previous chapter. The only somewhat tricky case occurs in the case that $\mathbf{A}$ is not functional, in which case we must check the environment model conditions obtain.)

Showing $v(\mathrm{App}(\mathrm{App}([\vee]_\sim, [A]_\sim), [B]_\sim)) = \max(v([A]_\sim), v([B]_\sim))$, and the related clauses for the other connectives, is routine, and follows from the negation completeness and consistency of T^+.

Showing that $v(\mathrm{App}([\exists_\sigma]_\sim, [F]_\sim)) = 1$ iff there is a term $[a]_\sim$ such that $v(\mathrm{App}([F]_\sim, [a]_\sim)) = 1$ follows from the witness completeness of T^+. The clauses for $\forall_\sigma$ follow straightforwardly.

The clause for identity follows because $a =_\sigma b \leftrightarrow \forall_{\sigma\to\iota} Z(Za \leftrightarrow Zb)$ is a theorem of H, and thus belongs to T. □

Corollary 15.2 (Completeness)**.** *If A is true in every model of T then T proves A*

If $T \nvdash_H A$ then $T \cup \{\neg A\}$ is consistent, and has a model by Theorem 15.3. So A is not true in every model of T.

15.4 The interpretation of identity and granularity

We introduced the name *Leibniz equivalence* for the relation $\lambda xy.\forall_{\sigma\to\iota} Z(Zx \leftrightarrow Zy)$. There is also a model-theoretic version of this condition, which we shall also call Leibniz equivalence when no ambiguity arises.

Definition 15.5 (Leibniz equivalent elements of a model)**.** *If $\mathbf{M} = (A, [\![\cdot]\!], v)$ is a general model, and $a, b \in A^\sigma$ then we say that a and b are Leibniz equivalent iff:*

For every $f \in A^{\sigma\to\iota}$, $v(\mathrm{App}(f, a)) = 1$ if and only if $v(\mathrm{App}(f, b)) = 1$

Write $a \sim_{LE} b$ to mean that a and b are Leibniz equivalent.

Exercise 15.5. *Suppose $\mathbf{M} = (\mathbf{A}, [\![\cdot]\!], v)$ is a general model. Show that Leibniz equivalence is a congruence on A, and that if $p, q \in A^\iota$ are Leibniz equivalent $v(p) = v(q)$.*

There is a *prima facie* tension between our discussion in Chapter 11 of the Russell-Myhill paradox and our discussion in Chapter 14 of term structures. On the one hand, we have

shown that within H, and even H_0 with $\beta\eta$ weakened to extensional β, one can prove the inconsistency of certain principles stating that reality is structured like a full λ-language. In Section 13.2, we also discussed the inconsistency of the view that reality is structured like the full λ-language quotiented by $\beta\eta$-equivalence. On the other hand, in Chapter 14 we found we could easily construct extremely fine-grained applicative structures in which the elements of the domain are literally terms of a language, such as the term structures of Example 14.2. We also constructed applicative structures where the elements were simply equivalence classes of terms related by $\beta\eta$-equivalence. One can extend both sorts of term structures into general models (or at least, the quasi-models of Remark 15.1) by endowing them with a valuation. Indeed we'll explore an example of this in an exercise below. Don't these fine-grained applicative structures establish the consistency of the fine-grained picture of reality, contrary to the Russell-Myhill theorem?

The conflict is easily resolved. A *model* $\mathbf{M}$ may be extremely fine-grained in the sense that different sentences A and B (and more generally, different terms) receive different interpretations $[\![A]\!] \neq [\![B]\!]$. However, that does not mean that the model 'thinks' that propositions are fine-grained unless the model also makes the corresponding non-identity claim true: $\mathbf{M} \models A \neq_t B$. If $[\![A]\!]$ and $[\![B]\!]$ are distinct but Leibniz equivalent then the model may think that A and B are identical.

Indeed, it is easy to see that we can equip a term structure with a valuation in such a way as to make any consistent theory of propositional granularity in that language true. The result will not be a model in the ordinary sense as we know that term structures do not contain combinators.

We can introduce a special class of models in which the models notion of identity (Leibniz equivalence) coincides with actual identity.

Definition 15.6 (Leibnizian models)**.** *A general model is* Leibnizian *if and only if distinct elements are Leibniz inequivalent. I.e. for every* σ*:*

If $a, b \in A^\sigma$ *are distinct, then there is some* $f \in A^{\sigma\to t}$ *such that* $v(\mathrm{App}(f,a)) \neq v(\mathrm{App}(f,b))$.

As one might have expected, every model can be turned into a Leibizian model by quotienting it by Leibniz equivalence.

Proposition 15.5. *Every model* $\mathbf{M}$ *can be turned into a Leibnizian model* $\mathbf{M}/\sim_{LE}$ *which makes the same sentences true.*

Most of the work for this proposition is in Exercise 15.5: Leibniz equivalence is a congruence on the underlying applicative structure of $\mathbf{M}$ which respects the valuation, so that it is possible to form a quotient. Note that once we have quotiented a fine-grained model by Leibniz equivalence we may end up with a very coarse-grained model.

With the preceding distinctions in place, we are in a position to diagnose a common mistake one can make when reasoning model-theoretically about the metaphysics of propositions. The mistaken line of thought goes like this: in order to describe a coherent picture of the structure of propositions, properties and relations it merely suffices to describe a structure which (i) delivers collections of things representing the propositions, properties, relations and so on, (ii) operations on these collections that tell us how these entities combine (such as application and λ-abstraction). The abstract formalism of an applicative structure with combinators achieves this, although other formalisms are possible.

For instance, one might naïvely justify a conception of propositions, properties and relations in which they are individuated by necessary equivalence by describing a model in which propositions are represented by sets of possible worlds, properties by functions from individuals to sets of worlds, and so on, and by showing that the operation of applying a property to an argument can be represented by function application.

Similarly, structural theories of propositions, in which propositions, properties and so on are individuated by their structural properties—are often presented in a similar manner. There are some basic or simple properties and operations in various types which do not have any proper constituents, and structured propositions and properties are represented by sequences of these basic constituents. On one popular model, the operation of applying a property, F, to an argument, a, can be presented by ordered pair formation: (F, a).

But this methodology is in general not OK unless the structures can be turned into *Leibnizian models*. For without that, no matter how the elements of the model are individuated, it will not be a model *of* the claim that propositions, etc. are individuated that way unless the model thinks distinct elements of the model are distinct. This issue is at the heart of why it turns out to be quite hard to prove the consistency of various theses about the granularity of reality.

As it turns out, the possible worlds applicative structure can be turned into a Leibnizian model as in Example 15.2.

Exercise 15.6. *Show that the possible worlds model of Example 15.2 is Leibnizian.*

But as one might expect, it is an extremely subtle matter how one can turn the model of propositions as sequences of constituents into a Leibnizian model. We explore this further in the following example.

Example 15.3. *Suppose that for each type σ, S^σ is a set informally representing the simple (unstructured) entities of type σ. Consider an applicative structure from Example 14.4 defined as the smallest type-indexed collection of sets such that:*

$S^\sigma \subseteq A^\sigma$

If $f \in A^{\sigma\to\tau}$ and $a \in A^\sigma$ then the ordered pair $(f, a) \in A^\tau$.

and define

$$\mathrm{App}^{\sigma\tau}(f, a) = (f, a).$$

We can turn this into an interpretation of a logical applicative language by supposing that $S^{(\sigma\to\iota)\to\iota}$ contains an element all_σ and some_σ that are denotations of $\forall_\sigma$ and $\exists_\sigma$, and similarly assume simple operations for the truth-functional connectives.

This applicative structure does not contain combinators so it cannot be turned into a general model. But it could be turned into a model of extensional β provided we can endow it with a valuation and find elements for each λ-term that have the right extensional behaviour. However, it is extremely hard to define a valuation on it (let alone a Leibnizian one) that satisfies the clause for the quantifiers.

Note that in the possible worlds or the Fregean theory propositions come equipped with a natural valuation, defined in terms of containing the actual world or being identical to *the true* respectively. The structured theory has no such thing. A natural thought would be to try to define v inductively, as one would define truth for a language. For instance, if $f \in A^{\iota\to\iota}$

we might stipulate that $v((all,f)) = 1$ iff $v((f,p)) = 1$ for every $p \in A^t$, hoping that $v((f,p))$ is already defined for all the relevant p. But it is not, because (all,f) itself is in A^t so we need, among other things, to know what $v((f,(all,f))$ is in order to calculate $v((all,f))$, and this itself may depend on $v((all,f))$ (for instance, when f is negation).

An alternative is to restrict the higher-order quantifiers to *simple* propositions, properties and relations. This makes it easier to generate well-founded definitions of truth, given assignments of extensions to all the simple propositions, properties and relations. But we do not validate H because we universal instantiation can have false instances when predicate is satisfied by all simple entities, but has complex counterinstances.

Another option, *predicativism* (following Russell's no vicious circles principle) is to restrict the quantifiers to complex propositions, properties and so on that do not involve the quantifiers themselves. It is also possible to endow the above applicative structure with Leibnizian valuations that satisfy a weakened clause for the quantifiers which ranges only tuples that do not contain any quantifiers, although the matter is somewhat more complicated.

In Section 13.2 we considered, among other things, the idea of taking a higher-order logic $\mathbf{L}$, and considering the theory you get by adding to $\mathbf{L}$ every sentence $a \neq_\sigma b$ that is not a theorem of $\mathbf{L}$ where a and b are closed in some signature Σ. If the result of doing this is consistent we call it $=$-coherent (see Section 8.3).

Definition 15.7 (Leibnizian homomorphism)**.** *Let* $\mathbf{M} = (\mathbf{A}, [\![\cdot]\!]_M, v)$ *and* $\mathbf{N} = (\mathbf{B}, [\![\cdot]\!]_N, v)$ *be models. A homomorphism of models* $h : \mathbf{M} \to \mathbf{N}$ *is a homomorphism of the underlying* λ*-interpretations* $(\mathbf{A}, [\![\cdot]\!]_M)$ *and* $(\mathbf{B}, [\![\cdot]\!]_N)$*. Say that a homomorphism of models in* Leibnizian *iff it preserves Leibniz equivalence: if* $a, b \in A^\sigma$ *are Leibniz equivalent in* $\mathbf{M}$ *then* $h(a), h(b) \in B^\sigma$ *are Leibniz equivalent in* $\mathbf{N}$.

Exercise 15.7. *Say that a class of models* $\mathcal{C}$*, is directed iff whenever* $\mathbf{M}_1, \mathbf{M}_2 \in \mathcal{C}$ *there exists a 'lowerbound'* $\mathbf{N} \in \mathcal{C}$ *such that there exist Leibnizian homomorphisms* $h_1 : \mathbf{N} \to \mathbf{M}_1$ *and* $h_2 : \mathbf{N} \to \mathbf{M}_2$*. Prove that if the models of* $\mathbf{L}$ *are directed, then* $\mathbf{L}$ *is coherent.*[ii]

15.5 Philosophical issues surrounding model theory

We will now take a break from proving things to discuss some philosophical issues concerning the use of model theory.

First, we should acknowledge that our models are set-theoretic constructions. For instance, in Henkin structures, the domain of properties $A^{e\to t}$ consists of functions from A^e to A^t. We have adopted a fairly general model theory in which the members of the set $A^{e\to t}$ don't need to be set-theoretic objects like functions. Even when they aren't, they are still individuals and fall within the range of the first-order quantifiers.

A sentence of the form $\exists_{e\to t} G$ is true in a model just in case there is some individual f belonging to the domain $A^{e\to t}$ such that the denotation of G applied to f is true. Quantification into predicate position is thus cashed out in terms of first-order quantification over members of $A^{e\to t}$.

Taken uncritically, this observation invites the Quinean objection that higher-order quantification is just first-order quantification (perhaps over sets) in disguise. This is not quite true: the *truth-in-*$\mathbf{M}$ of a higher-order quantificational claim *in a given model* $\mathbf{M}$ is tantamount to a first-order quantificational claim. But we have not shown that the truth *simpliciter* of a higher-order quantificational claim is tantamount to a first-order quantificational claim.

Indeed, our main contention in Section 0.5 was that higher-order generalizations are intelligible and are *sui generis* claims that should not be understood in terms of first-order quantification. The observation that our models analyse higher-order claims using first-order quantificational claims does, however, raise a couple of important questions. If higher-order generalizations are indeed *sui generis*, as opposed to disguised first-order generalisations, then our set-theoretic models do not articulate the intended meanings of higher-order claims. We might then ask:

1. What are we up to when we study model theory, if not articulating the intended meanings of expressions of a given higher-order language?
2. Can there be a discipline (if not model theory) whose aim is to articulate the intended meanings of expressions of a higher-order language?

Let's start with the first question. Even if our model theory does not provide us with intended meanings of higher-order sentences (and other expressions), it serves many other useful purposes. Primary among these is that it provides a means for proving consistency results. We have seen that many attractive theses in higher-order metaphysics turn out, on closer inspection, to be inconsistent. Before one can defend the truth of a thesis of higher-order metaphysics one often first want some guarantee that there isn't a straightforwardly logical refutation of it. A consistency proof provides exactly this sort of guarantee: it ensures that one cannot reason, using logical rules alone, from the thesis in question to an inconsistency. The soundness theorem guarantees that if a sentence (or collection of sentences) has a model, then it is consistent, for a model of a sentence makes true all the derivable consequences of that sentence, and yet doesn't make everything true, establishing that not everything is a derivable consequence of that sentence. A completeness theorem is also handy because it ensures that if a sentence is consistent, you will be able to find a model that makes it true. It ensures that one is not embarking on a potentially fruitless search when one looks for a model of a sentence which is (in fact) consistent.

Models are also practically useful when it comes to exploring the derivable consequences of a theory. Typically this process goes something like the following: you suspect that a claim is a consequence of your theory and try to derive it from the theory. If you are unable to find a derivation, you might start searching for a countermodel instead. Even if your original suspicion is correct, and there is no countermodel, often the process of searching for (and failing to find) a countermodel gives you insight into how a correct derivation of the claim should go.

Models can also be used to characterise inferences and state new theories. For any class of models, $\mathcal{C}$, the set of sentences that are true in all members of $\mathcal{C}$ is a higher-order theory. Often metaphysical intuitions about the structure of properties and relations can inform suitable choices of model. For instance, if one endorsed Booleanism (Table 6.4), one might look at models in which A^{ι} forms a Boolean algebra under the operations $[\![\wedge]\!]$, $[\![\vee]\!]$ and $[\![\neg]\!]$. If one wanted to investigate more structural conceptions of propositions one might restrict attention to models that behave abstractly like languages do (we'll investigate this further in Section 17.2). Or one might have some other model granularity. Kit Fine, for instance, has a model of propositions in which they are understood in terms of sets of truth-makers and sets of falsity-makers. Perhaps there is a natural class of models in which A^{ι} consists of pairs of sets of truth-makers and falsity-makers, in which case the resulting higher-order theory would be worthy of investigation and should be expected to reveal the metaphysical implications of the truth-maker framework.

So model theory is an extremely useful tool, even if it doesn't provide the intended meanings of expressions of a higher-order language. But is there still a place for the second project of giving intended meanings for expressions of a higher-order language? I think the first thing to note is that if that project is possible then it must be carried out in a metalanguage that is itself higher-order. A sentence like $\exists_{e\to t}G$ can then be cashed out in the metalanguage, not by using a first-order quantification over members of some set $A^{e\to t}$, but using a quantifier that binds into predicate position. If one accepts the intelligibility of higher-order quantification there should be nothing objectionable about theorising in a metalanguage like this. Moreover, it should come as no surprise that the intended meanings of the higher-order quantifiers should be spelled out using higher-order quantifiers. After all, the standard clause for first-order quantifiers in first-order logic is cashed out using first-order quantification:

$\exists xA$ is satisfied by assignment g iff **for some** a in the domain, A is satisfied by $g[a/x]$.

Similarly, the standard clause for conjunction is spelled out using conjunction in the metalanguage:

$A \wedge B$ is satisfied by g iff A is satisfied by g **and** B is satisfied by g.

And so on.

It would be hard (not to mention perverse) to spell out the semantic clause for $\exists xA$ without using first-order quantificational idioms in the metalanguage. It would be very strange to expect the clause for higher-order quantifiers to be any different.

I have emphasized here that the motivations for employing a higher-order metalanguage for a higher-order language are mostly just common sense, and are of a piece with the idea that you shouldn't use propositional logic as a metalanguage for first-order logic. But in fact there are more specific logical problems when attempting to give a first-order treatment of higher-order quantification. Quantification into predicate position allows one to form existential generalizations like $\exists_{e\to t}X(X =_{e\to t} \lambda x.x \notin x)$ and $\exists_{e\to t}X(X =_{e\to t} \lambda x.\neg \mathit{Inst}(x,x))$, but if we were to attempt to treat these quantifiers using first-order quantification over sets or properties our quantifiers would omit one or the other: there is no set of sets not containing themselves, or property of properties not instantiating themselves. In fact these problems generalize to any theory according to which interpretations are individuals (see Williamson (2003), Section 1).

Remark 15.2. Notice that in the standard clause from first-order logic the clause for the existential quantifier is cashed out in terms of *restricted* quantification over a set (the first-order domain).

Assuming that the intended meaning of a first-order quantifier is unrestricted, this clause in the model theory has the effect that even the first-order quantifiers fail to express their intended unrestricted meanings relative to a model. (No set contains itself, for instance, and consequently no quantifier restricted to a set ranges over everything.) So first-order model theory, as standardly stated, is in fact subject to the same criticism that we have raised against higher-order model theory. The same sorts of justifications of model theory we have just provided can be wielded here: model theory is still useful for proving consistency results and characterizing inferences (see Kreisel (1967)). But, following Friedman (MS), and Rayo

and Williamson (2003), one can also provide intended clauses for the quantifiers by lifting the restriction on the interpretation of quantifiers:

$\exists xA$ is satisfied by assignment g iff for some a , A is satisfied by $g[a/x]$.

Both Friedman and Rayo and Williamson theorize in a higher-order metalanguage, and lift the usual restriction that interpretations of predicates by sets, instead representing the interpretations of predicates by quantifying into predicate position.

What might a suitable metalanguage look like? In Section 1.3 we quoted Russell on the view that it is strictly speaking incorrect to talk about meaning as though it were a single relation: according to Russell there are really a collection of primitives. The reference relation is the correct semantic primitive for reasoning about names, but a different semantic primitive is needed to talk about the meanings of sentences ('means that' of type $e \to t \to t$), yet another semantic primitive is needed to talk about the meanings of predicates ('ascribes' of type $e \to (e \to t) \to t$), and so on.

Here is one option, for making this precise, following the formalism of Bacon (2018a).[9] Suppose that $\mathcal{L}(\Sigma)$ is a higher-order language. Now consider the following metalanguage, $\mathcal{L}(\Delta)$, where Δ contains the usual higher-order constants, and:

$\langle M\rangle \in \Delta^e$ for each term $M \in \mathcal{L}(\Sigma)\}$
$[\![\cdot]\!]^\sigma \in \Delta^{e\to\sigma}$

Here $\langle M\rangle$ is a first-order individual—a name for the expression M of $\mathcal{L}(\Sigma)$, for instance, a quotation name—subject to the constraint that $\langle M\rangle \neq \langle N\rangle$ whenever $M \neq N$.

$[\![\cdot]\!]^\sigma$ is a term of type $e \to \sigma$: for instance, if it is applied to a name $\langle F\rangle$ for a predicate $F \in \mathcal{L}^{e\to t}(\Sigma)$ then in yields a predicate of the metalanguage, which we write $[\![\langle F\rangle]\!]^{e\to t}$. When it is applied to something that is not a name for a predicate of the object language, we understand it as denoting some predetermined default property. We have suggestively chosen to notate in a way that parallels the denotation function of a $\mathcal{L}(\Sigma)$-interpretation. But note that the denotation function of a $\mathcal{L}(\Sigma)$-interpretation is a function, and thus a certain sort of set: it has type e not $e \to \sigma$.

Using these primitives we can begin to axiomatize the sort of higher-order metatheory we would need for theorizing about the semantics of a higher-order language, just as a semanticist using sets and functions might adopt the axiomatic framework of first-order ZFC as their metatheory. A basic set of principles telling us about the meanings of the logical operations are presented below.

C(app) $[\![\langle MN\rangle]\!]^\tau = [\![\langle M\rangle]\!]^{\sigma\to\tau}[\![\langle N\rangle]\!]^\sigma$
C($\forall$) $[\![\langle\forall\rangle]\!]^{(\sigma\to t)\to t} = \forall$
C($\wedge$) $[\![\langle\wedge\rangle]\!]^{t\to t\to t} = \wedge$
C($\neg$) $[\![\langle\neg\rangle]\!]^{t\to t} = \neg$
C(S) $[\![\langle S\rangle]\!]^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho} = S$
C(K) $[\![\langle K\rangle]\!]^{\sigma\to\tau\to\sigma} = K$
$[\![\beta\eta]\!]$ $[\![\langle M\rangle]\!] = [\![\langle N\rangle]\!]$ when M and N are $\beta\eta$ equivalent.

For further details about what one can do in this model theory, and its limitations, the reader should consult the appendix of Bacon (2018a). Note that unlike our discussion of Russell, we have treated our semantic primitive for theorizing about sentences (say) not as a relation of type $e \to t \to t$, between sentences and propositions, but as an operation of type $e \to t$ mapping a sentence to its propositional meaning. These approaches are equivalent given Functional Plenitude (see Section 6.5), but in the absence of that principle it may be safer to take the relation as the primitive and translate the above principles into this framework: some remarks on these variations may also be found in Appendix A.2 of Bacon (2018a).

15.6 Incompleteness and higher-order logic

Let me conclude our discussion of the completeness theorem with some brief remarks about incompleteness in higher-order logic. Conventional wisdom has it that 'higher-order logic' is incomplete, and that the 'theorems of higher-order logic' are not recursively axiomatizable. When a logic is recursively axiomatizable it means that an algorithm could be concocted that could verify, of each sentence that is in the logic, that it is indeed in that logic.[10] I think there is a good sense in which both of these claims are true, but we need to be careful in untangling them. As we have emphasized already (Section 5.1) there isn't really any such thing as 'higher-order logic' in the same way that there is 'first-order logic' or 'propositional logic': there are many interesting pairwise incompatible higher-order logics containing logical principles with substantive philosophical and mathematical implications. Some higher-order logics are recursively axiomatizable. For instance, our weakest logic H, and the logic C of Classicism are presented axiomatically, and therefore are recursively axiomatizable. Other logics are not: the logic of full general models—the set of formulas true in all full general models under every assignment—is not recursively axiomatizable, nor are the formulas valid in full Fregean models or full possible worlds models.

In contemporary logic, the question of completeness has another degree of freedom: even once you have fixed on a particular higher-order logic, it is often said to be sound or complete only with respect to a model theory.

Definition 15.8 (Soundness and Completeness of a Logic)**.** *Let* **L** *be a higher-order logic, and let* $\mathcal{C}$ *be a class of general models.* **L** *is:*

sound *with respect to* $\mathcal{C}$ *if and only if every formula in* **L** *is true in every model in* $\mathcal{C}$ *under every assignment.*

complete *with respect to* $\mathcal{C}$ *if and only if every formula true in every model in* $\mathcal{C}$ *under every assignment belongs to* **L**.

Our completeness theorem tells us that every higher-order logic **L**, whether recursively axiomatizable or not, is sound and complete with respect to the class of general models of that logic. It is sound with respect to this class by definition, and it is complete with respect to the class because for any formula A not in **L**, $\mathbf{L} \cup \{\neg A\}$ determines a consistent theory, and has a general model by our completeness theorem.

When contemporary logicians say that higher-order logic is incomplete they typically mean this: the logic of *Fregean models* (Example 15.1) is not recursively axiomatizable. Or in other words, no recursively axiomatizable logic is sound and complete with respect to the class of Fregean models. But the significance of this fact is very unclear: why should we

want a logic that is complete with respect to Fregean models? The logic of Fregean models contains many statements, such as the claim that there are only two propositions, that are widely rejected in contemporary metaphysics and one should not want one's system to prove. One can generalize Fregean models in various ways that avoid this particular consequence, but typically one will find other contentious consequences. The logic of full possible worlds models (Example 15.2) similarly cannot be recursively axiomatized, but this logic is again highly contentious. Similar results hold for other classes of full general models that contain some models with infinite type e domains. But fullness is a fairly non-obvious constraint: it means the model has an $e \to t$ property for every function from individuals to propositions. For instance, there is a function that maps you to the proposition that the moon is made of cheese: one might have thought that applying a property to an individual should yield a proposition that is about or involves that individual somehow.

In short, the significance of these model-theoretic kinds of incompleteness is highly obscure. However, the identification of completeness with completeness with respect to Fregean models (or indeed, any class of models) is a relatively modern identification. Logicians were raising the issue of completeness of logics long before the development of set-theoretic model theory for first and higher-order logic. A key difference between the thinking of these early logicians and the modern perspective on logic is that they were treating the languages of propositional, first-order and higher-order logic as *interpreted* languages whose sentences could already be evaluated for truth of falsity. For these early logicians, the set of sentences we are trying to axiomatize are not the sentences valid in some special class of set-theoretic models, whose relationship to reality may potentially be in doubt. The target set of sentences had a more direct relationship to reality: unlike the model-theoretic notion, they make reference to what is in fact *true*. For instance, when Hilbert and Ackermann (1928) talk of a logical truth of propositional, first-order or higher-order logic they mean a sentence A that has only true substitution instances, and this is essentially the notion used by Gödel (1929) in his original proof of the completeness of first-order logic.[11] As they note (p. 129), in the case of higher-order logic one can simply universally quantify out all the non-logical variables using the higher-order quantifiers, thus the logical truth of $\forall_e y(Fy \vee \neg Fy)$ amounts to the truth of $\forall_{e\to t} X \forall_e y(Xy \vee \neg Xy)$. (See also Bernays and Schönfinkel (1928) p. 347; this is also essentially the approach of Tarski (1936b).[12] Thus we offer the following definition:

Definition 15.9 (Logical Truth)**.** *A sentence $A(c_1 \dots c_n)$ of an interpreted higher-order language $\mathcal{L}(\Sigma)$, whose non-logical constants are $c_1 : \sigma_1, \dots, c_n : \sigma_n$, is a* logical truth *if and only if $\forall_{\sigma_1} x_1 \dots \forall_{\sigma_n} x_n . A(x_1 \dots x_n)$ is a truth.*[13]

Here we take the notion of truth simpliciter as given, although precise definitions of it may be possible for certain fragments of higher-order logic (including dyadic second-order logic, which is all that is needed here).[14] Note that what is a logical truth in this sense depends on what reality is like: if there are in fact only two propositions this will be a logical truth (because it is a purely logical statement), if the higher-order statement of the continuum hypothesis is true then it is a logical truth, and so on.

Remark 15.3. Hilbert and Ackermann do not take Extensionalism to be baked into higher-order logic—at this time it was often taken to be a convenient assumption, to be taken as true only when quantifiers binding into predicate position are restricted in a certain way (Whitehead and Russell, (1910), Introduction III.2, Gandy (1956))—so their sense of completeness

is very much more pertinent to present concerns than, say, the contemporary logician who defines it in terms of completeness with respect to Fregean models.

Now we may ask a question with obvious significance: is it possible to give a recursive axiomatization of the logical truths? And the answer here is *no*, given plausible assumptions. The reason relates to a well-known result due to Gödel: that it is not possible to give a recursive axiomatization of the truths of arithmetic.[15] In a higher-order language, every arithmetical truth may be stated in terms of two primitives: $<: e \to e \to t$, and $0 : e$, representing the numerical ordering of numbers, and the number 0 respectively. So every arithmetical truth may be written in the form $A(<, 0)$ where $<$ and 0 are the only non-logical constants appearing in this sentence. Moreover, one can define in purely logical vocabulary a relation $\mathrm{Nat} : (e \to e \to t) \to e \to t$ that characterizes the properties of $<$ and 0 up to isomorphism, so that $\mathrm{Nat} < 0$ and also if $\mathrm{Nat}\, Xy$ then X is order isomorphic to $<$ and has y as a minimal element. Assuming that the standard arithmetical axioms are true, it follows that every truth of arithmetic $A(<, 0)$ is equivalent to the truth of a pure sentence of higher-order (indeed second-order) logic: $\forall_{e\to e\to t} X \forall_e y(\mathrm{Nat}\, Xy \to A(X, y))$.[16] If we had a way of recursively axiomatizing the logical truths—i.e. the truths in purely logical signature of higher-order logic—we would have a way of deciding whether sentences of the form $\forall_{e\to e\to t} X \forall_e y(\mathrm{Nat}\, Xy \to A(X, y))$ are true or not, and thus a way of recursively axiomatizing the arithmetical truths (which we know we cannot do). This way of arguing for the impossibility of giving a recursive axiomatization of the logical truths is entirely neutral about metaphysical claims, such as Extensionalism. By contrast, the model-theoretic notion of completeness requires one to choose a particular class of models, and thus bake in metaphysically contentious assumptions.

Remark 15.4. In Classicism, there is another class of sentences one might reasonably try to axiomatize, namely the sentences $A(c_1 \dots c_n)$ such that $\Box \forall_{\sigma_1} x_1 \dots \forall_{\sigma_n} x_n . A(x_1 \dots x_n)$ is true.[17] Provided the laws of arithmetic are contingent it is consistent that these are recursively axiomatizable (these sentences could coincide with the theorems of Classicism—see Corollary 18.1) but Goodsell (2022) shows, given Rigid Comprehension, the truths of arithmetic are not contingent so that given this assumption these truths are plausibly not recursively axiomatizable either.

Given all of this, what is the purpose of our above completeness theorem and the general notion of model? As we have already pointed out, model theory is a powerful tool for establishing consistency. Models also have plenty of heuristic value: models can sometimes represent the structure of reality well, giving us a 'picture' to work with. Models can sometimes suggest further primitives we might add in a metaphysical theorizing (for example, we will later develop a heuristically useful model theory in Section 17.2 for representing the notion of metaphysical definability introduced in Section 13.1).

Endnotes

1. An equation is satisfied by a model when $[\![M]\!]^g = [\![N]\!]^g$ for every assignment g.
2. The models described here are based on the model theory for the full λ-language presented in Benzmüller et al. (2004).
3. But see the discussion in Section 4.1.
4. Fine (1977) develops a theory in which one can begin to make sense of the idea a set of worlds involves an individual (although there are complications). See also Bacon (2020).
5. See for instance Fine (1977), Stalnaker (2012), Fritz and Goodman (2016), Fritz (2018a), and Fritz (2018b).

6. In fact, Henkin introduced his technique to prove the completeness of Extensionalism relative to Henkin Fregean models.
7. Some steps of the resulting proof may require some intermediate steps to added so that it satisfies the official criteria for being a proof, but this can be done in a systematic way.
8. This can always be done. For instance, one could start with any old enumeration of the $\sigma \rightarrow t$ predicates and constants and define a new enumeration of the predicates as follows: F_{n+1} is the smallest numbered predicate (in the original enumeration) involving only $a_1, \ldots, a_n$ that hasn't already been selected (i.e. isn't $F_1, \ldots, F_n$).
9. Other approaches to higher-order metatheory can be found in Rayo and Williamson (2003) and Rayo and Uzquiano (1999).
10. See, for instance, Boolos et al. (1974).
11. Both authors also require it to be true no matter what the first-order quantifiers range over (a condition that is not imposed by the other papers cited above). Hilbert and Ackermann write: 'A formula of the predicate calculus is called logically true . . . only if, independently of the choice of the domain of individuals, the formula always becomes a true sentence for any substitution of definite sentences, of names of individuals belonging to the domain of individuals, and of predicates defined over the domain of individuals, for the sentential variables, the free individual variables and the predicate variables respectively.' (p. 68 of Hilbert and Ackermann (1928)). Note that they do not require one to consider restrictions of the higher-order quantifiers when determining logical truth (p. 129).
12. The main difference between Tarski and these prior authors is that Tarski formulates his definition of logical truth in terms of a precisely defined notion of truth and satisfaction lacking from the preceding accounts. For Tarski, the logical truth of $\forall_e y(Fy \vee \neg Fy)$ amounts to the claim $\forall_{e \rightarrow t} X(\mathrm{Sat}(\ulcorner \forall_e y(Xy \vee \neg Xy) \urcorner, X)$, as opposed to $(\mathrm{True}(\ulcorner \forall_{e \rightarrow t} X \forall_e y(Xy \vee \neg Xy) \urcorner)$. These claims are equivalent for Tarski, however.
13. Analogous definitions can be offered for a general higher-order language $\mathcal{J}(\Sigma)$, but if there are constants appearing in A in places that cannot be occupied by a bound variable, the definition of logical truth may lie in a richer higher-order language.
14. Tarski (1936a) defines truth of an extensional nth-order logic in $n + 1$th order logic.
15. Gödel (1962).
16. It is possible to prove $\mathrm{Nat}(<, 0) \rightarrow (A(<, 0) \leftrightarrow \forall_{e \rightarrow e \rightarrow t} X \forall_e y(\mathrm{Nat}\, Xy \rightarrow A(X, y)))$ in H. If we assume additionally that every theorem of H is true, and we assume 'Peano arithmetic' (i.e. $\mathrm{Nat}(<, 0)$) is true and we assume that truth is closed under modus ponens, it follows that the truth of an arithmetical sentence is equivalent to a sentence of pure higher-order logic.
17. This notion is defined in Bacon (2018c), and there called 'metaphysical validity'.

Hints for exercises

[i] **Hint:** Consult the substitution lemma from Chapter 14.

[ii] **Hint:** Show that any finite set of distinctness claims $a_1 \neq_{\sigma_1} b_1, \ldots, a_n \neq_{\sigma_n} b_n$ individually consistent in **L** have a model, and use this to show that one cannot prove a contradiction from an set of **L**-consistent distinctness claims.

16

Logical relations

In this chapter, we introduce a powerful tool for comparing applicative structures and constructing new applicative structures from old: *logical relations*. In Section 16.1 we introduce the key concept and prove the fundamental theorem of logical relations in Section 16.2. Sections 16.3–16.6 consider various applications of logical relations and can be mostly read independently of one another. The final Section 16.7, concerns Kripke logical relations and λ-definability; Kripke logical relations generalize the notion of a logical relation in a particularly natural way and are helpful in modelling notions like metaphysical definability.

16.1 Logical relations

Logical relations are a particularly versatile tool for manipulating applicative structures. The following motivating examples should give you a sense of their wide applicability; we'll consider each of these examples in more detail later in the chapter.

1. Intuitively an arbitrary Henkin structure is 'contained' in the corresponding full Henkin structure with the same base type domains. However, we will see that this containment cannot be spelled out as literal set-theoretic containment. How can we make this idea precise?
2. Certain applicative structures are functional even though the domains at functional types are not sets of functions (they are not Henkin). For instance, our model of structured propositions in Example 15.3, treated structured entities as tuples and application as ordered pair formation, rather than function application. A natural conjecture is that this structure (and indeed, any functional structure) is isomorphic to a Henkin structure in which the domains of type $\sigma \to \tau$ consists of functions from the domains of type σ to the domains of type τ, and application is interpreted simply as function application. The corresponding Henkin structure and isomorphism can be constructed via logical relations.
3. Given a non-functional applicative structure, is there a way to quotient the structure by identifying functionally equivalent entities in function types? Simply quotienting by functional equivalent—the relation that holds between $f, g \in A^{\sigma\to\tau}$ when $\mathrm{App}(f, a) = \mathrm{App}(g, a)$ for all $a \in A^{\sigma}$—does not produce an applicative structure, so we need a more nuanced approach.
4. What does it mean for an entity of arbitrary type to be 'extensional'? (In the same way that, e.g., an operator is extensional if it is truth-functional.)

DOI: 10.4324/9781003039181-21

5. Relating to 1 and 2, the term structure of some countable language quotiented by $\beta\eta$-equivalence, $\mathcal{L}(\Sigma)/\sim_{\beta\eta}$, is intuitively 'contained' in a full Henkin structure with infinite base type domains. This allows us to prove Friedman's result that if two terms are not $\beta\eta$-equivalent one can find an interpretation of them in a full Henkin structure that witnesses their distinctness.

The focus of this chapter will be model-theoretic applications of logical relations. However, there are many applications of logical relations that we will not cover here. Possibly the most important of which is that they can be used to prove the strong normalization and confluence theorems for the notion of β-reducibility and $\beta\eta$-reducibility; this has far-reaching consequences ranging from computer science to proof theory.

The key concept that will unify the solutions to these problems is that of a logical relation. Suppose that $A = (A^{\cdot}, \mathrm{App}_A^{\cdot})$ and $B = (B^{\cdot}, App_B^{\cdot})$ are applicative structures.

Definition 16.1. *A binary logical relation R between A and B is a type indexed collection of relations $R^\sigma \subseteq A^\sigma \times B^\sigma$ such that:*

$$R^{\sigma\to\tau} fg \text{ iff, for every } a \in A^\sigma \text{ and } b \in B^\sigma, \text{ if } R^\sigma ab \text{ then } R^\tau\ \mathrm{App}(f,a)\ \mathrm{App}(g,b)$$

More generally, an n-ary logical relation between structures $A_1, \ldots, A_n$ is a type-indexed set of n-ary relation $R^\sigma \subseteq A_1^\sigma \times \ldots \times A_n^\sigma$ such that

$$R^{\sigma\to\tau} f_1 \ldots f_n \text{ iff, for every } a_i \in A_i^\sigma, \text{ if } R^\sigma a_1 \ldots a_n \text{ then } R^\tau\ \mathrm{App}(f_1,a_1)\ldots\mathrm{App}(f_n,a_n)$$

Most of our examples will involve binary logical relation between two applicative structures, or between a single applicative structure and itself. Note that the clauses for being a logical relation effectively provide us with an inductive definition of R^σ in terms of the relations R^e and R^t at the base types. Thus any pair of relations $R^e \subseteq A^e \times A^e$ and $R^t \subseteq A^t \times A^t$ extend to a unique logical relation.

Convention 16.1. *When R is logical relation between A and B we will write $R : A \to B$.*

Note that the definition of a logical relation is symmetric about A and B, meaning that, unlike the similar notation indicating the domain and codomain of a function, whenever $R : A \to B$, the converse of R, $R^c : B \to A$ (defined by the converse of R^σ at each type σ) is also a logical relation which captures essentially the same connection between A and B in the opposite order. Often the choice of the direction of the arrow is merely an indication of emphasis.

Comprehension Check 16.1. *Convince yourself that the converse of a logical relation is a logical relation.*

Here are some simple examples of logical relations

Example 16.1 (Universal relation)**.** *Given any two applicative structure, the logical relation R^σ that relates all elements of A^σ to all elements of B^σ (i.e. $R^\sigma = A^\sigma \times B^\sigma$) is a logical relation.*

Example 16.2 (Identity)**.** *Suppose that A is a functional applicative structure. Then the identity relation I^σ between A^σ and A^σ, relating each element of A^σ to itself and nothing else, is a logical relation.*

Exercise 16.1. *a. Show that the universal relation is a logical relation between any two applicative structures.*

b. Establish that identity is a logical relation between any functional applicative structure and itself.

c. Explain why identity can fail to be a logical relation when A is not functional. (If you don't want to lose momentum, we discuss this case in more detail in Section 16.5.)

Another basic example of a logical relation are permutations

Example 16.3 (Permutations)**.** *Suppose A is a full Henkin structure, and suppose that* $\pi^e : A^e \to A^e$ *and* $\pi^t : A^t \to A^t$ *are permutations on the base domains. These may be extended to a permutation of arbitrary domains* A^σ *by conjugation. The definition is extended inductively as follows:* $\pi^{\sigma\to\tau} f := \pi^\tau \circ f \circ \pi^{\sigma\,-1}$*, where* $\pi^{\sigma\,-1}$ *is the inverse of* π^σ*.*

If we consider a permutation as a certain sort of relation, $R^\sigma_\pi ab$ *iff* $\pi^\sigma a = b$*, it is a logical relation.*

Exercise 16.2.

a. Show by induction on types that π^σ *is a permutation for each* σ*.*

b. Show that R_π*, as defined above, is a logical relation.*

These examples also have applications in metaphysics. For instance, Kit Fine and Robert Stalnaker have invoked them to elucidate notions relating to whether an entity is qualitative—is not about specific individuals, like the proposition that there are philosophers—or haecceitistic—about particular people, like the proposition that Ruth Barcan is a philosopher.[1]

It's worth noting that some logical relations are special cases of concepts we have already encountered, such as congruences and partial homomorphisms.

Exercise 16.3. *Suppose R is a logical relation between A and itself, and suppose that* R^e *and* R^t *are equivalence relations. Show that R is a partial congruence.*

Exercise 16.4. *Suppose* $R : A \to B$ *is a logical relation between A and B, and moreover, R is functional: if* $R^\sigma ab$ *and* $R^\sigma ab'$ *then* $b = b'$ *for any* σ*,* $a \in A^\sigma$ *and* $b, b' \in B^\sigma$*.* R^σ *is thus a partial function from* A^σ *to* B^σ *for each type* σ*. Show that R is a partial homomorphism.*

Notice that not every congruence or partial congruence is a logical relation. For $f, g \in A^{\sigma\to\tau}$ with $f \sim_{\sigma\to\tau} g$, a congruence only required to satisfy the condition that if $a \sim_\sigma b$ then $\mathrm{App}(f, a) \sim_\sigma \mathrm{App}(g, b)$, whereas a logical relation is required to satisfy the converse implication as well. Congruences, unlike the logical relations considered in Exercise 16.3, are not fully determined by their actions on base types. A useful analogy to invoke here is that congruences stand to these logical relations as Henkin structures stand to full Henkin structures. The same point applies to the Exercise 16.4 as well: functional logical relations are partial homomorphisms, but not conversely.

In the rest of this chapter we will adopt the following convention:

Convention 16.2. *Given an applicative structure A,* $f \in A^{\sigma\to\tau}$ *and* $a \in A^\sigma$ *we will write fa as short for* $\mathrm{App}^{\sigma\tau}(f, a)$*.*

As usual, we will associate brackets to the left, writing $fa_1a_2 \ldots a_n$ *for* $(\ldots((fa_1)a_2)\ldots a_n)$*.*

Although this notation allows us to employ the familiar notation for function application, it's important to remember that in a general applicative structure *f* needn't be a function, and *fa* will refer to the applicative structures notion of application. We should only employ this convention when the relevant notion of application is clear from context.

I'll end this section by mentioning a useful lemma. It is very easy to prove, but is worth highlighting as it makes manipulating logical relations a lot easier. Let $f \in A^{\sigma_1 \to \ldots \to \sigma_n \to \tau}$ and $g \in B^{\sigma_1 \to \ldots \to \sigma_n \to \tau}$ be elements of two applicative structure whose type is of a curried n-ary function $\sigma_1 \to \ldots \to \sigma_n \to \tau$. If we want to show that f Rs g it suffices to show that for any $a_1 \in A^{\sigma_1}, \ldots, a_n \in A^{\sigma_n}$ and any $b_1 \in B^{\sigma_1}, \ldots, b_n \in B^{\sigma_n}$ that if $R^{\sigma_1} a_1 b_1$ and ... and $R^{\sigma_n} a_n b_n$, that $R^{\tau}(fa_1 \ldots a_n)(gb_1 \ldots b_n)$.

Proposition 16.1. *$R^{\sigma_1 \to \ldots \to \sigma_n \to \tau} fg$ if and only if for any $a_1 \in A^{\sigma_1}, \ldots, a_n \in A^{\sigma_n}$ and any $b_1 \in B^{\sigma_1}, \ldots, b_n \in B^{\sigma_n}$, if $R^{\sigma_1} a_1 b_1$ and ... and $R^{\sigma_n} a_n b_n$, then $R^{\tau}(fa_1 \ldots a_n)(gb_1 \ldots b_n)$.*

16.2 The fundamental theorem of logical relations

We may extend the notion of a logical relation from applicative structures to $\mathcal{J}(\Sigma)$-interpretations by stipulating that the interpretations of the constants must be preserved by the relation.

Definition 16.2 (Logical relation between $\mathcal{J}(\Sigma)$-interpretations)**.** *Suppose that $(A, [\![\cdot]\!]_A)$ and $(B, [\![\cdot]\!]_B)$ are $\mathcal{J}(\Sigma)$-interpretations for a general λ-language $\mathcal{J}(\Sigma)$. A logical relation between the applicative structures A and B, $R : A \to B$, is a logical relation between these interpretations iff additionally*

For every constant $c \in \Sigma^{\sigma}$, $R^{\sigma} [\![c]\!]_A [\![c]\!]_B$.

In this case, we say that the interpretations of the constants are preserved by R.

We will sometimes also use the notion of invariance. Given a logical relation R between an applicative structure A and itself, we say that $a \in A^{\sigma}$ is *invariant* under R iff a Rs itself: $R^{\sigma} aa$.

Here is an example of a logical relation on a possible worlds model of the logical language containing the constants $\to$ and $\forall_{\sigma}$ for each σ.

Example 16.4. *Consider the signature $\Lambda = \{\forall_{\sigma}, \to\}$ of higher-order logic. Consider two full possible worlds models A and B that have the same type e domain (see Example 15.2: this means $A^{t} = P(X)$ and $B^{t} = P(Y)$ for some sets X and Y, and the truth-functional connectives interpreted with the evident set-theoretic operations, and universal quantifiers by arbitrary intersection). Suppose that $R^{t} : A^{t} \to B^{t}$ is a surjective function that preserves the set-theoretic operations:*

- *$R^{t}(\bigcap_i p_i) = \bigcap_i R^{t}(p_i)$ where p_i are a collection of subsets of X indexed by $i \in I$.*
- *$R^{t}(\bigcup_i p_i) = \bigcup_i R^{t}(p_i)$*
- *$R^{t}(X \setminus p) = Y \setminus R^{t}(p)$*

Since $A^{e} = B^{e}$ we will let R^{e} simply be identity. The resulting pair R^{e} and R^{t} defines a logical relation that is functional and surjective at each type. Moreover, it can be shown that it is a logical relation of $\mathcal{J}(\Sigma)$-interpretations as well. This means that:

- *$R^{(\sigma \to t) \to t}([\![\forall_{\sigma}]\!]_A) = [\![\forall_{\sigma}]\!]_B$*
- *$R^{t \to t \to t}([\![\to]\!]_A) = [\![\to]\!]_B$*

Exercise 16.5. *Show the final two bullet points in the preceding example.*

It is important that our homomorphism be a surjective function at the base types if it is to be functional at higher types; we'll explore the behaviour of functional and surjective logical relations shortly.

This brings us to the most basic result concerning logical relations: logical relations preserve the interpretations of arbitrary closed terms. Here we follow the convention of treating variables as part of every general λ-language by fiat.

Theorem 16.1 (The fundamental theorem of logical relations)**.** *Suppose that $(A, [\![\cdot]\!]_A)$ and $(B, [\![\cdot]\!]_B)$ are $\mathcal{J}(\Sigma)$-interpretations of a general λ-language $\mathcal{J}(\Sigma)$, and R is a logical relation between these interpretations. Then for any pair of closed terms $M : \sigma$:*

$$R^\sigma [\![M]\!]_A [\![M]\!]_B.$$

Indeed, if g and h are assignments, write Rgh iff $R^\sigma g(x)h(x)$ for each type σ and variable x of type σ. The fundamental theorem is proven by showing the more general result that if Rgh then $R^\sigma [\![M]\!]^g_A [\![M]\!]^h_B$ for arbitrary open or closed terms M.

Proof. We prove this by induction on term structure. Suppose Rgh. For constants, it's guaranteed that $R^\sigma [\![c]\!]^g_A [\![c]\!]^h_B$ by the definition of a logical relation between $\mathcal{J}(\Sigma)$-interpretations. For variables, it follows that $R^\sigma [\![x]\!]^g_A [\![x]\!]^h_B$ because $[\![x]\!]^g_A = g(x)$, $[\![x]\!]^h_B = h(x)$ and we have assumed that Rgh.

Suppose for induction that $M : \sigma \to \tau$ and $N : \sigma$ and that whenever Rgh, $R^{\sigma\to\tau} [\![M]\!]^g_A [\![M]\!]^h_B$ and $R^\sigma [\![N]\!]^g_A [\![N]\!]^h_B$. Thus, assuming Rgh, we know that $R^\tau \operatorname{App}([\![M]\!]^g_A, [\![N]\!]^g_B) \operatorname{App}([\![M]\!]^h_B, [\![N]\!]^h_B)$ because R is a logical relation. Finally this means that $R^\tau [\![MN]\!]^g_A [\![MN]\!]^h_B$ as this is how the interpretation function is defined.

Now suppose that the inductive hypothesis holds for $M : \tau$. We want to show it holds for $\lambda x.M : \sigma \to \tau$. Suppose Rgh. To show $R^{\sigma\to\tau} [\![\lambda x.M]\!]^g_A [\![\lambda x.M]\!]^h_B$ it suffices to show that for any $a \in A^\sigma$ and $b \in B^\sigma$, if $R^\sigma ab$ then $R^\tau \operatorname{App}([\![\lambda x.M]\!]^g_A, a) \operatorname{App}([\![\lambda x.M]\!]^h_B, a)$.

Here our reasoning will depend on whether we are dealing with functional or non-functional interpretations. In either case, we can establish that $\operatorname{App}([\![\lambda x.M]\!]^g_A, a) = [\![M]\!]^{g[a/x]}_A$ and $\operatorname{App}([\![\lambda x.M]\!]^h_B, b) = [\![M]\!]^{h[b/x]}_B$. This suffices to complete the argument: since $R^\sigma ab$, we have $Rg[a/x]h[b/x]$, so by our inductive hypothesis $R^\tau [\![M]\!]^{g[a/x]}_A [\![M]\!]^{h[b/x]}_B$, so $R^\tau \operatorname{App}([\![\lambda x.M]\!]^g_A, a) \operatorname{App}([\![\lambda x.M]\!]^h_B, a)$ as required.

The identities $\operatorname{App}([\![\lambda x.M]\!]^g_A, a) = [\![M]\!]^{g[a/x]}_A$ and $\operatorname{App}([\![\lambda x.M]\!]^h_B, b) = [\![M]\!]^{h[b/x]}_B$ are just built into the definition of a functional interpretation. With non-functional $\mathcal{J}(\Sigma)$-interpretations we need a slightly more involved chain of identities.

$$\operatorname{App}([\![\lambda x.M]\!]^g_A, a) = \operatorname{App}([\![\lambda x.M]\!]^{g[a/x]}_A, [\![x]\!]^{g[a/x]}_A) = [\![(\lambda x.M)x]\!]^{g[a/x]}_A = [\![M]\!]^{g[a/x]}_A$$

The first identity holds because (i) $[\![x]\!]^{g[a/x]}_A = a$ and (ii) g and $g[a/x]$ agree on the free variables in $\lambda x.M$, and it is built into the definition of $[\![\cdot]\!]$ that the interpretation of a term relative to any two assignments agreeing on the free variables in that term is the same. The second identity holds due to the definition of the interpretation of applicative terms. The third identity holds because it is built into the definition of $[\![\cdot]\!]$ that it is invariant under $\beta\eta$-equivalence. □

One consequence of the fundamental theorem of logical relations is that the interpretations of combinators are always preserved by logical relations, since combinators are closed terms in any signature. Thus we have, for instance:

$R^{\sigma\to\sigma}[\![\lambda x.x]\!]_A[\![\lambda x.x]\!]_B$
$R^{\sigma\to\tau\to\sigma}[\![\lambda xy.x]\!]_A[\![\lambda xy.x]\!]_B$

Although these claims follow from the fundamental theorem, it's instructive to work through a few examples like this to get a feeling for how the theorem works in specific instances. Consider the second example. Let us write k_A for $[\![\lambda xy.x]\!]_A$ and k_B for $[\![\lambda xy.x]\!]_B$. It is easily shown that $k_Aaa' = a$ for any $a \in A^\sigma$ and $a' \in A^\tau$, and that $k_Bbb' = b$ for any $b \in B^\sigma$ and $b' \in B^\tau$. Now by Proposition 16.1 to show the required $R^{\sigma\to\tau\to\sigma}k_Ak_B$, it suffices to show that for any $a \in A^\sigma, b \in B^\sigma$, $a' \in A^\tau$ and $b' \in B^\tau$, if $R^\sigma ab$ and $R^\tau a'b'$ then $R^\sigma(k_Aaa')(k_Bbb')$. But of course, given that $R^\sigma ab$, and the fact that $k_Aaa' = a$ and $k_Bbb' = b$ we get the result that $R^\sigma(k_Aaa')(k_Bbb')$ immediately.

In the following exercise, you will establish an analogue of the fundamental theorem of logical relations for interpretations of combinatory languages.

Exercise 16.6. *Suppose that $(A, [\![\cdot]\!]_A)$ and $(B, [\![\cdot]\!]_B)$ are interpretations the combinatory language $CL(\{S, K\}, \Sigma)$. Thus A has combinators, $s_A^{\sigma\tau\rho}$ and $k_A^{\sigma\tau}$ for each σ, τ and ρ, in the relevant domains, and similarly for B. Suppose R is a logical relation between these interpretations.*

a. Show that $R^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}s_A^{\sigma\tau\rho}s_B^{\sigma\tau\rho}$ and $R^{\sigma\to\tau\sigma}k_A^{\sigma\tau}k_B^{\sigma\tau}$.

b. Show that for every term $P : \sigma$ of $CL(\{S, K\}, \Sigma)$, $R^\sigma[\![P]\!]_A[\![P]\!]_B$.

Let's end this section with an application of the fundamental theorem to an issue in the philosophy of logic: what is it that distinguishes the logical notions, like conjunction, the universal quantifier, the identity relation, and so forth, from the non-logical ones? Many distinctions in logic, such as the notion of logical truth and consequence, depend on which notions are treated as logical, and which are treated as non-logical. So, at least if one thinks words like 'logical truth' and 'logical consequence' single out unique notions, some account of the logical notions is needed. In Corcoran and Tarski (1986), Tarski offered an account of what it was for a notion to be *logical*. His idea was that logical notions shouldn't be sensitive to particular individuals, and as such a logical notion ought to be invariant under arbitrary permutations of the individuals. In the following question, you will establish that the constants of higher-order logic denote operations that are invariant under such permutations in a Fregean model of higher-order logic, and in the second part you will use the fundamental theorem to show that anything definable from the logical constants using λs is logical. This means the combinators count as logical on Tarski's conception, as well as many interesting cardinality quantifiers that may be defined in higher-order logic, such as the quantifier 'there are inaccessibly many *F*s'.

Exercise 16.7. *Consider the Fregean model of Example 15.1 $(A, [\![\cdot]\!], v)$. That is, a full Henkin structure where A^e is some set, $A^t = \{0, 1\}$, and the truth-functional connectives and quantifiers are given their natural interpretations. Now suppose π^e is a permutation of A^e, π^t is the identity mapping on A^t, and that π^σ is extended up the hierarchy by conjugation, as defined in Example 16.3.*

a. Show that $\pi^{(\sigma\to t)\to t}[\![\forall_\sigma]\!]_A = [\![\forall_\sigma]\!]_A$ and $\pi^{t\to t\to t}[\![\to]\!]_A = [\![\to]\!]_A$.

b. By applying the fundamental theorem of logical relations, show that $\pi^\sigma[\![M]\!]_A = [\![M]\!]_A$ *for every closed term* M *of the language of pure higher-order logic* $\mathcal{L}(\Lambda)$.

It is worth noting that the logical notions by Tarski's account outstrip the notions definable by closed terms of pure higher-order logic. For instance, for any infinite cardinality κ, the quantifier 'there are κ many Fs' that maps an element $f \in A^{e\to t}$ to 1 if f maps at least κ elements to 1, and maps f to 0 otherwise, counts as logical in Tarski's sense. If A^e is sufficiently big, then these cardinality quantifiers will be distinct for distinct κ and could easily be uncountable in number. Yet there are only countably many cardinality quantifiers definable in the language of pure higher-order logic, because there are only countably many terms in that language.

16.3 Logical partial functions

In this section, we'll examine a particular class of logical relations that behave, at each type, like a partial function. These are called *logical partial functions*:

Definition 16.3 (Logical partial function)**.** *A logical relation* $R : A \to B$ *is a logical partial function iff*

$R^\sigma ab$ *and* $R^\sigma ab'$ *imply* $b = b'$.

When this condition is satisfied, we may employ the more familiar way of notating partial functions. If $R^\sigma ab$ for some b, then that b is unique, and we may write $R^\sigma(a)$ to denote it. If a is not related to anything by R^σ then $R^\sigma(a)$ is undefined. The following exercise demonstrates that a logical partial function is a partial homomorphism (recall Exercise 16.4.)

Notice that while every logical partial function is a partial homomorphism, the converse is not true. In particular, the conditions for a logical relation guarantee that if g is an element of $B^{\sigma\to\tau}$ such that $g(h^\sigma(a)) = h^\tau(fa)$ whenever $h^\sigma(a)$ is defined, thet $h^{\sigma\to\tau}f = g$. By contrast, we have no guarantee that $h^{\sigma\to\tau}f$ is even defined when h is an arbitrary partial homomorphism.

In what follows we will explore an extended example of a logical partial function which illustrates one important sort of application of logical relations.

Consider two Henkin structures, A and B that agree on the base types, so that $A^e = B^e$ and $A^t = B^t$, but they differ on the functional types. In particular, suppose that B is the full Henkin structure with these base types—$B^{\sigma\to\tau} = B^\sigma \to B^\tau$ for every σ and τ—and that A is not full. (Here we use $A \to B$ to denote the set of all functions with domain A and codomain B.) Suppose, in fact, that $A^{\sigma\to\tau}$ includes some but not all of the functions from A^σ to A^τ for each σ and τ: $A^{\sigma\to\tau} \subset A^\sigma \to A^\tau$, where this inclusion is proper for each σ and τ.

There is clearly an intuitive sense in which the applicative structure A is 'contained' in B. The problem is how to make this idea precise. A naïve attempt to spell out this relation would be to say that $A^\sigma \subset B^\sigma$ for each type σ. Unfortunately, this would be wrong: this relationship holds at type $\sigma \to \tau$ when σ and τ are base types, and obviously holds at the base types. But when σ is not a base type $A^{\sigma\to\tau}$ and $B^{\sigma\to\tau}$ are disjoint. The reader should try the following exercise before reading the explanation below:

Exercise 16.8.

a. Show that $A^\tau \subseteq B^\tau$ whenever $\tau = \sigma_1 \to \ldots \to \sigma_n$ and each σ_i is a base type.

b. Given the fact that $A^{t\to t}$ is a proper subset of $B^{t\to t}$ show that $A^{(t\to t)\to t}$ and $B^{(t\to t)\to t}$ are disjoint.

A function $f \in A^{(t\to t)\to t}$ is a function with domain $A^{t\to t}$, and a function in $B^{(t\to t)\to t}$ is a function with domain $B^{t\to t}$. Since the domains of the former sorts of functions are a proper subset of the domains of the latter sorts of functions they will never be identical. For instance, every function f of the latter sort will be defined on any argument $g \in B^{t\to t} \setminus A^{t\to t}$, but no function of the former sort will be defined on g. (Even if one identifies functions with sets of ordered pairs—an identification we previously cautioned against—the resulting sets will be distinct, since one will contain ordered pairs the other doesn't.)

Nonetheless, there is an intimate relation between the elements of A^σ and B^σ. For some but not all $b \in B^\sigma$, we can find an element $T^\sigma(b) \in A^\sigma$ that is b's correlate in A^σ. Note that the operation T is only partially defined—B is in some sense 'bigger' than A so not every element of B will correspond to an element of A.

Remark 16.1. One might wonder why we didn't define the correlation the other way around, from A to B instead, to obtain a total function. We shall see shortly that this would result in an overdefined 'function' rather than a partially defined one: a single element of A^σ will correspond to multiple elements of B^σ when σ is sufficiently complex (as in our running example of $(t \to t) \to t$).

When $\sigma = t$ or $t \to t$ we have inclusions $A^\sigma \subseteq B^\sigma$, so that bs correlate, $T^\sigma(b)$ is easily defined to be itself if b belongs to A^σ and undefined otherwise. What is the relation at type $(t \to t) \to t$? Well a given function $f \in B^{(t\to t)\to t}$ induces an operation taking elements of $A^{t\to t}$ to A^t, since f is a function which is defined on all the elements of $A^{t\to t}$ (and further elements of $B^{t\to t}$ as well). Suppose moreover that this induced function belongs to $A^{(t\to t)\to t}$: then we will call this fs correlate in $A^{(t\to t)\to t}$.

The way we have defined the correlate of a function in of type $(t \to t) \to t$ is part of a much more general pattern. Suppose we have found a correlate $T^\sigma(b)$ in A of each $b \in B^\sigma$, and similarly a correlate $T^\tau(b)$ for each $b \in B^\tau$, and we are looking to find a correlate of a function $f \in B^{\sigma\to\tau}$ in $A^{\sigma\to\tau}$. Suppose, moreover, that there is a $g \in A^{\sigma\to\tau}$ with the following property:

(*) For any $b \in B^\sigma$ such that $T^\sigma(b)$ is defined, $g(T^\sigma(b)) = T^\tau(fb)$.

then g is the correlate of f. So we may let $T^{\sigma\to\tau}(f)$ be defined and be equal to g. Another way to say that g is the correlate of f (i.e. $g = T^{\sigma\to\tau}(f)$), is just this: whenever f maps b to fb, g maps the correlate of b, $T^\sigma(b)$ to the correlate of fb, $T^\tau(fb)$.

Given the stipulation that $T^e(b) = b$ and $T^t(b) = b$ when b is in B^e and B^t respectively, this completes an inductive definition of T.

Now notice that our definition of T conforms exactly to the definition of a logical relation. Writing $T^\sigma ab$ to mean $T^\sigma(a)$ is defined and is equal to b, the definition in (*) is equivalent to:

(**) $T^{\sigma\to\tau} fg$ if and only if, for any $b \in B^\sigma, a \in A^\sigma$, if $T^\sigma ba$ then $T^\tau(fb)(ga)$.

For instance, given (**) we can show that if $T^{\sigma\to\tau}(f)$ is defined and $= g$, and $T^{\sigma}(b)$ is defined and $= a$, then $g(T^{\sigma}(b)) = T^{\tau}(fb)$. This is because (**) tells us, given these assumptions, that that $T^{\tau}(fb)$ is defined and $= ga$, and of course ga is just $g(T^{\sigma}(b))$.

Of course, we also need to check that T really is a well-defined partial function at each type. We need to show that there can't be two distinct g satisfying the above conditions: that T is a *logical partial function*. To establish this we will need another property of T that can be shown simultaneously with its well-definedness, namely that T is surjective.

Proposition 16.2. *For each σ, T^{σ} is a surjective partial function.*

Proof. $T^{e} : B^{e} \to A^{e}$ and $T^{t} : B^{t} \to A^{t}$ are defined by the identity function so automatically are surjective partial functions.

Suppose that T^{σ} and T^{τ} are surjective partial functions. We'll begin by showing that $T^{\sigma\to\tau}$ is functional. Suppose that $T^{\sigma\to\tau}fg$ and $T^{\sigma\to\tau}fg'$. So whenever $T^{\sigma}ba$, $T^{\tau}(fb)(ga)$ and $T^{\tau}(fb)(g'a)$. Since T^{σ} and T^{τ} are functional we may rewrite this using the functional notation: $g(T^{\sigma}b) = T^{\tau}(fb)$ and $g'(T^{\sigma}b) = T^{\tau}(fb)$ whenever $T^{\sigma}b$. Thus, whenever $T^{\sigma}b$ is defined $g(T^{\sigma}b) = g'(T^{\sigma}b)$. Now every element of A^{σ} is of the form $T^{\sigma}b$ for some b, since T^{σ} is surjective. So we know that g and g' are equal when applied to any argument whatsoever. Since A is functional, we have that $g = g'$ as required.

Now we show that $T^{\sigma\to\tau}$ is surjective. Suppose $g \in A^{\sigma\to\tau}$. Writing $T^{\sigma-1}(a)$ for $\{b \in B^{\sigma} \mid T^{\sigma}(b) = a\}$. When T^{σ} is functional, $T^{\sigma-1}(a)$ is disjoint from $T^{\sigma-1}(a')$ when $a \neq a'$. Choose any function f in $B^{\sigma\to\tau}$ that maps elements of $T^{\sigma-1}(a)$ to elements $T^{\tau-1}(ga)$ for each a. We know such a function exists because B is full, and by construction $T^{\sigma\to\tau}(f) = g$. □

If we inspect the above proof, the only facts we used were that A was functional, B was full. So we get a much more general theorem stating that if you define a logical relation from B to A using surjective partial functions at the base types, then you will get logical relation that is a surjective partial function at all types.

Theorem 16.2. *Suppose that A is a functional applicative structure, B is a full applicative structure, and $T : B \to A$ is a logical relation where T^{e} and T^{t} are surjective partial functions. Then T^{σ} is a surjective partial function for every σ.*

Let's finish this example by analysing how $T : B \to A$ encodes the idea that A is properly contained within B. The idea that the containment is proper at sufficiently complex types amounts simply to the fact not every element of B is related, by T, to an element of A (T is partial when considered as a function). For instance, since A is not full at type $t \to t$, and B is, not every $f \in B^{t\to t}$ will have a correlate in $A^{t\to t}$. The containment of A in B is represented by the fact that T is surjective: every element of A has at least one T-correlate in B.

What is more interesting is that an element of A can in fact have several correlates. The reason is simply this: when A^{σ} correlates with a proper subset X of B^{σ}, there will be lots of operations in $B^{\sigma\to\tau}$ that have a correlate in $A^{\sigma\to\tau}$ if there are any. In particular, if f and g are members of $B^{\sigma\to\tau}$ that agree about what they do to the elements of their domain that correspond to things in A^{σ}, i.e. the set $X \subseteq B^{\sigma}$, then f and g will have the same correlate in A. From As perspective f and g are the very same operation: they induce the same applicative behaviour with respect to As elements. So the situation in general is this:

- T associates with each B^{σ} a subset, $T^{\sigma}(A^{\sigma})$ of elements that correspond to elements of A^{σ}.
- Each such subset of B^{σ} is partitioned by the equivalence relation of being the same operation from As perspective: $b \sim_{\sigma} b'$ iff $T^{\sigma}(b) = T^{\sigma}(b')$.

$\sim_\sigma$ is clearly a partial equivalence relation at each type. We will call the relation $\sim_\sigma$ defined above the *kernel* of T^σ, with an obvious nod to the corresponding notion we introduced for homomorphisms. Interestingly, the kernel of this particular logical relation is itself a logical relation between B and itself called a logical partial equivalence relation. It satisfies the weaker property of being a partial congruence, we may apply the isomorphism theorem (Exercise 14.25) establishing that A is isomorphic to $B/\sim$, which spells out the true sense in which A is contained within B.

Logical relations are also useful for comparing applicative structures operating with different notions of application. Recall the possible world's applicative structure used to model intensionalism. There first-order properties were modelled as functions from individuals to propositions (sets of possible worlds). By contrast, in Example 14.4 we considered a model of structured propositions in a first-order property is modelled by a tuple of simple constituents, and the application of a property to an argument was represented by taking the ordered pair of the tuples representing the property and the argument. One might have thought that the identification of properties with functions—an identification pervasive in possible worlds semantics—is inherently linked with coarse-grained intensional metaphysics, and that more fine-grained metaphysics, such the structured view, would have to treat properties differently. But actually the opposite is true: using logical relations we'll show that the applicative structure from Example 14.4 is actually isomorphic to a Henkin structure, in which functional types are modelled by sets of functions. (More generally, we will show that any functional applicative structure is isomorphic to a Henkin structure.)

Let's begin by recapping the applicative structure from Example 14.4. We modelled structured propositions, properties, etc. by ordered tuples of simple constituents. For each type σ, we let S^σ stand for a set of simple entities of type σ, and we let A be the smallest typed collection of sets such that A^σ contains S^σ for each σ, and such that given a structured entity of functional type $t \in A^{\sigma\to\tau}$ and a structured entity of argument type $a \in A^\sigma$, the ordered pair (t, a) is also in A^τ. Let us additionally assume that A^σ is non-empty for each type σ (as would be guaranteed, for instance, if every S^σ was non-empty).

In this applicative structure the entities belonging to $A^{\sigma\to\tau}$ are not functions but ordered tuples, and application is therefore not interpreted as function application but as ordered pair formation with its argument. Nonetheless, the applicative structure is functional: if $\mathrm{App}(t, a) = \mathrm{App}(s, a)$ for every $a \in A^\sigma$ then $t = s$, because if the ordered pairs (t, a) and (s, a) are equal for *any* a at all, then t and s are equal (and by stipulation, A^σ is non-empty). A given tuple $t \in A^{\sigma\to\tau}$ may not be a function from A^σ to A^τ but it determines one, namely the function $\hat{t}$ that maps s to the ordered pair (t, s):

$$\hat{t} := s \mapsto (t, s) : A^\sigma \to A^\tau$$

In the terminology of Chapter 14, $\hat{t}$ is the applicative behaviour of t. For a concrete example, if $p \in S^t$, and $and \in S^{t\to t\to t}$, the tuple (and, p) is an element of $A^{t\to t}$, intuitively representing the operation of conjunction with p, and the function it determines is $q \mapsto ((and, p), q)$.

Here, then, is the conjecture: our applicative structure of structured tuples A is actually just isomorphic to a Henkin structure, B, in which $B^{\sigma\to\tau}$ is a set of function for each σ and τ, and which agrees with A on the base type domains. The naïve strategy for constructing such an isomorphism would be to simply map each tuple $t \in A^{\sigma\to\tau}$ to the function $\hat{t}$ defined above. However, this strategy will not work: $\hat{t}$ is a function from tuples in A^σ to tuples in A^τ. By contrast, in our envisioned Henkin structure, B^σ and B^τ will be sets of functions, not tuples, when σ and τ are themselves functional types, so the elements of $B^{\sigma\to\tau}$ should be

functions that take functions as inputs and outputs, not tuples. However, if we can construct our Henkin structure B and isomorphism, h, between B and A, together simultaneously, we could use the isomorphism between A^σ and B^σ and between A^τ and B^τ to define the relevant domain and isomorphism at type $\sigma \to \tau$. For if a given tuple t induces a function $\hat{t} : A^\sigma \to A^\tau$ as described above, and we have already constructed isomorphisms $h^\sigma : A^\sigma \to B^\sigma$ and $h^\tau : A^\tau \to B^\tau$, t also induces a function from $B^\sigma \to B^\tau$ through these isomorphisms. It is, of course, the same relation we discussed in the context of the logical relation T above:

> $h^{\sigma\to\tau}t = f$ iff for every $s \in A^\sigma$, $b \in B^\sigma$, if $h^\sigma s = b$ then $h^\tau(App_A(t,s)) = App_B(f,b)$.
>
> More informally: function f corresponds to the tuple t (i.e. $h^{\sigma\to\tau}t = f$) iff f takes the correspondent of a tuple s (i.e. $h^\sigma(s)$) to the correspondent of the ordered pair (t,s) (i.e. $h^\tau(t,s)$).

But because h^σ and h^τ are isomorphisms, and h^σ has an inverse, we can actually simplify this definition to:

$$h^{\sigma\to\tau}t := b \mapsto h^\tau(\hat{t}(h^{\sigma-1}(b)))$$

Thus we can simply define $h^{\sigma\to\tau}t := h^\tau \circ \hat{t} \circ h^{\sigma-1}$.

- $B^e := A^e$, $B^t := A^t$
- h^e and h^t are the identity functions.
- $B^{\sigma\to\tau} := \{h^{\sigma\to\tau}t \mid t \in A^{\sigma\to\tau}\}$
- $h^{\sigma\to\tau}t := h^\tau \circ \hat{t} \circ h^{\sigma-1}$

To convince yourself of this try the following more general exercise

Exercise 16.9. *Let A and B be arbitrary applicative structures with B functional. Suppose that for each σ, h^σ is a bijection from A^σ to B^σ.*

a. Show that h is an isomorphism of applicative structures if and only if it is a logical relation.

b. Show that $h^{\sigma\to\tau}f$ is $b \mapsto h^\tau(\mathrm{App}(f, h^{\sigma-1}b))$.

There is nothing particularly special about the functional structure A used in this example. The construction of B and the isomorphism h depended only on the fact that A was functional. Thus we can use the theory of logical relations to give a simpler proof of Theorem 14.4:

Theorem 16.3. *Every functional applicative structure is isomorphic to a Henkin structure.*

16.4 Applications to equational theories

An equational theory in a language $\mathcal{L}(\Sigma)$ is a set of equations $M = N$ between terms of $\mathcal{L}(\Sigma)$ where the type of M and N is the same. When most people talk about an 'equational theory' they will have in mind a set of equations with some basic equational logic built into it—for instance that it contains the self-identities $M = M$, that it contains the equation $M = N$ whenever it contains $N = M$, and so forth—but for the purposes of this section we don't need to get particularly precise. Two sorts of equational theories are of special interest.

The first is the equational theory of pure $\beta\eta$-equivalence in a λ-language, consisting of the equations between terms of the same type that are $\beta\eta$-equivalent:

$$\beta\eta := \{P = Q \mid P, Q : \sigma, P \sim_{\beta\eta} Q\}$$

The second is the equational theory of a $\mathcal{L}(\Sigma)$-interpretation $M = (A, [\![\cdot]\!])$, and is defined as follows:

$$Th^{=}(M) := \{P = Q \mid P, Q : \sigma, [\![P]\!]^g = [\![Q]\!]^g \text{ for every assignment } g \text{ on } M\}$$

Given a class of $\mathcal{L}(\Sigma)$-interpretations $\mathcal{C}$ we say that an equational theory E is sound and complete with respect to $\mathcal{C}$ if and only if $E = \bigcap_{M \in \mathcal{C}} Th^{=}(M)$. In other words, E contains an equation if and only if that equation holds in every interpretation in $\mathcal{C}$. E has *least model completeness* with respect to $\mathcal{C}$ if and only if it is the equational theory of a single interpretation, i.e. $E = Th^{=}(M)$ for some interpretation $M \in \mathcal{C}$. When you have least model completeness, the least model not only assigns the same interpretations to terms equated by the equational theory (terms P and Q such that $P = Q \in E$), but also assigns distinct interpretations to the terms not equated by the equational theory.

Remark 16.2. Notice that least model completeness is not treated in standard courses on propositional and first-order logic because it usually doesn't hold in that context: a classical model makes any given sentence or its negation true, but most propositional and first-order theories do not contain any sentence or its negation. It is only a live possibility in the case of equational theories because we are working in a language without negation.

Now the first observation we can make is that the theory $\beta\eta$ has least model completeness with respect to the class of $\mathcal{L}(\Sigma)$-interpretations. The reason is simply this: in Definition 14.17 we constructed a $\mathcal{L}(\Sigma)$-interpretation called the open term interpretation of $\mathcal{L}(\Sigma)$. The elements of A^σ are equivalence classes $[M]_{\beta\eta}$ of terms under $\beta\eta$-equivalence. Moreover, it can be shown that $[\![M]\!]^g = [M]_{\beta\eta}$, whenever $g(x_i) = [x_i]_{\beta\eta}$ for each variable x_i.[2] So $M = N$ is in the equational theory of the open term interpretation only if $[M]_{\beta\eta} = [N]_{\beta\eta}$, and thus only if $M \sim_{\beta\eta} N$. Conversely, because the open term interpretation is a $\mathcal{L}(\Sigma)$-interpretation, we proved in Theorem 14.2 that all the equations in $\beta\eta$ are satisfied.

It's incredibly natural to think that you can find least models within more restricted classes of interpretations. The open term interpretation is functional, but isn't necessarily full. For instance, in a countable signature, there will only be countably many terms (and thus equivalence classes of terms) of type $t \to t$, whereas in a full interpretation where the base types are infinite, there are uncountably many because there are uncountably many functions between any pair of infinite sets. (The domains of the base types *are* infinite in the open term interpretation because there are infinitely many variables of any type.) Intuitively, one might expect that one could fill out the domains of functional type in the open term applicative structure by adding elements realizing every function not already realized without making any more equations true. Here's why: the open term interpretation is functional, and so is isomorphic to a Henkin structure. Moreover, we have seen that in a certain sense any non-full Henkin structure can be faithfully embedded in the full Henkin structure with the same base type domains, in a way that preserves the interpretation of terms, so that any pair of terms with different interpretations in the smaller structure will have different interpretations in the bigger one.

For reasons that should be familiar already, this informal mode of reasoning must be made precise by invoking logical relations. Let's begin with the following lemma.

Lemma 16.1. *Suppose M and N are $\mathcal{L}(\Sigma)$-interpretations. If $R : M \to N$ is a logical surjective partial function from M to N, then $Th^{=}(M) \subseteq Th^{=}(N)$*

Proof. Suppose that $[\![P]\!]^g_M = [\![Q]\!]^g_M$ for every assignment g over N. Let h be an arbitrary assignment on N. Since R is surjective, we can find an assignment g on M, such that $R^\sigma(g(x)h(x)$ for every variable x. By the fundamental theorem we know that $R^\sigma([\![P]\!]^g_M) = [\![P]\!]^h_N$ and $R^\sigma([\![Q]\!]^g_M) = [\![Q]\!]^h_N$. Since $[\![P]\!]^g_M = [\![Q]\!]^g_M$ it follows that $[\![P]\!]^h_N = [\![Q]\!]^h_N$ as required. □

Note that in this derivation we needed the assumption that R was surjective. However, if M is a functional interpretation, we can actually do away with this assumption.

Exercise 16.10. *In this exercise, you will prove the preceding assertion.*

a. Show that for any interpretations M, N and logical partial function $R : M \to N$ the equations between closed terms that hold in M hold in N (whether or not M is functional or R is surjective).

b. Show that if M is additionally functional then $Th^{=}(M) \subseteq Th^{=}(N)$.[i]

Now suppose that $T = (\mathcal{L}(\Sigma)/\sim_{\beta\eta}, [\cdot]_{\beta\eta})$ is the open term interpretation for a countable signature Σ, and A is any full Henkin structure in which A^e and A^t is infinite. Then we may construct a interpretation over A, $M = (A, [\![\cdot]\!]_M)$ as follows.

1. We may define a logical relation $R : A \to \mathcal{L}/\sim_{\beta\eta}$ as follows:
   1. R^e and R^t are any surjective functions from A^e to $\mathcal{L}^e(\Sigma)/\sim_{\beta\eta}$ and A^t to $\mathcal{L}^t(\Sigma)/\sim_{\beta\eta}$ respectively.
   2. Since A is full, $\mathcal{L}(\Sigma)/\sim_{\beta\eta}$ is functional, and R^e and R^t are surjective partial functions, R^σ is a surjective partial function for every type σ, by Theorem 16.2.
2. For each constant $c \in \Sigma^\sigma$, let $[\![c]\!]_M$ be any element $a \in A^\sigma$ such that $R^\sigma(a) = [c]_{\beta\eta}$ (i.e. $[\![c]\!]_T$). We know such an a exists because R is surjective.
3. We know that our logical relation of applicative structures $R : A \to \mathcal{L}^e(\Sigma)/\sim_{\beta\eta}$ is also a logical relation of $\mathcal{L}(\Sigma)$-interpretations, i.e. $R : M \to T$, since by construction R preserves the interpretation of the constants.
4. By Lemma 16.1 we know that $Th^{=}(M) \subseteq Th^{=}(T) = \beta\eta$.
5. Morever, since M is a $\mathcal{L}(\Sigma)$-interpretation $\beta\eta$ equivalent terms receive the same interpretation: so $\beta\eta \subseteq Th^{=}(M)$.

Putting this all together we get a proof of Harvey Friedman's famous result the theory of $\beta\eta$ conversion enjoys least model completeness with respect to full Henkin structures.

Theorem 16.4 (Friedman's theorem)**.** *Let A be any full Henkin structure in which A^e and A^t are infinite. Then there is a interpretation over A, $M = (A, [\![\cdot]\!]_M)$ such that $Th^{=}(M) = \beta\eta$.*

16.5 Logical partial equivalence relations

Just as functional logical relations correspond to very special sorts of partial homomorphisms, we'll see that logical relations between a structure and itself that are (partial) equivalence relations at each type are very special sorts of (partial) congruences.

Definition 16.4 (Logical partial equivalences)**.** *Suppose that $R : A \to A$ is a logical relation of applicative structures or $\mathcal{J}(\Sigma)$-interpretations. Then R is a logical partial equivalence if and only if R^σ is a partial equivalence relation for each type σ.*

The reader should recall that S is a partial equivalence relation on a set X iff it is a transitive symmetric relation. This is equivalent to it being an equivalence relation when restricted to $\{x \in X \mid Sxx\}$.

Convention 16.3. *We will notate partial equivalences with the symbol $\sim$ instead of R, and we will use infix notation instead of prefix notation, writing $a \sim_\sigma b$ instead of $R^\sigma ab$.*

Clearly a logical partial equivalence relation is a partial congruence: a partial congruence is a type indexed collection of partial equivalence relations that satisfy one direction of the condition for being a logical relation: $f \sim_{\sigma\to\tau} g$ only if, whenever $a \sim_\sigma b$, $fa \sim_\sigma gb$. What makes logical partial equivalences special is that they satisfy the other direction of this implication: this means that $\sim_{\sigma\to\tau}$ is completely determined by the behaviour of $\sim_\sigma$ and $\sim_\tau$.

To check that a logical relation is a logical partial equivalence it suffices to check that it is a partial equivalence relation on the base types.

Exercise 16.11. *Suppose that $R : A \to A$ is a logical relation between A and itself. Then R is logical partial equivalence if and only if R^e and R^t are partial equivalence relations.*

Thus any pair of partial equivalence relations on the base type domains of an applicative structure determines a unique logical partial equivalence on that structure.

An equivalence relation is a special case of a partial equivalence relation, in which every element of the set is in the field of the relation. Thus Exercise 16.11 tells us that any pair of equivalence relations on the base types will define a unique logical partial equivalence relation on the whole applicative structure. One might have thought, in analogy with Exercise 16.11, that the stronger property would also be inherited by the higher types: if $\sim_e$ and $\sim_t$ and are equivalence relations then $\sim_\sigma$ is for any σ (as opposed to merely being a partial equivalence). However, this is not so. The reader should attempt the following exercise before reading the explanation below.

Exercise 16.12. *Suppose A is a full Henkin structure and that $A^t = \{a, b, c\}$. Let $\sim_t$ partition A^t into two equivalence classes $\{a, b\}$ and $\{c\}$ (and let $\sim_e$ be the identity relation on A^e). Show that $\sim_{t\to t}$ is not an equivalence relation by constructing a function $f : A^t \to A^t$ such that $f \not\sim_{t\to t} f$.*

If you have attempted the exercise read on. Note that the condition $f \sim_{t\to t} f$ is tantamount to saying that f preserves $\sim_t$: if x and y are equivalent then so are $f(x)$ and $f(y)$. But clearly not every function preserves an equivalence relation (unless the relation is identity). So to complete this exercise you need to find a function that doesn't preserve $\sim_t$.[3]

Remark 16.3. The phenomenon we have just observed—that the property of being a partial equivalence relation is inherited up the type hierarchy, but the property of being an

equivalence relation is not—goes some way to explaining why concepts like being a partial equivalence relation, a partial congruence or a partial homomorphism are often more useful than their totally defined cousins.

The simplest example of an equivalence relation is identity, thus we have a unique logical partial equivalence relation on any applicative structure determined by identity at the base types.

Example 16.5. *For any applicative structure A, the logical relation determined by identity on the base types is a logical PER.*

Again, one might have suspected that this logical relation would be identity at every type. Identity is a very special sort of equivalence relation, because every function automatically preserves it. This means that, unlike in Exercise 16.12, $\sim_{t\to t}$ is an equivalence relation, not merely a partial equivalence. However, although $\sim_{t\to t}$ is an equivalence relation, it is not necessarily the identity relation. Spelling out the definition and appealing to the fact that $\sim_t$ is identity we have:

$$f \sim_{t\to t} g \text{ if and only if, for any } a, b \in A^t, \text{ if } a = b \text{ then } f(a) = g(b)$$

in other words, $f \sim_{t\to t} g$ iff f and g are cofunctional: $f(a) = g(a)$ for all $a \in A^t$. But cofunctional entities needn't be identical in a non-functional applicative structure. In which case $\sim_{t\to t}$ is merely an equivalence relation that isn't identity. And since a function can fail to preserve an equivalence relation (unless its identity), it follows that there could be elements in the higher type domain $A^{(t\to t)\to(t\to t)}$ that do not bear $\sim$ to themselves. For instance, if A was full then every applicative behaviour (i.e. function) from $A^{t\to t}$ to $A^{t\to t}$ would be realized by some element of $A^{(t\to t)\to(t\to t)}$, including the functions that do not preserve $\sim_{t\to t}$. (On the other hand, recall that in Example 16.2 and the subsequent exercise, we showed that when A is functional the identity relation at each type is a logical relation.)

Exercise 16.13. *Using the non-functional applicative structure from Example 14.3 (over some signature Σ of your choice), find an element (f, N) of $A^{(t\to t)\to t}$ such that $(f, N) \not\sim_{(t\to t)\to t} (f, N)$.*

Exercise 16.14. *Show that the type indexed relation $\approx$ that is identity on the base types, and is the relation of cofunctionality ($f \approx_{\sigma\to\tau} g$ iff for all $a \in A^\sigma$ $fa = ga$) is not a logical relation in any applicative structure that is full but not functional and on of whose base types contain at least two elements.*[ii]

These observations seem to suggest that for any given non-functional applicative structure, we can find functional structure 'inside' it, by quotienting somehow. The naïve strategy for doing this would be to simply identify cofunctional entities. But as we have seen in the previous exercise, this is not enough—the relation of cofunctionality is not even a congruence. We also need to throw away any functional entities that do not preserve cofunctionality between their argument and target types. What's more, we can identify distinct entities f and g in $A^{\sigma\to\tau}$ that are not cofunctional, so long as f and g respectively take pairs of elements of type A^σ that are identified to pairs of elements of A^τ that are identified: whenever a and b are identified fa and gb are identified according to our quotienting process. So what we are really dealing with is an inductively defined identification relation, $\sim_\sigma$ on each type σ, that identifies f and g iff f and g map identified elements to identified elements. And, of course, is just the condition for being a logical relation.

Definition 16.5 (Functional collapse)**.** *The functional collapse of an applicative structure A is the quotient of A by the logical relation $\sim$ generated by the identity relation at the base types.* $A/\sim$.

The following exercises establish that this structure really is functional. Indeed, the quotient of any applicative structure by a logical partial equivalence relation is a functional applicative structure.

Exercise 16.15. *Show that logical relation defined by partial equivalence relation on the base types is a partial equivalence relation at each type.*

The previous exercise implies that the logical relation generated by identity at the base types is a logical partial equivalence. The next exercise will imply that quotienting by it will yield a functional applicative structure.

Exercise 16.16. *Let A be any applicative structure and $\sim$ any logical partial equivalence on A. Show that $A/\sim$ is a functional applicative structure.*

Remark 16.4. The idea of a functional collapse was originally used by Gandy (1956) to show the relative consistency of the theory of types with the Functionality axiom given the consistency of the theory of types without it. His original application with a proof-theoretic one, and not a model-theoretic one. In the language of pure higher-order logic, one can define an object language term $\sim_\sigma$: $\sigma \to \sigma \to t$ that corresponds to a logical relation: $\sim_e$ is $=_e$, $\sim_t$ is $=_t$ and $\sim_{\sigma\to\tau}$ is $\lambda XY.\forall_\sigma xy(x \sim_\sigma y \to Xx \sim_\tau Yy)$. For each sentence A, one may define a translation A^* that is obtained by replacing the type σ quantifiers appearing in A with quantifiers restricted by the predicate $x \sim_\sigma x$. It may be shown that whenever HF (H with the Functionality axiom) proves a sentence A, H proves A^*. So any derivation of a contradiction in HF would yield a derivation of the translation of a contradiction (which is also a contradiction) in H.

Consider the notion of truth-functionality, from propositional logic. In that context, truth-functional connectives are usually identified with truth-functions. But without assuming the Fregean axiom, this identification is highly problematic: there are lots of operators that, like negation, map truths to falsehoods and falsehoods to truths—'John believes that' would behave like this if John just happened to believe all and only the falsehoods. The truth-functional behaviour of a connective does not pick out the connective uniquely.[4] And even assuming the Fregean axiom, you need the axiom of Functionality to ensure there is only one connective with any given truth-functional behaviour. The idea of 'having the same truth-functional behaviour' then is best thought of as a non-trivial equivalence relation between connectives. We can use the technology of logical relations to capture the notion of truth-functional behaviour more precisely: it is a partial logical equivalence relation obtained by treating materially equivalent propositions as equivalent. But what is interesting about this application is that it yields a notion of truth-functional behaviour that extends well beyond n-ary connectives—entities of type $t^n \to t$—to entities of higher types, such as $(t \to t) \to t$.

Example 16.6 (Truth-functional behaviour)**.** *Given a model of higher-order logic, $(A, [\![\cdot]\!], v)$, the relation of identity at type e and material equivalence $p \sim_t q$ iff $v(p) = v(q)$, generates a logical relation. We may say that an element $a \in A^\sigma$* has *a truth-functional behaviour iff $a \sim_\sigma a$, and that a and b have the same truth-functional behaviour iff $a \sim_\sigma b$.*

Remark 16.5. As in Remark 16.4, one can introduce the extensionality logical relation into the object language. By restricting all quantifiers by the formula $x \sim_\sigma x$ we obtain a translation of higher-order logic into itself under which Extensionalism is true. One can see this as making good on Russell's 'no class' theory of classes in which they are supposed to be ultimately explained in terms of quantification into predicate position in a non-extensional higher-order logic (see Russell (1919) Chapter 17; Whitehead and Russell, (1910), Introduction III.2).

Exercise 16.17. *Consider a full possible worlds structure from Example 15.2 generated from a set of worlds with at least two elements. Give an example of an element of this structure that does not have a truth-functional behaviour.*

Exercise 16.18. *Show that the kernel of a logical partial function is a logical equivalence relation. That is, suppose that $R : A \to B$ is a logical partial function from a full applicative structure A to a functional applicative structure B. The kernel of R on A^σ is defined by saying that $a \sim_\sigma b$ iff there exists a $c \in B^\sigma$ such that $R^\sigma ac$ and $R^\sigma bc$.*

16.6 λ-definability

The fundamental theorem of logical relations tells us that in any applicative structure, every closed pure λ-term of $\mathcal{L}(\emptyset)$ denotes an element that is invariant under every logical relation. This fact thus provides us with a sufficient condition for showing that an element of an applicative structure is not definable in the full λ-language: one simply has to construct a logical relation under which that element is not preserved. There is similar semantic condition that is sufficient for determining that one element of a structure cannot be defined from some others: the fundamental theorem tells us that if the elements in a set $X \subseteq \bigcup_\sigma A^\sigma$ are invariant under a given logical relation then so is $[\![M]\!]^g$ where M is any λ-term and g an assignment taking values in X. These notions of definability are especially useful in the semantic analysis of certain metaphysical notions, such as metaphysical definability introduced in Chapter 13.

Definition 16.6 (λ-definability)**.** *An element $d \in A^\sigma$ of a interpretation $(A, [\![\cdot]\!])$ is λ-definable iff there is a closed pure term $M \in \mathcal{L}^\sigma(\emptyset)$ such that $[\![M]\!] = d$.*

$d \in A^\tau$ is definable from *some elements $a_1 \in A^{\sigma_1}, \ldots, a_n \in A^{\sigma_n}$ iff $d = [\![M]\!]a_1 \ldots a_n$ for some pure closed term $M : \sigma_1 \to \ldots \to \sigma_n \to \tau$.*

Let's work through a concrete example.[5] Let $(A, [\![\cdot]\!])$ be a full Henkin structure for the pure λ-language $\mathcal{L}(\emptyset)$ and suppose that A^t contains at least two distinct elements p and q. There are lots of operations in this structure that have a broadly logical nature, which might be candidates for being definable in the full λ-language. For instance, consider an operation that decides the identity of any two elements of type t: if it is given two identical elements of A^t it outputs a designated element signifying that they're identical, and if it is given distinct elements it outputs a different designated element which we'll take to signify their distinctness. To give this operation a fighting chance of being λ-definable we will choose the two designated elements to be things we know to be denoted by closed λ-terms: $[\![\lambda xy.x]\!]$ and $[\![\lambda xy.y]\!] \in A^{t \to t \to t}$ (since A^t contains at least two distinct elements p and q we know that these two elements are different because they have different behaviours when applied to p and q respectively). So our equality function e has type $t \to t \to (t \to t \to t)$:

$e(a)(b) = [\![\lambda xy.x]\!]$ if $a = b$ and $e(a)(b) = [\![\lambda xy.y]\!]$ if $a \neq b$.

e is not λ-definable. We may show this by considering the logical relation generated by $R^t \subseteq A^t \times A^t$, where $R^t := \{(p,p),(p,q),(q,p)\}$ (because types involved in defining e only involve t, it will not matter what R^e is).

Firstly we can observe that it's not the case that our two designated elements are related: it is not the case that $R^{t\to t\to t}[\![\lambda xy.x]\!][\![\lambda xy.y]\!]$. For suppose otherwise: then using the defining condition of a logical relation we can infer from the fact that R^tqp, that $R^{t\to t}([\![\lambda xy.x]\!]q)([\![\lambda xy.y]\!]p)$. And from this and the fact that since R^tpq it would follow that $R^t([\![\lambda xy.x]\!]qp)([\![\lambda xy.y]\!]pq)$, which after simplifying the two arguments of R, is R^tqq. This contradicts the definition of R^t, which does not include (q,q).

Now we may see that it's not the case that $R^{t\to t\to(t\to t\to t)}ee$. For suppose otherwise. Since R^tpp we have $R^{t\to(t\to t\to t)}(e(p))(e(p))$, and since R^tpq we have $R^{t\to t\to t}(e(p)(p))(e(p)(q))$. Now $e(p)(p) = [\![\lambda xy.x]\!]$ since $p = p$ and $e(p)(q) = [\![\lambda xy.y]\!]$ since $p \neq q$, so the previous line contradicts the fact we established in the previous paragraph that it's not the case that $R^{t\to t\to t}[\![\lambda xy.x]\!][\![\lambda xy.y]\!]$. Now the fundamental theorem of logical relations may be applied: every closed λ-term denotes something that is invariant under every logical relation, and since the equality function is not invariant under every logical relation it is not λ-definable. Thus we have:

Example 16.7. *The equality function (defined by $e(a)(b) = [\![\lambda xy.x]\!]$ if $a = b$ and $e(a)(b) = [\![\lambda xy.y]\!]$ otherwise) is not λ-definable.*

Let's now turn to the question defining one element of an applicative structure from another. Suppose you're given the equality operation, one might wonder what expressive power this would give us. Could it, for instance, allow us to define a surrogate for quantification. Writing τ for the type $t \to t \to t$, this would be a function a which takes a function $f \in A^{t\to\tau}$ and spits out the designated truth (i.e. $[\![\lambda xy.x]\!]$) if f maps every element of A^t to the designated truth, and spits out the designated falsehood (i.e. $[\![\lambda xy.y]\!]$) if f doesn't map every element of A^t to the designated truth.

$a \in A^{(t\to\tau)\to\tau}$.
$a(f) = [\![\lambda xy.x]\!]$ if $f(d) = [\![\lambda xy.x]\!]$ for every $d \in A^t$, and $a(f) = [\![\lambda xy.y]\!]$ otherwise.

In the next exercise you will use the fundamental theorem of logical relations to show that a is not λ-definable in terms of equality.

Exercise 16.19. *Continuing with the structure A above, consider the logical relation generated by $R^t := \{(p,p),(q,q)\}$ (again the behaviour of R^e will not matter for this question). Let $\tau = t \to t \to t$.*

a. Show that $R^{t\to t\to\tau}ee$.

b. Find a pair of elements $f, g \in A^{t\to\tau}$ such that $R^{t\to\tau}fg$ but not $R^\tau(a(f))(a(g))$.

c. Use the fundamental theorem of logical relations to show that a is not definable from e.

Let's attempt one more application of logical relations to the question of definability. The simplest type in which there is more than one closed λ-term in the pure signature up to $\beta\eta$-equivalence is the type of the Church numerals, $(t \to t) \to t \to t$. In Chapter 3, we saw that for each natural number n, we may define in the full λ-language a term which takes an

operator of type $t \to t$ and a proposition of type t and applies the operator to the proposition n times:

$$\lambda X \lambda p.p$$
$$\lambda X \lambda p.(Xp)$$
$$\lambda X \lambda p.(X(Xp))$$
$$\vdots$$

Let us assume, as before, that $(A, [\![\cdot]\!])$ is a full Henkin structure and that, moreover, $A^t = \mathbb{N}$. So each of the λ-terms above denotes a distinct element of $A^{(t\to t)\to t\to t}$ because they each, for instance, map the successor function, *suc*, and 0 to different numbers. The Church numerals, in fact, are the only closed terms of $\mathcal{L}^{(t\to t)\to t\to t}(\emptyset)$ up to $\beta\eta$-equivalence.

In the final exercises of this section we will prove that for certain types, λ-definability coincides with being invariant under every logical relation between a Henkin structure and itself. We begin with the proposition below, which characterizes the functions in $A^{(t\to t)\to t\to t}$ that are invariant under every binary logical relation between a structure and itself as exactly those that are denoted by the Church numerals. These functions are what we will call *iterators*: they take a function $g \in A^{t\to t} = \mathbb{N}^{\mathbb{N}}$ and a natural number $a \in A^t = \mathbb{N}$ to the result of applying g to a some fixed number of times, which we henceforth write $g^n(a)$.[6]

Proposition 16.3. *Let $(A, [\![\cdot]\!])$ be a full Henkin structure such that $A^t = \mathbb{N}$. $f \in A^{(t\to t)\to t\to t}$ is invariant under every logical relation $R : A \to A$ iff f is an iterator: for some n, $f = g \mapsto a \mapsto g^n(a)$. That is to say, invariance under all logical relations on A and λ-definability coincide at type $(t \to t) \to t \to t$.*

Since an iterator function is denoted by a Church numeral, the right-to-left direction of this theorem follows from the fundamental theorem of logical relations, so we attend to the left-to-right direction.

Suppose that $f \in A^{(t\to t)\to t\to t}$, is invariant under every logical relation. To show that f is an iterator/denoted by a Church numeral, we must show that $f = g \mapsto a \mapsto g^n(a)$, for some $n \in \mathbb{N}$. The following exercise takes you through the steps of proving this.

Exercise 16.20. *Say that $f \in A^{(t\to t)\to t\to t}$ in an iterator iff for some n, $f(g)(a) = g^n(a)$ for every $g \in A^{t\to t}, a \in A^t$.*

a. Show that if f is an iterator then for every $g \in A^{t\to t}$ and $a \in A^t$, $f(g)(a) = g^{f(suc)(0)}(a)$.

b. Let $g \in A^{t\to t}$ and $a \in A^t$, and consider the logical relation generated by $R^t = \{(g^n(a), suc^n(0)) \mid n \in \mathbb{N}\}$. Show that $Ra0$ and $Rg(suc)$.

c. Show that if $f \in A^{(t\to t)\to t\to t}$ is invariant under R (i.e. $R^{(t\to t)\to t\to t}ff$) then $R(f(g)(a))(f(suc)(0))$.

d. Show that if f is invariant under every logical relation then, for an arbitrary $g \in A^{t\to t}$ and $a \in A^t$, $f(g)(a) = g^{f(suc)(0)}(a)$. That is, conclude that f is an iterator.

Up to $\beta\eta$-equivalence there is only one closed term of $\mathcal{L}(\emptyset)$ of type $t \to t$, namely $\lambda p.p$. Thus the identity function λ-definable element of $A^{t\to t}$ in any Henkin structure.

Exercise 16.21. *Suppose that $(A, [\![\cdot]\!])$ is a Henkin structure. Show that if $f \in A^{t\to t}$ is invariant under every unary logical relation, then f is the identity function.*

Observe that a completely analogous argument shows that if $f \in A^{t\to t}$ is invariant under ever binary logical relation on A it is identity identity.

Up to $\beta\eta$-equivalence there are only two closed term of $\mathcal{L}(\emptyset)$ of type $t \to t \to t$, namely $\lambda pq.p = K^{tt}$ and $\lambda pq.q = C^{ttt}K^{tt}$. Thus the functions $k = a \mapsto b \mapsto a$ and $k^c = a \mapsto b \mapsto b$ are the only λ-definable elements of $A^{t\to t\to t}$ in any Henkin structure.

Exercise 16.22. *Suppose that $(A, [\![\cdot]\!])$ is a Henkin structure, that $|A^t| \geq 2$, and that $0, 1 \in A^t$ are two distinct elements.*

a. Show that if $a, b \in A^t$ and if $f \in A^{t\to t\to t}$ is invariant under the relation generated by $R^t = \{(a, 0), (b, 1)\}$, then $f(a)(b) = a$ and $f(0)(1) = 0$ or $f(a)(b) = b$ and $f(0)(1) = 1$.

b. Show that if f is invariant under every logical relation, then $f = k$ or $f = k^c$ depending on whether $f(0)(1) = 0$ or $f(0)(1) = 1$.

16.7 Kripke logical relations

Proposition 16.3 and the subsequent exercises suggest a more general conjecture: that the λ-definable elements are exactly those that are invariant under every logical relation between a structure and itself. Unfortunately this conjecture is false, although the proof is beyond the scope of this book.[7]

In order to provide a semantic characterization of λ-definability, Gordon Plotkin (1973, 1980) suggested a generalization of logical relations.[8] Let's start by introducing a with a central notion from modal logic:

Definition 16.7 (Kripke frame)**.** *A Kripke frame is a pair $(W, \leq)$ where W is a set, called the* worlds*, and $\leq$ is a preorder on W, called the* accessibility relation*. That $\leq$ is a preorder means that it is:*

1. *Reflexive: $x \leq x$ for any $x \in W$.*
2. *Transitive: If $x \leq y$ and $y \leq z$ then $x \leq z$ for any $x, y, z \in W$.*

Observe that in some mathematical contexts the symbol '$\leq$' is reserved only for transitive reflexive relations that are also *anti-symmetric*: $x \leq y$ and $y \leq x$ implies $x = y$. We do not follow that convention here, and $\leq$ may well denote a relation that is not anti-symmetric.

Kripke frames $(W, \leq)$ are widely appealed to in the of study modal logics containing **S4**.[9] In Chapter 7 we observed that according to Classicism, the broadest necessity has a logic containing **S4**, and Kripke frames and the subsequent concepts in this section will play a role in modelling that theory. For now, we use them to characterize λ-definability.

Definition 16.8 (Kripke logical relation)**.** *Given applicative structures $A_1, \ldots, A_n$, and a Kripke frame $(W, \leq)$, a Kripke logical relation is a family of relations $R^\sigma_w \subseteq A^\sigma_1 \times \ldots \times A^\sigma_n$ for each type σ and world $w \in W$ such that:*

1. *If $R^\sigma_w a_1 \ldots a_n$ and $w \leq v$ then $R^\sigma_v a_1 \ldots a_n$, where σ is a base type.*
2. *$R^{\sigma\to\tau}_w f_1 \ldots f_n$ if and only if, for every $v \geq w$ and $a_1 \in A^\sigma_1, \ldots, a_n \in A^\sigma_n$, if $R^\sigma_v a_1 \ldots a_n$ then $R^\sigma_v f_1(a_1) \ldots f_n(a_n)$.*

As before, we can treat this as an inductive definition of R^σ_w. The first condition is often called 'persistence'. Our definition only requires that $R^e_\cdot$ and $R^t_\cdot$ be persistent, but it follows from the definition that R is persistent at all types. The reader is encouraged to work through

this proof on their own, for the case of binary Kripke logical relations between two structures A and B:

Exercise 16.23. *Suppose that $R^e_w \subseteq A^e \times B^e$ and $R^t_w \subseteq A^t \times B^t$ for each $w \in W$ and suppose, moreover, that $R^e_.$ and $R^t_.$ are persistent: $R^e_w \subseteq R^e_v$ and $R^t_w \subseteq R^t_v$ whenever $w \leq v$.*

Show, by induction on types, that $R^\sigma_.$ is persistent for each type σ: if $R^\sigma_w ab$ and $w \leq v$ then $R^\sigma_v ab$.

We may introduce the notion of invariance for Kripke logical relations:

Definition 16.9. *Given an n-ary Kripke logical relation R on A (so $R^\sigma_w \subseteq A^\sigma \times A^\sigma \times \ldots \times A^\sigma$) we say that $d \in A^\sigma$ is* invariant *under R iff for every $w \in W$, $R^\sigma_w dd \ldots d$.*

Exercise 16.24. *Consider a full Henkin structure where $A^t = \mathbb{N}$. Let $q : \mathbb{N} \to \mathbb{N}$ be any function mapping natural numbers to natural numbers.*

a. Show that the function $f \in A^{(t\to t)\to t\to t}$ defined by:

$$f(g)(n) = g^{q(n)}(n)$$

is invariant under every unary Kripke logical relation. I.e. if H is a unary Kripke logical relation on some Kripke frame, $(W, \leq)$, then $H^{(t\to t)\to t\to t}_w f$ for every $w \in W$. Conclude that some elements of $A^{(t\to t)\to t\to t}$ that are invariant under unary Kripke logical relations are not λ-definable (i.e. are not iterators).

b. Using Proposition 16.3 show that if f is invariant under every binary Kripke logical relation it is an iterator.[iii]

Given $\mathcal{J}(\Sigma)$-interpretations $(A, [\![\cdot]\!]_A)$ and $(B, [\![\cdot]\!]_B)$, that are $\mathcal{J}(\Sigma)$-interpretations of a λ-language $\mathcal{J}(\Sigma)$, a Kripke logical relation between them is a Kripke logical relation between the underlying applicative structures, over a frame $(W, \leq)$, such that $R^\sigma_w [\![c]\!]_A [\![c]\!]_B$ for every $c \in \Sigma^\sigma$ and $w \in W$. Like logical relations, we can prove that the denotations of λ-terms are always related by logical relations.

Theorem 16.5 (The Fundamental Theorem (Kripke logical relations))**.** *Suppose that $(A, [\![\cdot]\!]_A)$ and $(B, [\![\cdot]\!]_B)$ are $\mathcal{J}(\Sigma)$-interpretations of a λ-language $\mathcal{L}(\Sigma)$, and R is a Kripke logical relation between these interpretations on the Kripke frame $(W, \leq)$. Then for any pair of closed terms $M : \sigma$:*

$$R^\sigma_w [\![M]\!]_A [\![M]\!]_B \text{ for every } w \in W.$$

By analogy with our previous notation, write $R_w gh$ iff $R^\sigma_w g(x) h(x)$ for each type σ and variable x of type σ. The fundamental theorem is again proven by showing the more general result that if $R_w gh$ then $R^\sigma_w [\![M]\!]^g_A [\![M]\!]^h_B$ for arbitrary open or closed terms M.

Proof. We prove this by induction on term structure. Suppose $R_w gh$. For constants, it's guaranteed that $R^\sigma_w [\![c]\!]^g_A [\![c]\!]^h_B$ by the definition of a logical relation between $\mathcal{J}(\Sigma)$-interpretations. For variables, it follows that $R^\sigma_w [\![x]\!]^g_A [\![x]\!]^h_B$ because $[\![x]\!]^g_A = g(x)$, $[\![x]\!]^h_B = h(x)$ and we have assumed that $R_w gh$.

Suppose for induction that $M : \sigma \to \tau$ and $N : \sigma$ and that whenever $R_w gh$, $R^{\sigma\to\tau}_w [\![M]\!]^g_A [\![M]\!]^h_B$ and $R^\sigma_w [\![N]\!]^g_A [\![N]\!]^h_B$. Thus, assuming $R_w gh$, we know that $R^\tau_w \mathrm{App}([\![M]\!]^g_A, [\![N]\!]^g_B) \mathrm{App}([\![M]\!]^h_B, [\![N]\!]^h_B)$ using the condition for logical relations at type

$\sigma \to \tau$, the fact that $w \leq w$, and the fact that $R^{\sigma}_{w}[\![N]\!]^{g}_{A}[\![N]\!]^{h}_{B}$. Finally, this means that $R^{\tau}_{w}[\![MN]\!]^{g}_{A}[\![MN]\!]^{h}_{B}$ (when $v = w$) as this is how the interpretation function is defined.

Now suppose that the inductive hypothesis holds for $M : \tau$. We want to show it holds for $\lambda x.M : \sigma \to \tau$. Suppose $R_{w}gh$. To show $R^{\sigma\to\tau}_{w}[\![\lambda x.M]\!]^{g}_{A}[\![\lambda x.M]\!]^{h}_{B}$ it suffices to show that for any $v \geq w$, $a \in A^{\sigma}$ and $b \in B^{\sigma}$, if $R^{\sigma}_{v}ab$ then $R^{\tau}_{v}\,\mathrm{App}([\![\lambda x.M]\!]^{g}_{A}, a)\,\mathrm{App}([\![\lambda x.M]\!]^{h}_{B}, a)$.

Here our reasoning will depend on whether we are dealing with functional or non-functional interpretations. In either case, we can establish that $\mathrm{App}([\![\lambda x.M]\!]^{g}_{A}, a) = [\![M]\!]^{g[a/x]}_{A}$ and $\mathrm{App}([\![\lambda x.M]\!]^{h}_{B}, b) = [\![M]\!]^{h[b/x]}_{B}$. This suffices to complete the argument: since $R^{\sigma}_{v}ab$, we have $R_{v}g[a/x]h[b/x]$, so by our inductive hypothesis $R^{\tau}_{v}[\![M]\!]^{g[a/x]}_{A}[\![M]\!]^{h[b/x]}_{B}$, so $R^{\tau}_{v}\,\mathrm{App}([\![\lambda x.M]\!]^{g}_{A}, a)\,\mathrm{App}([\![\lambda x.M]\!]^{h}_{B}, a)$ as required.

The identities $\mathrm{App}([\![\lambda x.M]\!]^{g}_{A}, a) = [\![M]\!]^{g[a/x]}_{A}$ and $\mathrm{App}([\![\lambda x.M]\!]^{h}_{B}, b) = [\![M]\!]^{h[b/x]}_{B}$ are just built into the definition of a functional interpretation. With non-functional interpretations, we need a slightly more involved chain of identities.

$$\mathrm{App}([\![\lambda x.M]\!]^{g}_{A}, a) = \mathrm{App}([\![\lambda x.M]\!]^{g[a/x]}_{A}, [\![x]\!]^{g[a/x]}_{A}) = [\![(\lambda x.M)x]\!]^{g[a/x]}_{A} = [\![M]\!]^{g[a/x]}_{A}$$

The first identity holds because (i) $[\![x]\!]^{g[a/x]}_{A} = a$ and (ii) g and $g[a/x]$ agree on the free variables in $\lambda x.M$, and it is built into the definition of $[\![\cdot]\!]$ that the interpretation of a term relative to any two assignments agreeing on the free variables in that term is the same. The second identity holds due to the definition of the interpretation of applicative terms. The third identity holds because it is built into the definition of $[\![\cdot]\!]$ that it is invariant under $\beta\eta$-equivalence. □

In fact, the relationship between being λ-definable and invariance under Kripke logical relations is much tighter than the corresponding relationship between λ-definability and invariance under ordinary logical relations. For instance, Plotkin (1980) establishes that in the full Henkin structure with infinite base types, an element of a domain is λ-definable if and only if it is invariant under every ternary Kripke logical relation. Jung and Tiuryn (1993) introduce a generalization of Kripke logical relations—in which the arity of the Kripke logical relation can increase along the accessibility relation $\leq$—and show that invariance under the more general notion coincides with λ-definability in an arbitrary applicative structure.

Endnotes

1. Fine (1977) and Stalnaker (2012). See Fritz and Goodman (2016), Fritz (2018a,b) for a detailed exploration of these theories.
2. More generally, $[\![M]\!]^{g} = [M[N_1/x_1, \ldots, N_k/x_k]]_{\beta\eta}$ where $x_1, \ldots, x_k$ are the free variables in M, and $N_1, \ldots, N_k$ are arbitrarily chosen elements of the equivalence classes $g(x_1), \ldots, g(x_n)$.
3. For example, any function f that maps a to c and b to a, because a and b are equivalent, and c and a are not.
4. Of course, 'John believes that' only contingently has the truth-functional behaviour of negation. A theory such as Classicism implies that any connective that *necessarily* has the same truth-functional behaviour as negation is identical to negation. But no such theorem exists in H, and would presumably be refutable in structural theories that prove the distinctness of $\neg$ and $\lambda p.(\neg\neg\neg p)$.
5. The following examples are taken from Plotkin (1980). See also Plotkin (1973).
6. Defining it inductively, $g^{0}(a) = a$, $g^{n+1}(a) = g(g^{n}(a))$.
7. Mike Gordon originally proposed invariance under logical relations R of arbitrary arity on a structure A as a way of semantically characterizing λ-definability. That this conjecture is false follows from Ralph Loader's results concerning the undecidability of λ-definability Loader (2001).

8. His stated guiding thought here is that one 'intepret the implication sign in the definition of $R^{\sigma\to\tau}$ in an intuitionistic way, hoping thereby to make any f satisfying $R^{\sigma\to\tau}$ more likely to be constructive, and therefore λ-definable. In order to do this, [he uses] Kripke's ideas on the interpretation of intuitionist logic.' Plotkin (1980), p. 367.
9. They also play a role in the modelling of propositional logics containing intuitionistic logic, which is what is motivating Plotkin in the above.

Hints for exercises

[i] **Hint:** Observe that $[\![\lambda x_1 \ldots x_n.M]\!]^g g(x_1)\ldots g(x_n) = [\![M]\!]^g$ for any M.

[ii] **Hint:** First show that every domain contains at least two elements that are not related to one another, and use the existence of distinct but cofunctional entities and the fullness of the structure to construct an example as in the previous exercise.

[iii] **Hint:** The trick is to find a Kripke frame (W, R) such that the notion of a binary Kripke logical relation of (W, R) coincides with the ordinary notion of a binary Kripke logical relation, so you may appeal to Proposition 16.3

17

Modalized sets, M-sets and cartesian closed categories

In this chapter, we outline a more abstract approach to the interpretation of full λ-languages. We begin, in Sections 17.1–17.2, by outlining two particular kinds interpretations of full λ-languages based on the idea of a possible world and a metaphysical substitution respectively. In Section 17.3, we discuss some applications of the latter to the modeling of the primitives, such as metaphysical definability and fundamentality, introduced in Section 13.1. In Sections 17.4 and 17.5, we take a more abstract perspective to the notions of 'fullness' and 'functionality' that played a central role in our earlier discussions. Using the tools of category theory we will specify general conditions—being 'cartesian closed'—which a class of mathematical objects must satisfy in order to have the structure needed to interpret the full λ-language. Finally in Section 17.6 we briefly present a final example of a category of mathematical structures that unifies the modalized and substitution structures by including them as limiting cases.

In Chapters 14 and 15 we constructed interpretations of λ-languages in various applicative structures. Among those structures, the full Henkin structures—in which the type $A^{\sigma\to\tau}$ consists of the set of all functions from A^{σ} to A^{τ}—can be singled out as having several special properties.

First, they are full. This means that the function $a \mapsto [\![M]\!]^{g[a/x]}$ required to interpret the λ-term $\lambda x.M$ is always in the relevant functional domain. When we are constructing non-full applicative structures where we are restricting which functions from A^{σ} to A^{τ} are realized by elements of $A^{\sigma\to\tau}$ we have to be mindful that we are not omitting functions necessary for the interpretation of λ-terms; we have to choose restrictions that let the combinators in. In the full Henkin structure, we don't have to worry about this. It is a general fact that full Henkin structures have combinators; this is not something that needs to be reproved every time we encounter a different full Henkin structure. Note that non-full applicative structures are needed to construct model theories that invalidate certain principles like Functional Plenitude from Section 6.5, or Brouwer's principle in Section 8.1.

Second, they are functional. In Chapter 15, we constructed compositional interpretations of λ-languages over functional applicative structures with combinators. Compositionality ensures that all we have to do in order to interpret the whole language is supply interpretations of the constants. The rest of the process is on rails: the interpretation of the complex terms is inductively determined by their parts (relative to assignments). The situation was less satisfactory for non-functional applicative structures: the interpretation of complex terms was no-longer defined compositionally in terms of the interpretations of the constants. Moreover, the interpretations of the complex terms were not in general determined uniquely from the interpretations of the constants. Some ingenuity is thus needed

DOI: 10.4324/9781003039181-22

to construct a non-functional λ-model, since the interpretation of the complex terms needs to be chosen carefully so as to satisfy some highly non-trivial constraints. For compositional interpretations, no such ingenuity is needed: any choice of interpretations of the constants uniquely extends to an interpretation of all terms. The difficulty of constructing non-functional $\mathcal{L}(\Sigma)$-interpretations is unsatisfactory because we need non-functional interpretations in order to construct model theories for logics that do not include various contentious principles of metaphysics, such as the Functionality principle of Section 6.5, or the Barcan formula from Section 8.1.

Finally, the combination of both fullness and functionality enjoyed by full Henkin structures means that a Henkin structure is fully determined by the choice of sets A^e and A^t for the base types. This sort of uniqueness does not hold in the class of arbitrary functional structures, even up to isomorphism: two functional structures can agree about the domains for type σ and τ but disagree about which functions between these domains are realized by elements in the domain of type $\sigma \to \tau$. Nor does this sort of uniqueness hold among the class of all full structures (even up to isomorphism), since structures agreeing about the domains of type σ and τ can still disagree about how many elements of the domain $\sigma \to \tau$ realize a given function between the two domains.

In this chapter, we'll introduce weakenings of the notions of functionality and fullness that allow us to construct applicative structures that are non-functional and non-full in the sense of Chapter 14, but which still enjoy all of the aforementioned benefits of full Henkin structures. In Chapter 18, we'll apply these ideas to construct models in which various principles like Functional Plenitude, Brouwer's principle, Functionality, the Barcan formula, and so on, fail. The key to ensuring the final property—that A^σ and A^τ uniquely determine $A^{\sigma\to\tau}$—is to endow the domains with more information than is encoded by the mere set of their elements. Typically this will involve identifying the domains with sets with some further structure, and then $A^{\sigma\to\tau}$ can be uniquely identified with the set of functions that preserve that structure, endowed with the same sort of structure so that it can be fed into higher types. Examples of sets endowed with extra structure with accompanying notions of structure-preserving maps are ubiquitous in mathematics—see, for example, groups, vector spaces, rings, and so on—and the abstract study of such classes of objects and the maps between them is called category theory. This will play an important role in this chapter.

The material in this chapter and the next is slightly more advanced than in previous chapters. The pace will be slightly faster, and more routine proofs will be left to the reader as exercises.

17.1 Modalized applicative structures

Classicism has among its theorems the converse Barcan formula and the necessity of identity at each type. At type e, this means that if two individuals are identical they are necessarily identical, and that things can't "go out of existence"—if an individual exists it necessarily exists. By contrast, we asserted (without proof) that the necessity of distinctness and the Barcan formula—which, crudely speaking, ensure that distinct things are necessarily distinct and that things can't "come into existence'—were not theorems of Classicism.

One can model these phenomena in a classical first-order modal logic in a possible worlds framework. As usual one begins with a set, W, of worlds, and an accessibility relation defined over W telling us which worlds are possible relative to others. To model the identity and existence of individuals at different worlds we introduce the notion of a *modalized set*

D, which tells us which individuals exist at each worlds and which individuals are identified there.[1] A modalized set is a world-indexed family of sets, D_w, for each world w, whose elements are to be thought of as representing each individual that exists at that world. It is important to maintain a fairly abstract view of the formalism: we do not, for instance, *identify* the individuals being represented by elements of D_w with the elements of D_w—indeed, we could have different elements of D_w and D_v representing the same individual. Thus we have a function $i_{wv} : D_w \to D_v$ that maps the representative of an individual at w to its representative at v (provided v is accessible to w). The fact that existence is necessary, is reflected by the fact that if v is accessible to w, then every individual that is represented at w, by an element $a \in D_w$, is also represented at v, by an element of D_v, namely $i_{wv}a$. The fact that what might exist needn't in fact exist is reflected by the fact that not every element of D_v need be the representative of some individual represented at w: there may be 'new' individuals at v, elements of D_v that are not in the range of this function i_{wv}. The fact that identity is necessary is reflected by the fact that functions always map identical things to identical things, and the fact that distinctness is not necessary by the fact functions need not map distinct things to distinct things. Cases where the function i_{wv} is not injective thus represent a failure of the necessity of distinctness.

The logical situation with respect to identity and existence is the same for propositions, relations, operators and so on. So where, in Chapter 15, we modeled these entities with sets, A^t, $A^{t\to t}$, $A^{e\to e\to t}$, and so on, we could instead model these entities with sets with further 'modal' structure telling us relative to each world which propositions, relations, operators, etc. exist and are identical there. So we will begin by examining a mathematical object obtained by replacing a domain—a plain set, A—with something that has this extra modal structure consisting of a domain for each world, and counterpart functions for identifying entities across worlds. In this section, we see how by replacing sets and set-theoretic constructions with these structures and analogous operations on them, we can obtain new classes of interpretations of the full λ-language. In Chapter 18, we will use them to obtain models of Classicism which invalidate the necessity of distinctness and the Barcan formula.

First, recall Definition 16.7

Definition 17.1 (Kripke Frame)**.** *A pointed Kripke frame, or just a* frame*,* $\mathcal{F} = (W, \leq, @)$ *consists of a set* W*, a preorder (a transitive reflexive relation)* $\leq$ *on* W*, and a* designated world $@ \in W$ *such that* $@ \leq w$ *for all* $w \in W$*.*

Definition 17.2. *Given a subset* $p \subseteq W$ *and* $w \in W$*, we will write* $p{\uparrow}w$ *for the set* $\{x \in p \mid w \leq x\}$ *and call this the* truncation *of* p *by* w*.*

For the rest of this section, we will assume that a particular fixed frame, $(W, \leq, @)$, has been chosen, and that subsequent mentions of W and $\leq$ refer to that particular frame. However, the reader should be aware that subsequent definitions all depend on this choice, and that a different choice would yield a different class of mathematical structures.

There are two reasons to restrict attention to preorders.[2] One of the intended applications of this formalism is to find models of Classicism (see Chapter 18). In Classicism, the logical connectives give rise to certain modal notions, including the operator we called 'broad necessity': $\Box := \lambda p.p = \top$. Moreover, we could prove all theorems of the modal logic **S4** for $\Box$. Consequently, we have required the accessibility relation, $\leq$, to be transitive and reflexive, for it is a well-known fact from modal logic that **S4** is the modal logic of transitive reflexive accessibility relations.[3] Aside from applications to Classicism, we also expressed interest in constructing non-functional $\mathcal{L}$-interpretations, since the techniques of Chapter

14 were insufficient. The modalized $\mathcal{L}$-interpretations considered in this section achieve this, but in order to show that λ-terms have denotations in these structures $\leq$ must be a preorder. This is related to the connection between preorders in the modeling of intuitionistic logic due to Kripke (see Appendix B), and on the connection between intuitionistic logic and the interpretation of typed λ-terms (see Appendix B).

Definition 17.3 (Modalized set)**.** *A modalized set $(D_., i^D_.)$ for a frame $(W, \leq, @)$ consists of*

- *A set, D_w, for each world $w \in W$.*
- *A function, $i^D_{wv} : D_w \to D_v$, for any pair of worlds w and v such that $w \leq v$ such that:*

  – *$i^D_{ww} : D_w \to D_w$ is the identity function*
  – *Whenever $x \leq y \leq z$, $i^D_{xz} = i^D_{yz} \circ i^D_{xy}$*

For convenience, we will often refer to a modalized set $(D., i_{wv})$ by the first component, D, and will drop the superscript of i^D_{wv} when no ambiguity arises.

Observe that we motivated the mathematical definition by thinking of each element of D_w as representing an individual (or other kind of entity), but these individuals themselves make no appearance in the mathematical definition. This way of thinking is nonetheless helpful for gaining an intuition for various definitions that follow.

We will refer to D_w as the 'domain at world w', and will talk of the elements of D_w as representing individuals that 'exist at w'. If $a \in D_w$ represents a given individual at w, then the element $i_{wv}(a) \in D_v$ represents that same individual at D_v. Thus if an individual exists at w (is represented by any element of the model), then it exists at every world accessible to w, but not necessarily at every world that accesses w. This accords with our previous remarks about the necessity of existence encoded by the converse Barcan formula (but not the necessity of non-existence/the Barcan formula). Similarly, if individuals have the same representative at w, $a \in D_w$, then they will have the same representative at every world v accessible to w, namely $i_{wv}a$, in accord with the necessity of identity (but not the necessity of distinctness).

Example 17.1. *For any pointed Kripke frame $\mathcal{F}$, we will write $B(\mathcal{F})$ for the modalized set $(D., i.)$ where for each w, $D_w = P(W{\uparrow}w)$, the set of sets of worlds accessible to w, and given $p \in P(W{\uparrow}w)$, $i_{wv}(p) = p{\uparrow}v \in W{\uparrow}v$.*

A modalized set is properly expanding iff there exist $x, y \in W$ with $x \leq y$ such that i_{xy} is not surjective. That is, there exists $a \in D_y$ such that $i_{xy}b \neq a$ for any $b \in D_x$.

Exercise 17.1. *Show that the previous example is a modalized set. Is it properly expanding? Suppose that $\leq$ is not an equivalence relation. Show that there must be 'contingent distinctness': that there exist $p, q \subseteq W{\uparrow}w$ and $x, y \in W$ such that $x \leq y$, $p \neq q$ and $i_{xy}p = i_{xy}q$.*

Example 17.2 (Product of Modalized Sets)**.** *For any two modalized sets $(A., i^A_.)$ and $(B., i^B_.)$ their product is $((A \times B)., i^{A\times B}_.)$ where*

- $(A \times B)_w = (A_w \times B_w)$
- $i^{A\times B}_{xy}(a, b) = (i^A_{xy}a, i^B_{xy}b)$.

Exercise 17.2. *Prove that the product of two modalized sets is a modalized set.*

A key idea we will encounter frequently is the notion of a 'truncation about w'. When we have some operation or other mathematical object defined on all of W we can truncate that operation or object by restricting its behaviour to the worlds above and including w. In general, we denote the truncation by w using $\uparrow w$. So, as in Definition 17.2, the truncation of a set of worlds $p \subseteq W$ about w is written $p \uparrow w$. The truncation of a frame $\mathcal{F} = (W, \leq, @)$ by w is the result of throwing away the worlds that are not above w from the frame: $\mathcal{F} \uparrow w = (W \uparrow w, \leq \cap (W \uparrow w) \times (W \uparrow w), w)$. Quite often the counterpart function i_{xy} will be a truncation of sorts, as we have already seen in the definition of $B(\mathcal{F})$. We truncate a modalized set A about w by truncating the underlying frame and consequently throwing away the set A_v where $v \not\leq w$.

Example 17.3 (Truncation of a modalized set)**.** *If A is a modalized set on the frame $(W, \leq, @)$ and $w \in W$, then $A \uparrow w$ is a modalized set on the truncated frame $(W \uparrow w, \leq, w)$, where*

- $(A \uparrow w)_v = A_v$ *for each* $v \geq w$
- $i^{A \uparrow w}_{xy} a = i^A_{xy} a$ *whenever* $x \geq y \geq w$.

It's often useful to know when two modalized sets are 'the same', or isomorphic. Let's begin with the notion of a structure-preserving map between modalized sets, A and B, over the same frame—a *homomorphism*. Intuitively, we think of this as a way of mapping the entities being represented by A to the entities being represented in B. Since A and B consist of the representatives of such entities, not the entities themselves, these mappings will be identified with functions on these representatives. And because the same entity can be represented by different elements of A and B at different worlds, we need to make sure these mappings cohere, in the sense that these functions preserve the entity being represented at each world. A little more precisely, a homomorphism will consist of a collection of unary functions for each world w, mapping the domain at w, A_w, of the former modalized set to the domain at w, B_w of the latter. We can represent this as a binary function $f(\cdot, \cdot)$ where for each world w, $f(w, \cdot) : A_w \to B_w$ is the unary function described.

We require, then, that if $a \in A_x$ and $b \in A_y$ represent the same individual at worlds x and y where $x \leq y$—i.e. $b = i^A_{xy} a$—then $f(x, a) \in B_x$ and $f(y, b) \in B_y$ must also represent the same individual—i.e. $i^B_{xy}(f(x, a)) = f(y, b)$. Putting these two identities together, we must always have $i^B_{xy}(f(x, a)) = f(y, i^A_{xy} a)$. A homomorphism must thus 'commute' with the counterpart function.

Definition 17.4 (Homomorphism of modalized sets)**.** *A homomorphism $f : A \to B$ of modalized sets A and B is a world indexed collection of functions $A_w \to B_w$, which we represent with a binary function f taking a world w in its first argument and an element of A_w in its second, and yielding a value in B_w:*

- $f(w, \cdot) : A_w \to B_w$ *for each* $w \in W$.
- $f(y, (i^A_{xy} a)) = i^B_{xy} f(x, a)$ *whenever* $x \leq y$ *and* $a \in D_x$.

Note that we use the same notation as we used for functions between sets, so that $f : A \to B$ means a different thing when A and B are modalized sets than when they are sets.

Example 17.4 (The identity homomorphism)**.** *For any modalized set, A, $id^A : A \to A$ is defined by $id^A(w, a) = a$ for every $w \in W$ and $a \in A_w$*

Example 17.5 (Composition of homomorphisms)**.** *If $f : A \to B$ and $g : B \to C$ are homomorphisms, then there is a homomorphism $g \circ f : A \to C$ defined as $(g \circ f)(w, a) = g(w, f(w, a))$.*

Example 17.6 (Truncation of a homomorphism)**.** *If $f : A \to B$ and $w \in W$, the truncation of f about w is the homomorphism $(f \uparrow w) : A \uparrow w \to B \uparrow w$ defined by restricting fs behaviour to the worlds at w and above:*

$(f \uparrow w)(v, a) = f(v, a)$ *whenever* $v \geq w$ *and* $a \in A_v$.

Exercise 17.3. *a. Show that $id_A : A \to A$ is a homomorphism*

b. Show that $g \circ f : A \to C$ is a homomorphism whenever $f : A \to B$ and $g : B \to C$ are.

Definition 17.5. *A homomorphism $f : A \to B$ is an isomorphism iff there exists a homomorphism $g : B \to A$ such that $f \circ g = id_A$ and $g \circ f = id_A$.*

Recall that an applicative structure assigns to each type σ a set A^σ, and for each pair of types σ and τ a notion of application $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$. We may thus introduce a notion of a 'modalized' structure by analogy: one assigns to each type σ a modalized set A^σ, and for each pair of types σ and τ a notion of application $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$. Since application is intuitively an operation on the entities being represented by a modalized set, not the representatives themselves, $\mathrm{App}^{\sigma\tau}$ should be a homomorphism. Applying the representative w of a functional entity at w to the representative of another entity at w and finding the representative of the result at a world v accessible to w should be the same as finding the representative of the function and argument entities at v and applying one to the other. In symbols: $i^\tau_{wv}\,\mathrm{App}^{\sigma\tau}(d, a) = \mathrm{App}^{\sigma\tau}(i^{\sigma\to\tau}_{wv} d, i^\sigma_{wv} a')$.

Definition 17.6 (Modalized applicative structure)**.** *A* modalized applicative structure $(A^{\cdot}, \mathrm{App}^{\cdot})$ *for a frame $(W, \leq, @)$ consists of a family A^σ for each type σ, and a family $\mathrm{App}^{\sigma\tau}$ for each pair of types σ and τ such that:*

- *A^σ is a modalized set.*
- *$\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$ is a homomorphism of modalized sets, i.e. for any $w, v \in W$ with $w \leq v$, $d \in A^{\sigma\to\tau}_w, a \in A^\sigma_w$:*

$$- \; i^\tau_{wv}\,\mathrm{App}^{\sigma\tau}(w, (d, a)) = \mathrm{App}^{\sigma\tau}(v, (i^{\sigma\to\tau}_{wv} d, i^\sigma_{wv} a))$$

Applicative structures can be categorized as full, or as functional depending on whether the association of elements of $A^{\sigma\to\tau}$ to functions $A^\sigma \to A^\tau$ (applicative behaviors) is surjective or injective: every applicative behaviour is realised by an element of the model in the first case, and by at most one element in the second.

There are corresponding notions in a modalized applicative structure. The analogue of the applicative behaviour of an element of $A^{\sigma\to\tau}$ is a map that takes each element $d \in A^{\sigma\to\tau}_w$ to a homomorphism $A^\sigma \uparrow w \to A^\tau \uparrow w$.

Given $w \in W$ and $d \in A^{\sigma\to\tau}_w$, let $\overline{\mathrm{App}}^{\sigma\tau}(w, d) : A \uparrow w \to B \uparrow w$ be the homomorphism defined as $(v, a) \mapsto \mathrm{App}^{\sigma\tau}(v, (d, a))$ where $v \geq w, a \in A_v$.

Definition 17.7. *A modalized applicative structure is:*

- Quasi-functional *iff* $\overline{\mathrm{App}}^{\sigma\tau}(w,\cdot)$ *is injective.*
  I.e. for every σ *and* τ, $w \in W$, *and* $d, d' \in A_w^{\sigma\to\tau}$, *if for every* $v \geq w$ *and* $a \in A_v^{\sigma}$, $\mathrm{App}^{\sigma\tau}(v,(i_{wv}d,a)) = \mathrm{App}^{\sigma\tau}(v,(i_{wv}d',a))$, *then* $d = d'$.
- Quasi-full *iff* $\overline{\mathrm{App}}^{\sigma\tau}(w,\cdot)$ *is surjective.*
  I.e. for every σ *and* τ, $w \in W$ *and homomorphism* $f : A^{\sigma}\uparrow w \to A^{\tau}\uparrow w$, *there exists a* $d \in A_w^{\sigma\to\tau}$ *such that* $\mathrm{App}^{\sigma\tau}(w,(d,a)) = f(w,a)$ *for every* $a \in A_w^{\sigma}$.
- Has combinators *iff for every* σ,τ,ρ *there are elements* $k^{\sigma\tau} \in A_{@}^{\sigma\to\tau\to\sigma}$ *and* $s^{\sigma\tau\rho} \in A_{@}^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$ *such that, for each* $w \in W$:

  – $\mathrm{App}(w,(\mathrm{App}(w,(i_{@w}k^{\sigma\tau},a)),b)) = a$ *for every* $a \in A_w^{\sigma}, b \in A_w^{\tau}$.

  – $\mathrm{App}(w,(\mathrm{App}(w,(\mathrm{App}(w,(i_{@w}s^{\sigma\tau\rho},d)),e)),a)) = (\mathrm{App}(w,(d,a)),\mathrm{App}(w,(e,a)))$ *for every* $d \in A_w^{\sigma\to\tau\to\rho}$, $e \in A_w^{\sigma\to\tau}$, *and* $a \in A^{\sigma}$.

Quasi-functionality and quasi-fullness are weakenings of the corresponding notions for applicative structures. Quasi-functionality articulates, in the language of possible worlds, the idea that necessarily cofunctional operations are identical (the reader should pause for a moment to convince themselves of this). Quasi-fullness effectively states that any function from the σ entities to the τ entities existing at w should have a counterpart in the entities of type $\sigma \to \tau$ existing at w, here taking into account our earlier remarks that functions on the entities represented in a modalized set correspond to homomorphisms on the representatives that officially constitute the modalized set.

Each modalized applicative structure and world determines an ordinary applicative structure.

Definition 17.8. *Any modalized applicative structure,* $(A^{\cdot},\mathrm{App}^{\cdot})$, *determines an ordinary applicative structure whose type* σ *domain is the set* $A_{@}^{\sigma}$ *and whose application function is* $\mathrm{App}(@,\cdot)$.

Observe that the designated world hasn't played a role prior to this definition.

With the notion of a modalized applicative structure in hand, we may define certain class of modalized applicative structures that stand to modalized sets as full Henkin structures stand to sets.

Crucial to the definition of a full Henkin structure is the operation on sets that takes two sets, A and B, and produces another set, $A \to B$ consisting of all functions from A to B which is full and functional in the abstract sense analysed in Chapter 14. We will thus begin by introducing an operation that takes two modalized sets A and B, and produces another modalized set $A \Rightarrow B$ which we will later show to be 'full' and 'functional' an abstract sense that is analogous to that enjoyed by the set-theoretic function space. $(A \Rightarrow B)_w$—the operations that exist according the world w—should consist of all the functions from A-entities that exist at w to B-entities that exist at w. As we discussed previously, we officially only *represent* A-entities and B-entities at each world, w, by elements in A_w and B_w respectively, and what we would think of as a function between such entities is properly represented by a collection of functions between these representatives of that is sufficiently well-behaved in the sense captured in our definition of *homomorphism*. We thus identify $(A \Rightarrow B)_w$ with the set of all homomorphisms from $A\uparrow w$ to $B\uparrow w$. These homomorphisms

'start' at w, since we only care about what happens from w onwards. The importance of this stipulation is explored in Exercise 17.4

Definition 17.9 (Exponential of Modalized sets)**.** *For any two modalized sets (A, i^A) and (B, i^B) their exponential is $(A \Rightarrow B, i^{A\Rightarrow B})$ where*

1. $(A \Rightarrow B)_w = \{f \mid f : A{\uparrow}w \to B{\uparrow}w \text{ a homomorphism}\}$.
2. *When $x \leq y$ and $f : A{\uparrow}x \to B{\uparrow}x$, the homomorphism $i^{A\Rightarrow B}_{xy} f : A{\uparrow}y \to B{\uparrow}y$ is defined by 'truncation':*

   when $z \geq y$, $(i^{A\Rightarrow B}_{xy} f)(z, a) = f(z, a)$. I.e. $i_{xy} f = f{\uparrow}y$.

The following exercise demonstrates the importance of letting all homomorphisms $f : A{\uparrow}w \to B{\uparrow}w$ into the functional domain at world w, and not merely those homomorphisms that are truncations of global homomorphisms $f : A \to B$.

Exercise 17.4. *Consider the Kripke frame $(\{0, 1\}, \leq, 0)$ where $\leq$ is the numerical ordering on $\{0, 1\}$. Suppose that D is the modalized set where $D_0 = \{a, b\}$ and $D_1 = \{a, b, c\}$, where a, b and c are all distinct and i_{01} is the inclusion mapping. Find a homomorphism $f : D{\uparrow}1 \to D{\uparrow}1$ such that $f \neq h{\uparrow}1$ for any homomorphism $h : D \to D$. Show that $D \Rightarrow D$ has 'new' elements at world 1—i.e. show that $i^{D\Rightarrow D}_{01}$ is not surjective.*

The reader should attempt the following exercises to gain some familiarity with the exponential of two modalized sets. The first order of business is to show that $A \Rightarrow B$ really satisfies the definition for being a modalized set, which is the first exercise below.

Exercise 17.5. *Show that $A \Rightarrow B$ is indeed a modalized set. That is, show that i_{xx} is the identity on $(A \Rightarrow B)_x$, and $i^{A\Rightarrow B}_{yz} \circ i^{A\Rightarrow B}_{xy} = i^{A\Rightarrow B}_{xz}$.*

Exercise 17.6. *Let A, B and C be modalized sets.*

- *Let $x \in W$. Define $k_x := (y, a) \mapsto (z, b) \mapsto i_{yz}a$ where $z \geq y \geq x$, $a \in A_y$, $b \in B_z$. Show that $k_x \in (A \Rightarrow (B \Rightarrow A))_x$ (i.e. that it is a homomorphism, $k_x : A{\uparrow}x \to (B \Rightarrow A){\uparrow}x$).*
- *For $x \in W$, define $s_x := (y, f) \mapsto (z, g) \mapsto (w, a) \mapsto f(w, a)(w, g(w, a))$ where these dummy variables range so that $w \geq z \geq y \geq x$, $f \in (A \Rightarrow (B \Rightarrow C))_y$, $g \in (A \Rightarrow B)_z$ and $a \in A_w$. Show s_x is in $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))_x$ for any $x \in W$.*
- *Show that $i_{xy}k_x = k_y$ and $i_{xy}s_x = s_y$*

Exercise 17.7. *Consider the frame $(\mathbb{N}, \leq, 0)$ where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers and $\leq$ is the ordinary ordering on the natural numbers. Let A be the modalized set where $A_n = P(\mathbb{N}{\uparrow}n)$ and $i_{nm}p = p{\uparrow}m$. Show that $(A \Rightarrow A)_0$ differs from the full set theoretic function space on $A_0 (= P(\mathbb{N}))$ by finding a function $g : A_0 \to A_0$ that is not of the form $f(0, \cdot)$ for $f \in (A \Rightarrow A)_0$.*

We will frequently come across iterated exponential function spaces of the form $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B$, which are used to represent n-ary operations and relations. Given $w_1 \leq \cdots \leq w_n$ and $a_k \in (A_k)_w$, for $k = 1 \ldots n$, it is often convenient to have an equivalent way of expressing the result of applying $f \in (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)_@$ to the arguments $(w_1, a_1) \ldots (w_n, a_n)$ in such a way that the arguments all belong to domains of the same world. The way to do this is to shift each of $a_1 \ldots a_{n-1}$ up to the domain of world w_n, and

apply the function to the resulting sequence of arguments all taken from the w_n domains: $(w_n, i_{w_1 w_n} a_1), \ldots (w_{n-1}, i_{w_{n-1} w_n} a_{n-1})(w_n, a_n)$.

Proposition 17.1. *Suppose that* $f \in (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)_{w_0}$, $w_1 \leq \cdots \leq w_n$, *and that* $a_k \in (A_k)_w$, *for* $k = 1 \ldots n$. *Then:*

$$f(w_1, a_1) \ldots (w_n, a_n) = f(w_n, i_{w_1 w_n} a_1), \ldots (w_{n-1}, i_{w_{n-1} w_n} a_{n-1})(w_n, a_n)$$

To illustrate, consider the $n = 2$ case: let $f \in (A \Rightarrow B \Rightarrow C)_{@}$, and consider $f(w, a)(v, b)$. Since f is a homomorphism $i_{wv}(f(w, a)) = f(v, i_{wv} a)$. Since $i_{wv}(f(w, a))$ is the truncation of the homomorphism $f(w, a) : B{\uparrow}w \to C{\uparrow}w$ by v, namely $f(w, a){\uparrow}v$, then $(i_{w,v}(f(w, a)))(v, b) = f(w, a)(v, b)$. So putting these identities together $f(w, a)(v, b) = f(v, i_{wv} a)(v, b)$.

An important corollary of this is that in order to define a homomorphism $f \in (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)_{w_0}$ all one needs to do is define its behaviour on arguments drawn from domains of a given fixed world.

Corollary 17.1. *A homomorphism* $f \in (A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow B)_{w_0}$ *is uniquely determined by its values on arguments of the form* $f(w, a_1) \ldots (w, a_n)$.

With the notion of the exponential of two modalized sets in hand, we can now create a structure that stands to modalized sets as Henkin structures stands to sets. Recall that in a Henkin structure the domain of type $\sigma \to \tau$, $A^{\sigma \to \tau}$ was a subset of the full function space $A^{\sigma} \to A^{\tau}$. The analogue of a subset for modalized sets is a *subdomain*:

Definition 17.10 (Subdomain)**.** *Let A and B be modalized sets. A is a subdomain of B, written* $A \leq B$, *iff*

- *For all* $x \in W$, $A_x \subseteq B_x$.
- *For all* $x \leq y$, *and* $a \in A_x$, $i^A_{xy} a = i^B_{xy} a$

Observe that the second condition ensures that A is 'closed' under Bs counterpart function, i^B: if $x \leq y$ and $a \in A_x$ then $i^B xy \in A_y$.

Definition 17.11 (Concrete modalized applicative structure)**.** *A* concrete modalized applicative structure *is a certain kind of quasi-functional modalized applicative structure, where for all types* σ *and* τ

- $A^{\sigma \to \tau} \leq A^{\sigma} \Rightarrow A^{\tau}$.
- $\mathrm{App}^{\sigma\tau}(w, (f, a)) = f(w, a)$.

A standard *modalized applicative structure is a concrete modalized applicative structure such that*

- $A^{\sigma \to \tau} = A^{\sigma} \Rightarrow A^{\tau}$.

In order to show that this is indeed a modalized applicative structure, we need to prove that App is a homomorphism. Thus we must show that App commutes with i: $i_{xy}\,\mathrm{App}(x, (f, a)) = \mathrm{App}(y, i_{xy}(f, a))$. By the definition of App and the fact that f is a homomorphism and commutes with i, we get the following chain of identities: $i_{xy}\,\mathrm{App}(x, (f, a)) = i_{xy} f(x, a) = f(y, i_{xy} a)$. Then, since we defined $i_{xy} f$ as the result of restricting f to the worlds

above y, we have $f(y, i_{xy}a) = (i_{xy}f)(y, i_{xy}a)$, and this $= \mathrm{App}(y, (i_{xy}f, i_{xy}a)$ by the definition of App, and finally $= \mathrm{App}(y, i_{xy}(f, a))$ by the definition of the product.

Exercise 17.8. *Show that every concrete modalized structure is indeed quasi-functional.*

In the next exercise, we show that some quasi-functional structures, indeed some standard structures, are not functional in the sense of Definition 14.6.

Exercise 17.9. *Consider the frame $(\{0,1\}, \leq, 0)$ where $\leq$ is the standard numerical ordering on $\{0,1\}$. Suppose that D is the modalized set where $D_0 = \{a\}$ and $D_1 = \{a, b\}$ where $a \neq b$. Consider the standard modalized applicative structure $(A^{\cdot}, \mathrm{App}^{\cdot}_B)$ where $A^e = A^t = D$. As in Definition 17.8 we obtain an ordinary applicative structure $(B^{\cdot}, \mathrm{App}^{\cdot}_B)$ obtained by setting $B^\sigma = A^\sigma_0$ as in Definition 17.8. Show that $(B^{\cdot}, \mathrm{App}^{\cdot}_B)$ is not functional.*

We showed in Chapter 16 that every functional applicative structure is isomorphic to a Henkin structure. We have a similar theorem in the present context:

Proposition 17.2. *Every quasi-functional modalized applicative structure $(A^{\cdot}, \mathrm{App}^{\cdot}_A)$ is isomorphic to a concrete structure $(B^{\cdot}, \mathrm{App}^{\cdot}_B)$: there exists a type indexed family of isomorphisms of modalized sets $h^\sigma : A^\sigma \to B^\sigma$ such that $h^\tau(App_A(d, a)) = App_B(h^{\sigma\to\tau}(d), h^\sigma(a))$ for every $a \in A^\sigma$ $d \in A^{\sigma\to\tau}$.*

The proof of this is similar to that of Theorem 16.3.

Recall the important role that functionality played in the theory of $\mathcal{L}$-interpretations. The functionality requirement ensured that if any element in the domain of type $\sigma \to \tau$ had the applicative behaviour of the function $a \mapsto [\![M]\!]^{g[a/x]}$ then it had it uniquely. Thus the interpretation of $[\![\lambda x.M]\!]^g$ was determined uniquely. We introduced a notion of $\mathcal{L}$-interpretation for non-functional applicative structures, but the clause for λ-terms had the form of a constraint that had to be satisfied—and could potentially be satisfied by multiple things—rather than an inductive definition. Here we'll see that the weaker property of quasi-functionality can play the same role of determining the interpretation of λ-terms uniquely in a modalized applicative structure. We have observed above that each modalized applicative structure determines a normal applicative structure, and moreover the determined applicative structure can fail to be functional even when the modalized applicative structure it is generated from is quasi-functional. So modalized applicative structures with combinators provide us with a rich source of examples of non-functional $\mathcal{L}$-interpretations.

Definition 17.12. *An interpretation of a signature Σ in a modalized applicative structure $(A^{\cdot}, \mathrm{App}^{\cdot}, [\![\cdot]\!])$ involves assigning each element of Σ^σ an element of $A^\sigma_@$:*

- *For each $c \in \Sigma^\sigma$, $[\![c]\!] \in A^\sigma_@$.*

The following definition shows how one can interpret a λ-language in a concrete modalized applicative structure (although by Proposition 17.2 we can transfer this to any quasi-functional modalized applicative structure). Ultimately we want to assign an interpretation from $A^\sigma_@$ to each term of the language of type σ. However, when our structure is not functional, differences in the applicative behaviour of a λ-term at non-designated worlds make for differences at the designated world, so in order to give a suitable inductive definition of the interpretation of λ-abstraction it will be necessary to assign interpretations to terms relative to non-designated worlds as well. We can talk about the interpretation of a constant c at other worlds w using the counterpart function $i_{@w}[\![c]\!]$. A variable assignment, g, for w is a function mapping variables of type σ into A^σ_w.

Definition 17.13 (Concrete modalized $\mathcal{L}$-interpretations)**.** *Let $\mathcal{L}(\Sigma)$ a λ-language. A concrete modalized $\mathcal{L}$-interpretation is an interpretation of Σ, $(A^{\cdot}, \mathrm{App}^{\cdot}, [\![\cdot]\!])$ where $(A^{\cdot}, \mathrm{App}^{\cdot})$ is a concrete modalized applicative structure with combinators. In this case, we can extend the interpretation function to assign values $[\![M]\!]^g_w \in A^\sigma_w$ whenever $M : \sigma$, $w \in W$ and g is a variable assignment for w:*

$[\![x]\!]^g_w = g(x)$.

$[\![c]\!]^g_w = i_{@w}[\![c]\!]$ *when* $c \in \Sigma$.

$[\![(MN)]\!]^g_w = \mathrm{App}^{\sigma\tau}(w, ([\![M]\!]^g_w, [\![N]\!]^g_w)) = [\![M]\!]^g_w(w, [\![N]\!]^g_w)$ *when* $M : \sigma \to \tau$, $N : \sigma$ *when* $M : \sigma \to \tau, N : \sigma$.

$[\![\lambda x.M]\!]^g_w = (v, a) \mapsto [\![M]\!]^{(i_{wv} \circ g)[a/x]}_v$ *when* $x : \sigma$, $M : \tau$. *($[\![\lambda x.M]\!]^g$ is undefined if this function is not in the structure, although we will show below that this never happens.)*

The following lemma is useful

Lemma 17.1. $i_{xy}[\![a]\!]^g_x = [\![a]\!]^{i_{xy} \circ g}_y$

The proof of this lemma is a routine induction on terms, but it is a good way to familiarize yourself with the clauses for interpreting λ-terms

Exercise 17.10. *Prove Lemma 17.1.*

We end by mentioning two theorems that guarantee that we can interpret the full λ-language in standard modalized applicative structures. We will not prove them, although the proofs follow the steps of similar proofs from Chapter 14. First, we must show, as we did in Theorem 14.1, that the interpretation of $\lambda x.M$ is always well defined. As in the proof of Theorem 14.1 for ordinary applicative structures with combinators, we need to choose a fairly strong inductive hypothesis: not only must $[\![M]\!]^g_w$ be well-defined, but a certain operation defined in terms of $[\![M]\!]^{\cdot}$ must be well-defined and belong to the interpretation.

Theorem 17.1. *If A is a modalized $\mathcal{L}$-interpretation, then for every term M of $\mathcal{L}^\tau(\Sigma)$ any variables $x_1, \ldots, x_n$ of type $\sigma_1, \ldots, \sigma_n$, and assignment g for w, not only is $[\![M]\!]^g_w$ well-defined, but the function*

$$(v_1, a_1) \mapsto \cdots \mapsto (v_n, a_n) \mapsto [\![M]\!]^{(i_{wv_n} \circ g)[i_{v_1 v_n} a_1/x_1, \ldots, i_{v_n v_n} a_n/x_n]}_{v_n}$$

is a well-defined homomorphism and belongs to $A^{\bar{\sigma} \to \tau}$.

Of course we need some guarantee that there are $\mathcal{L}$-interpretations. That is to say, concrete structures that have combinators. This is secured by the fact that the quasi-full modalized applicative structure—where $A^{\sigma\to\tau} = A^\sigma \Rightarrow A^\tau$—has combinators. Quasi-full modalized applicative structures are easy to construct: one simply chooses modalized sets for the base type A^e and A^t, and the remaining domains are completely determined by the exponential operation $\Rightarrow$.

Proposition 17.3. *Any interpretation over a standard modalized applicative structure has combinators.*

The reader has already proved this proposition in Exercise 17.6, where they showed that the s and k functions belong to the relevant modalized sets, and exist at all worlds.

17.2 Substitution structures

In this section, we will define a class of structures that have applications to metaphysical views in which properties, relations, propositions, etc. are structured entities themselves built out of simple properties, relations, etc. Russell, an early proponent of this sort of structural view, introduces the idea of complex facts as follows. After observing that a mark of complexity in a linguistic expression is the possibility of performing substitutions that map the simple words to other expressions of the same type, he writes that there is similarly 'a possibility of cutting up a fact into component parts, of which one component can be altered without altering the others'.[4] Let us think of a 'metaphysical substitution' as something which maps the simple properties, relations, etc. to other, possibly complex, properties, relations, etc. of the same type, just as a linguistic substitution maps constants to possibly complex terms of the same type. If a proposition is a structured entity built up from simple properties and relations then it is sensible to ask, given a metaphysical substitution, what proposition you would obtain from a given proposition by running through its simple constituents and replacing them according to the substitution. So Russell's observation suggests that someone who accepted a structured picture of reality could model propositions by a mathematical object that at least determines a set, A^t, with additional structure telling you how different metaphysical substitutions affect the propositions in A^t that are built from simple properties and relations. And, of course, since properties, relations, operators, etc. are also structured every type σ should similarly be associated with a mathematical object A^σ with at least that much structure. It will turn out that mathematical structures like these provide a natural mathematical model of the notion of metaphysical definability introduced in Section 13.1.

We begin by modeling the idea of a metaphysical substitution abstractly. We suppose there is an identity substitution, 1, and a binary operation $\circ$ on substitutions where $j \circ i$ represents the substitution obtained by first applying i and then applying j to the result. There should be no difference between applying a substitution and applying that substitution preceded or followed by the identity substitution, so $1 \circ i = i = i \circ 1$. And there is no difference between applying the substitution $(j \circ i)$ and then the substitution k, and between applying i and then $(k \circ j)$ so that $(k \circ j) \circ i = k \circ (j \circ i)$. Our abstract notion of substitution thus that of a monoid.

Definition 17.14 (Monoid)**.** *A monoid is a triple* $(M, 1, \circ)$ *where* $1 \in M$, $\circ : M \times M \to M$ *subject to the laws*

- $1 \circ i = i = i \circ 1$
- $(k \circ j) \circ i = k \circ (j \circ i)$

for all $i, j, k \in M$.

Let's begin by looking at some examples. The first is a straightforward linguistic example that reflects our opening ideas involving structured entities, and other two illustrate more abstract notions of substitution.

Example 17.7 (Concrete substitutions)**.** *A concrete substitution on a typed language* $\mathcal{L}(\Sigma)$ *is a function* $i : \Sigma \to \mathcal{L}(\Sigma)$ *mapping constants to closed terms of the same type: if* $c \in \Sigma^\sigma$ *then* $i(c)$ *is closed and belongs to* $\mathcal{L}^\sigma(\Sigma)$. *Thus, if* $F : e \to t$ *is a predicate constant, then* $i(F) : e \to t$ *is a (possibly complex) closed predicate.*

The concrete substitutions form a monoid: $j \circ i$ is the function that maps a constant c to the result of replacing each constant c' appearing in the term $i(c)$ with $j(c')$. 1 is the identity function from $\Sigma \to \Sigma$.

When Σ is a signature extending pure higher-order logic (so that $\Lambda \subseteq \Sigma$) a natural variation of the above example would be to consider the subclass of concrete substitutions that map the logical constants, Λ, to themselves.

Example 17.8. *X^X, the set of all functions from a set X to itself, is a monoid where $\circ$ denotes function composition and 1 the identity mapping.*

Example 17.9. *When X is a non-empty set, $X^{<\omega}$, the set of finite sequences of elements of X forms a monoid where $i \circ j$ denotes the concatenation of the two sequences i and j, and 1 is the empty sequence $()$.*

We will frequently refer to a monoid by the underlying set M when no ambiguity is present. For the rest of this section, we will assume a particular monoid, $(M, 1, \circ)$, has been chosen and subsequent references to M, 1 and $\circ$ will refer to it.

Our key notion will be the idea of a set, A, equipped with information telling you how to apply a substitution $i \in M$ to an element $a \in A$ to get another element $ia \in A$.

Definition 17.15 (M-set)**.** *Given a monoid M, an M-set is a pair (A, μ), where A is a set and $\mu : M \times A \to A$ a function such that:*

1. $\mu(1, a) = a$ *for all* $a \in A$.
2. $\mu(i, \mu(j, a)) = \mu(i \circ j, a)$ *for all* $i, j \in M$ *and* $a \in A$.

We shall often follow a convention of simply writing ia or $i(a)$ instead of $\mu(i, a)$, and ij instead of $i \circ j$. Note that ija disambiguates to $(i \circ j)a$ and $i(j(a))$, but due to the second clause of Definition 17.15 it does not matter. As with monoids, we will often refer to an M-set by the set component A and leave μ implicit from context.

Here are some examples:

Example 17.10. *If M is the set of concrete substitutions of a language $\mathcal{L}(\Sigma)$, then (A, μ) is an M-set where*

- *A, the set of closed terms of this language of type σ.*
- *$\mu(i, N)$ is the result of replacing all of the constants c appearing in the term N with the term $i(c)$.*[5]

Since i preserves type, $\mu(i, N) \in \mathcal{L}^\sigma(\Sigma)$ and is closed whenever $N \in \mathcal{L}^\sigma(\Sigma)$ and is closed.

The set of substitutions M can itself be thought of as an M-set.

Example 17.11. *$(M, \circ)$ is an M-set with the action $\circ : M \times M \to M$.*

Bolzano defined a sentence to be logically true iff all of its substitution instances are true, and logically consistent iff some substitution instance is true. This parallels the possible worlds analysis of necessity and possibility, with substitutions playing the role of worlds. Someone interested in a *propositional* notion of logical necessity/possibility might therefore seek a similar analysis in terms of metaphysical substitutions. The following M-set could be used to model the propositions along such lines; here the propositions are identified with sets of metaphysical substitutions instead of sets of possible worlds.

Example 17.12. *For any monoid M, $(P(M), \delta)$ is an M-set with the action of "division"*

- $P(M) = \{p \mid p \subseteq M\}$, *the set of subsets of* M.
- $\delta(i,p) = \{j \mid j \circ i \in p\}$.

The reader may wonder why the action of division is a natural notion in this context. In the possible worlds formalism, a proposition (set of worlds) is said to be true at a world if that world belongs to it, and in the substitutional setting a proposition is true under it iff the result of applying the substitution to the proposition is true. The next exercise shows that if these two notions coincide then the result of applying a substitution to a set of substitutions must be the result of dividing the proposition by that substitution. Recall that in a model of higher-order logic truth is modeled by a valuation, a function $v : A^t \to \{0,1\}$ from the propositional type to truth values, $v : A^t \to \{0,1\}$.

Exercise 17.11. *Let $(P(M), \mu)$ be an M-set where $P(M)$ is the powerset of the substitutions, and μ is some arbitrary action of M on $P(M)$. Suppose, moreover, that $v : P(M) \to \{0,1\}$ is subject to the constraint that a proposition $p \subseteq M$ contains the substitution i (i.e. p is 'true at' i) if and only if $v(\mu(i,p)) = 1$ (i.e. the result of applying i to p is true). You will show that $\mu = \delta$, the action of division from Example 17.12.*

a. Show that $v(p) = 1$ iff $1 \in p$ (the true propositions are those containing 1).

b. Show that $1 \in \mu(i,p)$ iff $i \in p$.

c. Show that $j \in \mu(i,p)$ iff $j \circ i \in p$.

d. Conclude that $j \in \mu(i,p)$ iff $j \in \delta(i,p)$ for all $j \in M$, and thus that $\mu = \delta$.

As with modalized sets, any two M-sets have a product.

Example 17.13 (Product of M-sets)**.** *For any two M-sets (A, μ_A) and (B, μ_B) their product is $(A \times B, \mu_{A\times B})$ where*

- $\mu_{A\times B}(i,(a,b)) = (\mu_A(i,a), \mu_B(i,b))$.

Exercise 17.12. *Prove that the product of two M-sets is an M-set.*

It will also be convenient to have the notion of a structure-preserving map between M-sets. This will consist of a function between from one set to another to the other that preserves commutes with the application of substitutions:

Definition 17.16. *Given two M-sets, (A, μ) and (B, ν) a function $f : A \to B$ is a homomorphism of M-sets iff for any $i \in M, a \in A$, $f(\mu(i,a)) = \nu(i, f(a))$.*

An isomorphism is a homomorphism with an inverse, which compose, in either order, to the identity homomorphism of the relevant M-set.

With these concepts in place, we can supplement the notion of an applicative structure with the extra information about how substitutions act on entities. As before, we will assign to each type σ not a set, but an M-set A^σ. Moreover, we need a notion of application $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$ (where $A^{\sigma\to\tau} \times A^\sigma$ denotes a product of M-sets). To figure out what constraints we must impose on App we may consult our linguistic intuitions. If I apply a substitution to an application term (Fa) (where F and a may be complex terms) I get a term $i(Fa)$. This should be the same as first applying i to F and i to a and then applying the results together: $((iF)(ia))$. Thus we should require that for every substutition i, $iApp^{\sigma\tau}(d,a) =$

$\mathrm{App}^{\sigma\tau}(id, ia)$. The reader should notice that this is exactly the condition for $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^{\sigma} \to A^{\tau}$ to be a homomorphism from the M-set $A^{\sigma\to\tau} \times A^{\sigma}$ to A^{τ}.

Definition 17.17 (Substitution Structure)**.** *A Substitution Structure* $(A^{\cdot}, \mathrm{App}^{\cdot})$ *consists of a family* A^{σ} *for each type* σ*, and a family* $\mathrm{App}^{\sigma\tau}$ *for each pair of type* σ *and* τ *such that:*

- A^{σ} *is an M-set with action* μ_{σ}.
- $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^{\sigma} \to A^{\tau}$ *is a homomorphism of M-sets, i.e.:*

  – *For all* $d \in A^{\sigma\to\tau}, a \in A^{\sigma}$, $\mu_{\tau}(i, \mathrm{App}^{\sigma\tau}(d, a)) = \mathrm{App}^{\sigma\tau}(\mu_{\sigma\to\tau}(i, d), \mu_{\sigma}(i, a))$.

A substitution structure is:

- Quasi-functional *iff for every* σ *and* τ *and* $f, d \in A^{\sigma\to\tau}$ *if* $\mathrm{App}(if, a) = \mathrm{App}(id, a)$ *for every* $a \in A^{\sigma}$ *and* $i \in M$*, then* $f = d$.
- Quasi-full *iff for every* σ *and* τ*, and homomorphism* $h : M \times A^{\sigma} \to A^{\tau}$ *there is an element* $d \in A^{\sigma\to\tau}$ *such that for all substitutions* $i \in M$ *and* $a \in A^{\sigma}$ $\mathrm{App}^{\sigma\tau}(id, a) = h(i, a)$.
- Has combinators *iff for every* σ, τ *and* ρ *there are elements* $k^{\sigma\tau} \in A^{\sigma\to\tau\to\sigma}$ *and* $s^{\sigma\tau\rho} \in A^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$ *such that*

  – *For all* $i \in M$, $x \in A^{\sigma}, y \in A^{\tau}$ $(ik^{\sigma\tau})xy = x$.
  – *For all* $i \in M$, $x \in A^{\sigma\to\tau\to\rho}$, $y \in A^{\sigma\to\tau}$ *and* $z \in A^{\sigma}$, $(is^{\sigma\tau\rho})xyz = (xz)(yz)$.

Quasi-functionality and quasi-fullness again stand for weakenings of the corresponding notions for applicative structures. Their role will become much clearer once we have introduced the notion of a concrete substitution structure, analogous to the notion of a concrete quasi-functional modalized applicative structure.

Clearly by throwing away the mathematical structure concerning substitutions from a substitution structure you get an ordinary applicative structure:

Definition 17.18. *Every substitution structure* $(A^{\cdot}, \mathrm{App}^{\cdot})$ *determines an ordinary applicative structure whose type* σ *domain is the underlying set of the M-set* A^{σ} *and whose application function is just the function* $\mathrm{App}^{\cdot}$.

However, the extra structure can be useful in determining a suitable domain $A^{\sigma\to\tau}$ of $\sigma \to \tau$ operations from the domains A^{σ} and A^{τ}. The mathematical structure telling us how substitutions behave on A^{σ} and A^{τ} determine what sort of thing could act as a $\sigma \to \tau$ operations without violating the condition that substitutions should commute with the notion of application (i.e. $iApp(d, a) = \mathrm{App}(id, ia)$).

Given two sets A and B, we have another set $A \to B$ of all functions between them. In the last section, we introduced an analogous operation between pairs of modalized sets, and now we should do the same for M-sets.

Definition 17.19 (Exponential of M-sets)**.** *If A and B are M-sets, the exponential of A and B, written* $A \Rightarrow B$ *is defined as follows:*

- *The underlying set is* $A \Rightarrow B := \{f : M \times A \to B \mid f(i \circ j, ix) = i(f(j, x))$ *for all* $x \in A$ *and* $i, j \in M\}$.
- *The action on this set is:* $if := (j, x) \mapsto f(j \circ i, x)$ *for each* $f : M \times A \to B$.

The condition for f to belong to $A \Rightarrow B$ is just that f be a homomorphism from $M \times A$ to B, where we consider M as an M-set itself with the action $\mu = \circ : M \times M \to M$ (as in Example 17.11).

To get an intuition for our definition of the exponential, we should consult the intended application in which A and B represent sets of structured entities, and $A \Rightarrow B$ represents structured operations taking A entities to B entities. Suppose, then, that $f : M \times A \to B$ belongs to $A \Rightarrow B$ and interprets some structured property F, which possibly has constituents. $f(i, a)$ may then be thought of as the result of taking an i substitution of F and applying it to a: take all of the constituents of F and substitute them with their i-counterparts in F, then apply the result to a. By this reasoning, then, we can see that $f(1, a)$ corresponds to Fa. More generally, if $f(j, a)$ corresponds to the result of performing the j substitution to F and applying the result to the argument, $i(f(j, a))$ should be the same as applying the $i \circ j$ substitution to F and applying the result to ia (since the application of a substitution i should commute over application). Thus $i(f(j, a))$ should be identical to $f(i \circ j, ia)$. And this is just the condition that a function $f : M \times A \to B$ must satisfy in order to belong to $A \Rightarrow B$.

The action of M on $A \Rightarrow B$ tells us what the i-counterpart of each function $f : M \times A \to B$ is. As before we write if for the action of i on f. If f corresponds to a property F then if intuitively corresponds to the result of replacing all the constituents in F with their i-counterparts. Recall that $f(j, a)$, on the above interpretation, is the result of substituting the constituents of F with their j-counterparts and applying the result to a. So by that reasoning, $(if)(j, a)$ is the result of substituting the j-counterparts in [the result of substituting for i-counterparts in F], and then applying to a. This is, of course, exactly the same as $f(j \circ i, a)$; so our definition of if similarly makes sense.

The analogue of a subset for an M-set is a *subact*: a subset of an M-set that is closed under substitutions.

Definition 17.20 (Subact)**.** *Let A and B be M-sets. A is a subact of B, written $A \leq B$, iff*

- $A \subseteq B$.
- *For every $a \in A$ and $i \in M$, $ia \in A$.*

It is easily seen that a subact of any M-set is also an M-set, with the action obtained by restriction.

Definition 17.21 (Concrete Substitution Structure)**.** *A* concrete *quasi-functional substitution structure (or* concrete substitution structure*) is a substitution structure such that*

- $A^{\sigma\to\tau} \leq A^\sigma \Rightarrow A^\tau$.
- $\mathrm{App}^{\sigma\tau}(f, a) = f(1, a)$.

A standard substitution structure is a concrete substitution structure such that

- $A^{\sigma\to\tau} = A^\sigma \Rightarrow A^\tau$.

for every pair of type σ and τ.

Exercise 17.13. *To show that a concrete substitution structure really is a substitution structure we must show that $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$ is a homomorphism of M-sets. That is, it must be shown that $i\mathrm{App}^{\sigma\tau}(f, a) = \mathrm{App}^{\sigma\tau}(if, ia)$. Show this.*

Exercise 17.14. *Show that every concrete quasi-functional substitution structure is quasi-functional: for $f, g \in A^{\sigma\to\tau}$, if $\mathrm{App}(if, a) = \mathrm{App}(ig, a)$ for every $a \in A^\sigma$, $f = g$.*

Exercise 17.15. *Suppose that $(A^\cdot, \mathrm{App}^\cdot)$ is a standard substitution structure.*

a. *Show that the function $k = (i,a) \mapsto (j,b) \mapsto ja$ is a member of $A^{\sigma\to\tau\to\sigma}$.*
b. *Show that the function $s = (k,f) \mapsto (i,g) \mapsto (j,a) \mapsto f(j \circ i, a)(1, g(j,a))$ is in $A^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$.*
c. *Prove that standard substitution structures have combinators.*

Observe that in a concrete substitution structure $A^{\sigma\to\tau}$ is not a set of functions from A to B, but rather a set of functions from $M \times A$ to B. This makes it easy to see how concrete substitution structures can fail to be functional, since it could be that $f(1,a) = g(1,a)$ for every $a \in A^\sigma$ but $f(i,a) \neq g(i,a)$ for some $i \neq 1$.

Exercise 17.16. *Suppose that $(A^\cdot, \mathrm{App}^\cdot)$ is a substitution structure, and that the action of each substitution is surjective at each type: $\mu_\sigma(i,\cdot) : A^\sigma \to A^\sigma$ is surjective for each σ. Show that the applicative structure determined by this substitution structure is functional.*

Exercise 17.17. *Let $M = \{0,1\}^{\{0,1\}}$ be the monoid of functions on $\{0,1\}$ and let $A^e = A^t = \{0,1\}$ with the obvious action of M on $\{0,1\}$ given by function application. Show that the standard substitution structure generated by A^e and A^t does not have a functional underlying applicative structure.*

Exercise 17.18. *$(G, \circ, 1)$ is a group iff $(G, \circ, 1)$ is a monoid and for each $i \in G$ there is an element $i^{-1} \in G$ such that $i \circ i^{-1} = i^{-1} \circ i = 1$. Suppose $(A^\cdot, \mathrm{App}^\cdot)$ is a substitution structure and the substitutions form a group.*

a. *Show that the action of i, $\mu(i,\cdot)$ on A^σ is a permutation.*
b. *Show that if the applicative structure underlying $(A^\cdot, \mathrm{App}^\cdot)$ is functional, then the action on functional types is given by 'conjugation': $\mu_{\sigma\to\tau}(i,f) = a \mapsto \mu_\tau(i, \mathrm{App}(f, \mu_\sigma(i^{-1}, a)))$ (adopting previous conventions this amounts to the more readable condition that $if = a \mapsto i(f(i^{-1}a))$).*
c. *Show that $(A^\cdot, \mathrm{App}^\cdot)$ is quasi-functional iff its underlying applicative structure is functional.*
d. *Show that if $(A^\cdot, \mathrm{App}^\cdot)$ is a standard substitution structure, and the substitutions form a group, then its underlying applicative structure is isomorphic to the full Henkin structure.*

Just as we have previously shown that every functional applicative structure is isomorphic to a Henkin structure, and every quasi-functional modalized applicative structure is isomorphic to a concrete modalized applicative structure, we may also show that quasi-functional substitution structures are isomorphic to concrete quasi-functional substitution structures.

Proposition 17.4. *Every quasi-functional substitution structure $(A^\cdot, \mathrm{App}^\cdot_A)$ is isomorphic to a concrete structure $(B^\cdot, \mathrm{App}^\cdot_B)$: there exists a type indexed family of isomorphisms of M-sets $h^\sigma : A^\sigma \to B^\sigma$ such that $h^\tau(App_A(d,a)) = App_B(h^{\sigma\to\tau}(d), h^\sigma(a))$ for every $a \in A^\sigma$ $d \in A^{\sigma\to\tau}$.*

Proof. See Theorem 15 of Bacon (2019b). □

In a quasi-functional substitution structures with combinators provide us with another rich class of examples of models of the full λ-language that do not correspond to functional applicative structures. In a non-functional applicative structure, the interpretation of a λ-term is not uniquely determined by the applicative behaviour of the entity it denotes. The

extra structure provided by a substitution structure fills this gap: we require that the interpretation of a λ-term, d, not only have a particular applicative behaviour, but also that its applicative behaviour under each substitution—i.e. the applicative behaviour of *id* for each $i \in M$—be a certain way. Since in a quasi-functional applicative structure an element of a functional type is uniquely determined by the applicative behaviour of its substitutions this uniquely determines the interpretation of a λ-term.

Definition 17.22. *An interpretation of a signature Σ in an M-set structure $(A^{\cdot}, \mathrm{App}^{\cdot}, [\![\cdot]\!])$ is involves assigning each element of Σ^{σ} an element of A^{σ}:*

- *For each $c \in \Sigma^{\sigma}$, $[\![c]\!] \in A^{\sigma}$.*

The following definition shows how one can interpret a λ-language in a concrete substitution structure (although by Proposition 17.4 we can transfer this to any quasi-functional substitution structure). Notions from Chapter 15, such as that of a variable assignment, transfer straightforwardly to substitution structures.

Definition 17.23 (Substitution Interpretations)**.** *Consider the full λ-language over signature Σ, $\mathcal{L}(\Sigma)$. A substitution interpretation is an interpretation of Σ, $(A^{\cdot}, \mathrm{App}^{\cdot}, [\![\cdot]\!])$ where $(A^{\cdot}, \mathrm{App}^{\cdot})$ is a concrete substitution structure with combinators. In this case, we can introduce a class of interpretation functions $[\![\cdot]\!]^g_i$, that assigns a value $[\![M]\!]^g_i \in A^{\sigma}$ whenever $M : \sigma$, g is a variable assignment and $i \in M$ is a substitution:*

$[\![x]\!]^g_i = g(x)$.
$[\![c]\!]^g_i = i[\![c]\!]$ *when* $c \in \Sigma$.
$[\![(MN)]\!]^g_i = \mathrm{App}^{\sigma\tau}([\![M]\!]^g_i, [\![N]\!]^g_i) = [\![M]\!]^g_i(1, [\![N]\!]^g)$ *when* $M : \sigma \to \tau$, $N : \sigma$ *when* $M : \sigma \to \tau, N : \sigma$.
$[\![\lambda x.M]\!]^g_i = (j, a) \mapsto [\![M]\!]^{(j \circ g)[a/x]}_{j \circ i}$ *when* $x : \sigma, M : \tau$. *($[\![\lambda x.M]\!]^g_i$ is undefined if this function is not in the structure.)*

The reader will notice that we have not defined a single way to extend the interpretation function to complex terms but rather a class of extensions $[\![\cdot]\!]_i$ for each substitution $i \in M$. These functions differ from each other concerning the interpretations of the constants: roughly $[\![\cdot]\!]_1$ is the extension to complex terms of the original interpretation function, and $[\![\cdot]\!]_i$ is the extension to complex terms of the interpretation function obtained by applying i to the original interpretations of the constants, $i[\![c]\!]$. The need for these variant interpretations is only apparent in the clause for λ, and follows from the informal interpretation we gave for a function $[\![\lambda x.M]\!]^g : M \times A^{\sigma} \to A^{\tau}$ in a concrete substitution structure. $[\![\lambda x.M]\!]^g(i, a)$ is supposed to be understood informally as the result of applying the substitution i to the structured operation $[\![\lambda x.M]\!]^g$ and then applying it to the argument a. The constituents that can contribute to $[\![\lambda x.M]\!]^g$ are determined by the interpretations of the free variables and the constants appearing in it: thus we shift both the interpretations of the variables by using the variable assignment $i \circ g$, and the interpretations of the constants to obtain a variant interpretation $[\![\lambda x.M]\!]^{i \circ g}_i$ and apply it to a. At this point we can apply the usual idea for λ terms, and replace whatever $(i \circ g)$ maps x to with a in $[\![M]\!]^{i \circ g}_i$, giving us that $[\![\lambda x.M]\!](i, a) = [\![M]\!]^{(i \circ g)[a/x]}_i$ as required.

As always, we need to check that the interpretation of the λ-terms always belong to the structure. This is straightforward if the structure is a quasi-full substitution structure. Since in that case every homomorphism from the product M-set $M \times A^\sigma$ to A^τ is in the structure, we just need to check that the interpretation of the λ-term is indeed such a homomorphism.

Exercise 17.19.

a. Prove by induction that $j[\![M]\!]_i^g = [\![M]\!]_{j\circ i}^{j\circ g}$.

b. Show that $k([\![\lambda x.M]\!]_i^g(j,a)) = [\![\lambda x.M]\!]_i^g(k \circ j, ka)$ *for every* $j,k \in M$, *and conclude that* $[\![\lambda x.M]\!]_i^g$ *belongs to the standard substitution structure.*

More generally, we have that the interpretation function is well-defined in any concrete substitution structure with combinators. We state the relevant theorem without proof.

Theorem 17.2. *If A is a substitution structure, then for every term M of $\mathcal{L}^\tau(\Sigma)$ any variable x of type σ, assignment g, and substitution i the function $(a,j) \mapsto [\![M]\!]_{j\circ i}^{(j\circ g)[a/x]}$ is well-defined and belongs to $A^{\sigma\to\tau}$.*

17.3 Applications of substitution structures

Substitution structures provide us with the tools to model the notions we introduced in Section 13.1 for theorizing about propositions, properties, relations and so on in a more structured framework.

There we introduced the notion of some entities being *metaphysically definable* from some others, where this is understood to be a metaphysical analogue of the linguistic notion of definability. In the linguistic case, a closed term M of a typed language is *definable* from some constants $c_1, \ldots, c_n$ iff the constants appearing in M are among (but need not exhaust) those constants. We can explain this in a substitutional way in a couple of equivalent ways. The first is that any substitution, i, of the language that fixes the constants $c_1, \ldots, c_n$ (i.e. maps them to themselves) must also fix M.[6] The second is that any pair of substitutions of the language i,j that agree on $c_1, \ldots, c_n$ must agree on M. That is, if $ic_k = jc_k$ for $k = 1, \ldots, n$ then $iM = jM$.[7] This inspires the following definition:

Definition 17.24 (Metaphysical Definability)**.** *Let $(A^\cdot, \mathrm{App}^\cdot)$ be a substitution structure. If $X \subseteq \bigcup_\sigma A^\sigma$ and $a \in \bigcup_\sigma A^\sigma$ are respectively a set of elements and an element of the substitution structure we say that a is* metaphysically definable *from X iff*

For any substitution $i \in M$, if $ib = b$ for every $b \in X$ then $ia = a$.

The variant (slightly stronger) definition of metaphysical definability may be similarly cashed out as follows:

For any pair of substitution $i,j \in M$, if $ib = jb$ for every $b \in X$, then $ia = ja$.

Comprehension Check 17.1. *Observe that the second definition of metaphysical definability implies the first by setting $j = 1$.*

In previous chapters, we observed that combinators appear to express entities that do not have constituents. For instance, given β $(\lambda p.p)A$ and A are always identical and so have the same constituents, suggesting that $\lambda p.p$ does not contribute any constituents to a proposition in which occurs. We suggested the label *pure* for constituentless entities, and we have

explored general λ-languages that allow us to theorize without postulating such entities. We are now in a position to offer a model-theoretic analysis of a pure entity: if P is a combinator then any substitution of the constants will leave P alone, since P does not contain any constants. This suggests the following definition:

Definition 17.25 (Purity)**.** *Let* $(A^{\cdot}, \mathrm{App}^{\cdot})$ *be a substitution structure. An element* $a \in \bigcup_{\sigma} A^{\sigma}$ *of the structure is* pure *iff*

For every $i \in M$, $ia = a$.

Exercise 17.20. *Show that an element of a substitution structure is pure if and only if it is metaphysically definable from the empty set.*

Let me end the discussion of metaphysical definability by posing an (open) problem concerning the axiomatizability of the notion.

Problem 17.1. *Consider language* $\mathcal{L}(\Sigma)$ *whose signature contains constants* $\triangleright_{\bar{\sigma}\tau} : \bar{\sigma} \to \tau \to t$ *for every sequence of type* $\bar{\sigma}$ *and type* τ *representing metaphysical definability (see Section 13.1). A substitutional model of this language is a general higher-order model* $(A, [\![\cdot]\!], v)$ *where* $A = (A^{\cdot}, \mathrm{App}^{\cdot})$ *is a substitution structure and*

$v([\![\triangleright_{\bar{\sigma}\tau}]\!]a_1 \ldots a_n b) = 1$ *if and only if, for every substitutions* $i, j \in M$ *such that* $ia_m = ja_m$ *for* $m = 1, \ldots, n$, $ib = jb$.

Let $\mathbf{T}$ *be the higher-order theory of these models. Find a recursive axiomatization of* $\mathbf{T}$*, or show that it cannot be done.*

We can also use the present substitutional framework to shed some light on the simple/complex distinction between entities. The constants of a language, unlike the complex expressions, have the special property that they can be altered freely and independently in a complex expression. Take the sentence 'Socrates is wise and mortal'. I can replace the words 'Socrates', 'wise' and 'mortal' with any three expressions of matching grammatical type, say 'Plato', 'bearded', and 'a student of Socrates' a get another sentence 'Plato was bearded and a student of Socrates'. On the other hand, complex expressions cannot be altered independently of other simple expressions. The expressions 'wise' and 'wise and mortal' are not independent: if I wanted to uniformly replace 'wise' with 'bearded', then the complex expressions 'wise and mortal' must go to something of the form 'bearded and . . . '. Similarly while there is a substitution mapping a constant to any expression with the same grammatical type, a complex expression can only be mapped to another complex expression with the same logical form. For instance, 'wise and mortal' can be mapped by a substitution to 'bearded and a student of Socrates' (by replacing 'wise' with 'bearded' and 'mortal' with 'a student of Socrates') but it cannot be mapped by any substitution to 'bearded' since logical structure must be preserved.

So we have the following pair of facts. The constants are independent: Any function from a set of constants to closed terms which preserves type extends to a substitution of the language. Complex terms by contrast are not independent; indeed since substitutions preserve logical form, any function mapping a complex term to another term with a different logical form cannot be extended to a substitution. Secondly, the set of all constants are complete: any type preserving function from a set containing all the constants to closed terms extends

to at most one substitution. This is because once you have said where each constant must go, you have no choice about the behaviour of the substitution on a complex term.

Definition 17.26. *Let* $(A^\sigma, \mathrm{App}^\tau)$ *be a substitution structure. A type-indexed collection of subsets* $S^\sigma \subseteq A^\sigma$*, thought of as a candidate for being the* metaphysically simple *entities, or the* fundamental *entities, is said to satisfy* fundamental independence *and* fundamental completeness *respectively, if the following conditions obtain:*

Fundamental Independence *Every type-indexed collection of functions* $f^\sigma : S^\sigma \to A^\sigma$ *extends to at least one substitution* $i \in M$*.*

Fundamental Completeness *Every type-indexed collection of functions* $f^\sigma : S^\sigma \to A^\sigma$ *extends to at most one substitution* $i \in M$*.*

A type indexed collection of functions f extends to a substitution i iff, for every type σ *and* $a \in A^\sigma$*,* $f^\sigma(a) = ia$*.*

17.4 Abstract operation spaces

The notion of an applicative structure, modalized applicative structure and substitution structure, and the hierarchy of definitions associated with them, have much in common and serve similar purposes. In this section and the next, we will attempt to formulate an abstract notion of model of the full λ-language that has these particular examples as special cases. We will begin in this section by examining the notion of an exponential of two sets: the set of functions with domain A and codomain B, which we will denote $A \to B$. It is 'not too big', in the sense that there are no two elements of $A \to B$ with the same applicative behaviour, and it is 'not too small' in the sense that every possible applicative behaviour is had by some element of $A \to B$. We'll provide an analysis of these two constraints, and show how the notion of an exponential of two modalized sets or M-sets also satisfy them.

Recall that any two sets, A and B, determine a third set, $A \to B$, consisting of the set of all functions from A to B. It comes with an operation of function application, $fapp : (A \to B) \times A \to B$, itself a function, defined by:

$$fapp(f, a) = f(a)$$

Of course, the introduction of the function *fapp* seems redundant since the elements of $A \to B$ are functions, and thus already contain the information needed to apply them to their arguments. However, going forwards we will not assume that the set of elements of a function space are actually functions, and instead take notions of application, like *fapp*, as basic, allowing us to take a more abstract approach to function spaces. This is similar to the approach we took when we introduced applicative structures: each functional type $A^{\sigma\to\tau}$ came with a primitive notion of application $\mathrm{App}^{\sigma\tau} : A^{\sigma\to\tau} \times A^\sigma \to A^\tau$, even when $A^{\sigma\to\tau}$ happened to be a set of functions.

Definition 17.27 (Abstract operation space)**.** *An abstract operation space between two sets A and B,* (F, app)*, consists of*

1. *A set, F, representing candidate operations to be thought of as mapping elements of A to elements of B.*
2. *An operation* $app : F \times A \to B$*.*

Example 17.14. *The canonical example of an abstract operation space between A and B is $(A \to B, fapp)$, where $fapp = (f, a) \mapsto f(a)$.*

Example 17.15. $(\mathbb{N}, +)$ *can be thought of as an abstract operation space from* $\mathbb{N}$ *to* $\mathbb{N}$ *with the result of applying n to m being $n + m$.*

Example 17.16. *For any applicative structure A and pair of types σ and τ, $(A^{\sigma\to\tau}, \mathrm{App}^{\sigma\tau})$ is an abstract operation space. So all of the applicative structures we considered in Chapter 14 have analogues here.*

Following analogous definitions involving applicative structures, some abstract operation spaces will be functional in the sense that if $d, d' \in F$ and $app(d, a) = app(d', a)$ for every $a \in A$, then $d = d'$. And some abstract operation spaces will be *full*, in the sense that for any function $f : A \to B$, there is an element d of F such that $app(d, a) = f(a)$ for every $a \in A$. And when an abstract operation space has both properties it intuitively ought to be isomorphic to the concrete function space $(A \to B, fapp)$.

In order to state this precisely, we need to say what it means for two abstract operation spaces to be isomorphic. And once we have done that, it would be nice to find a more intrinsic characterization of a full and functional abstract operation space that characterize them up to a relevant notion of isomorphism, and doesn't do so simply by stating they are isomorphic to the function space. This will be key to generalizing the notion of function space to mathematical objects with more structure than a set.

Let's begin by saying what it means for two abstract operation spaces between sets A and B to be isomorphic.

Definition 17.28. *A homomorphism between two abstract operation spaces between A and B, (F, app) and (F', app') is a function $h : F \to F'$ that maps an element d of F to an element $h(d)$ of F' with the same applicative behaviour as d. d and $h(d)$ have the same applicative behaviour when:*

$app(d, a) = app'(h(d), a)$ *for every* $a \in A$.

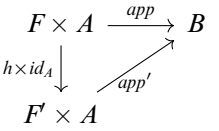
A homomorphism $h : F \to F'$ is an isomorphism iff there exists a homomorphism $g : F' \to F$ such that $g \circ h = 1_F$ and $h \circ g = 1_{F'}$ where 1_F and $1_{F'}$ are the identity functions on F and F' respectively.

A homomorphism thus maps an element $d \in F$ to an element $h(d) \in F'$ that has the same applicative behaviour as d, where the applicative behaviour relative to app is the function $a \mapsto app(d, a)$ (recall the similar notion for applicative structures). The reader may check, if they wish, that an isomorphism is simply a bijective homomorphism.

Given two functions $f : A \to C$ and $g : B \to D$, it is sometimes convenient two write $(f \times g) : A \times B \to C \times D$ for the function defined by $f \times g(a, b) = (fa, gb)$. Then $h : F \to F'$ is a homomorphism of abstract operation spaces if the following identity of functions obtains

$$app = app' \circ (h \times 1_A)$$

where $1_A : A \to A$ is the identity function. Pictorially, this means that following the arrows in the diagram below will always lead to the same thing.



We'll begin by examining the notion of a functional abstract operation space. Suppose that F is not functional in the usual sense that there are distinct $e, e' \in F$ with the same applicative behaviour: $app'(e, a) = app'(e', a)$ for every $a \in A$. Now suppose we have an element d from another abstract operation space F' with the same applicative behaviour as e and e': for every $a \in A$, $app(d, a) = app'(e, a) = app'(e', a)$. Since all a homomorphism of abstract operation spaces has to do is preserve applicative behaviour, it follows that a homomorphism from F' to F, assuming there are any, is free to send d to e or e'. By contrast, suppose F is functional, in the sense that there is at most one element of F with any given applicative behaviour, and d is an element of some other abstract operation space F'. If there is any element of F that has the same applicative behaviour as d, there is only one such element, and that is the only element of F that d could be mapped to by a homomorphism. So there can be at most one homomorphism (but possibly none) into a functional abstract operation space. We will make this our official definition:

Definition 17.29 (Functional abstract operation space)**.** *An abstract operation space for A and B, (F, app), is functional iff there is at most one homomorphism from any other abstract operation space for A and B into (F, app). I.e. iff*

- *For any homomorphisms $h, h' : (F', app') \to (F, app)$, $h = h'$.*

Exercise 17.21. *Show that (F, app) is functional if and only if, for any $d, d' \in F$, if $app(d, a) = app(d', a)$ for every $a \in A$, then $d = d'$.*

We may similarly give the notion of fullness an abstract analysis in terms of homomorphisms. Suppose that (F, app) is an abstract operation space for A and B which is not full, in the sense that there is a function from A to B which is not the applicative behaviour of any element of F (for some $f : A \to B$ no $d \in F$ is such that $app(d, a) = f(a)$ for every $a \in A$). Then there cannot be a homomorphism into (F, app) from any abstract operation space (F', app') that *does* have an element d with that applicative behaviour. For a homomorphism has to map d to something with the same applicative behaviour as d. On the other hand, if (F, app) does have an element for every applicative behaviour, there is at least one homomorphism from any other abstract operation space into (F, app).

Definition 17.30 (Full abstract operation space)**.** *An abstract operation space for A and B, (F, app), is full iff there is at least one homomorphism from any other abstract operation space for A and B into (F, app). I.e. iff*

- *For any abstract operation space (F', app') for A and B there exists a homomorphism $h' : (F', app') \to (F, app)$.*

Exercise 17.22. *Show that an abstract operation space (F, app) for A and B is full if and only if, for any function $f : A \to B$, there is an element $d' \in F$, such that $app(d, a) = f(a)$ for every $a \in A$, then $d = d'$.*

Definition 17.31 (Exponential object)**.** *(F, app) is an exponential object for A and B iff it is full and functional. That is to say*

- *For any abstract operation space* (F', app') *for* A *and* B *there exists a* unique *homomorphism* $h : (F', app') \to (F, app)$.

It follows straightforwardly from the previous two exercises that $(A \to B, fapp)$ is full and functional, and so is an exponential object for A and B. The following exercise substantiates our previous claim to have an abstract characterization of $(A \to B, fapp)$ (up to isomorphism) that does not look at the intrinsic features of the elements of $A \to B$.

Exercise 17.23. *Show that if* (F, app) *and* (F', app') *are full and functional they are isomorphic. You should only appeal to the official definitions of fullness and functionality and the definition of an isomorphism (and you should not use the previous two exercises).*

Conclude that all full and functional abstract operation spaces for A *and* B *are isomorphic to* $(A \to B, fapp)$.

To summarize what we have done in this section, let us present an alternative (but equivalent) definition of an applicative structure using our new functional and full abstract operation space.

Definition 17.32 (Applicative structure)**.** *An applicative structure* A *consists of a pair of sets* A^e *and* A^t*, and for each pair of types an abstract operation space* $(A^{\sigma\to\tau}, \mathrm{App}^{\sigma\tau})$.

- A *is functional iff the abstract operation space* $(A^{\sigma\to\tau}, \mathrm{App}^{\sigma\tau})$ *is functional (there is at most one homomorphism from any other abstract operation space into it).*
- A *is full iff the abstract operation space* $(A^{\sigma\to\tau}, \mathrm{App}^{\sigma\tau})$ *is full (there is at least one homomorphism from any other abstract operation space into it).*

The reader should convince themselves, using the observations and exercises from this section, that this definition really is equivalent to the original.

The plan, in what remains of this section, is to observe that by systematically replacing the word 'set' with 'modalized set', or 'M-set', and replacing the word 'function' with 'homomorphism of modalized sets' or 'M-set homomorphism' we obtain parallel definitions of abstract operation space, homomorphism of abstract operation space. Moreover we will see (in the exercises) that the parallel notion of 'full' and 'functional' operation spaces obtained this way correspond to the notion of quasi-fullness and quasi-functionality that we applied to modalized sets and M-sets.

We begin with M-sets

Definition 17.33. *A abstract operation space between* M*-sets* A *and* B *is a pair* (F, app) *where*

1. F *is an* M*-set.*
2. $app : F \times A \to B$ *is a homomorphism of* M*-sets from* $F \times A$ *to* B.

Definition 17.34. *A homomorphism between two abstract operation spaces between* M*-sets* A *and* B*,* (F, app) *and* (F', app') *is a homomorphism of* M*-sets* $h : F \to F'$ *such that* $app = app' \circ (h \times 1_A$*. This means that* h *maps an element* d *of* F *to an element* $h(d)$ *of* F' *with the same applicative behaviour as* d*.* d *and* $h(d)$ *have the same applicative behaviour when:*

$app(d, a) = app'(h(d), a)$ *for every* $a \in A$.

A homomorphism of M-set abstract operation spaces $h : F \to F'$ is an isomorphism iff there exists a homomorphism $g : F' \to F$ such that $g \circ h = 1_F$ and $h \circ g = 1_{F'}$ where 1_F and $1_{F'}$ are the identity functions on F and F′ respectively.

Definition 17.35. *An abstract operation space (F, app) between M-sets A and B is:*

- quasi-functional *iff for any other abstract operation space between A and B, (F', app') there is at most one homomorphism of abstract operation spaces from (F', app') to (F, app).*
- quasi-full *iff for any other abstract operation space between A and B, (F', app') there is at at least one homomorphism of abstract operation spaces from (F', app') to (F, app).*
- an exponential *iff for any other abstract operation space between A and B, (F', app') there is exactly one homomorphism of abstract operation spaces from (F', app') to (F, app).*

Exercise 17.24. *Suppose that A and B are M-sets and that (F, app) is an abstract operation space between them.*

a. Suppose that (F, app) is quasi-functional. Show that if $app(id, a) = app(ie, a)$ for every $a \in A$ and $i \in M$, then $d = e$.

b. Suppose that (F, app) is quasi-full. Show that for every function $f \in A \Rightarrow B$, there is an element $d \in F$ such that $f(i, a) = app(id, a)$ for every $i \in M$ and $a \in A$.

The hierarchy of definitions for modalized sets is essentially the same as in Definitions 17.33–17.35. To obtain the notion of an abstract operation space between modalized sets, of a homomorphism, and of quasi-functionality, fullness and exponential (relative to some fixed frame $\mathcal{F}$), simply uniformly replace the words 'M-set' and 'M-set homomorphism' in Definitions 17.33–17.35 with the words 'modalized set' and 'homomorphism of modalized sets'.

The new definitions of quasi-functionality and quasi-fullness are equivalent to our original definitions for modalized sets. You will verify one direction of these implications in the following exercise.

Exercise 17.25. *Suppose that A and B are modalized sets and that (F, app) is an abstract operation space between them.*

a. Suppose that (F, app) is quasi-functional as an operation space. Show that if $app(i_{xy}d, a) = app(i_{xy}e, a)$ for every $a \in A$ and $x \leq y \in W$, then $d = e$.

b. Suppose that (F, app) is quasi-full. Show that for every element $f \in (A \Rightarrow B)_{@}$, there is an element $d \in F$ such that $f(w, a) = app(w, (d, a))$ for every $w \in W$ and $a \in A_w$.

17.5 Categories

As we mentioned previously, category theory will provide us with the means to generalize these ideas to other contexts. A category $\mathcal{C}$ consists of a class of 'objects' (often, but not always, a set endowed with algebraic structure) and a class of 'arrows' or 'morphisms' (often, but not always, mappings between sets preserving algebraic structure). For any two objects A and B, there is a class $Hom_{\mathcal{C}}(A, B)$ of arrows h between A and B, which we write $h : A \to B$ borrowing the notation we used for functions and homomorphisms earlier.

Definition 17.36 (Category)**.** *A category $\mathcal{C}$ consists of a class of objects $Obj(\mathcal{C})$, for any pair of objects $A, B \in Obj(\mathcal{C})$ a class of arrows $Hom_{\mathcal{C}}(A, B)$, an arrow $1_A \in Hom_{\mathcal{C}}(A, A)$ for any*

object A, and a partially defined binary operation $\circ$ *mapping a pair of arrows to an arrow. We write* $h : A \to B$ *to mean that* $h \in Hom_{\mathcal{C}}(A, B)$*, and we refer to A as hs domain and B as hs codomain. Furthermore:*

- $f \circ h$ *is defined iff* $h : A \to B$ *and* $f : B \to C$*, and* $f \circ h : A \to C$*. Moreover, for any arrows* $h : A \to B, f : B \to C, g : C \to D$*:*

  $-\ ((g \circ f) \circ h) = (g \circ (f \circ h))$

- *For every* $f : A \to B$*:*

  $-\ f \circ 1_A = f = 1_B \circ f$

Category theory is a vast subject, which we will not have space to properly survey in this book. We will therefore focus on three or four concrete categories that have direct application in the interpretation of higher-order logic. Our ultimate goal in this chapter will be to find categories of mathematical structures that have, for any pair of objects A and B, another object $A \Rightarrow B$ that plays a role structurally like the full function space $A \to B$ between two sets.

Let's begin, then, by defining one of the most basic examples of a category—the category of sets.

Example 17.17 (The Category of Sets)**.** *The category of sets,* Set*, has as objects sets, and as arrows* $h : A \to B$ *functions with domain A and codomain B.* $f \circ g$ *simply denotes function composition, and* $1_A : A \to A$ *is the identity function.*

It is trivial to check that our definitions of identity and composition conform to the requirements of a category. Of course, the other two examples we have introduced are:

Example 17.18 (The Category of M-sets)**.** *The category of M-sets has as objects M-sets, and as arrows* $h : A \to B$ *M-set homomorphisms.* $f \circ g$ *simply denotes function composition, and* $1_A : A \to A$ *is the identity homomorphism.*

Example 17.19 (The Category of Modalized Sets)**.** *The category of modalized sets has as objects modalized sets, and as arrows homomorphisms* $h : A \to B$ *.* $h \circ g$ *denotes composition of homomorphisms, and* $1_A : A \to A$ *is the identity homomorphism for A.*

Let us return to the category of sets. We have already talked about the function space between two sets, and in the last section, we effectively gave a category-theoretic characterization of the function space. An even more basic operation, which we have also appealed to frequently, is the product $A \times B$ of two sets A and B: the set of all ordered pairs from A and B respectively $\{(a, b) \mid a \in A, b \in B\}$. A similarly abstract approach to products is possible. The concrete product defined above comes equipped with two projection functions:

- $\pi_1 : A \times B \to A$, where $\pi_1(a, b) = a$
- $\pi_2 : A \times B \to B$, where $\pi_2(a, b) = b$

We could then define a generalization of the notion of a product of A and B to simply be a set P equipped with two functions $p_1 : P \to A$ and $p_2 : P \to B$ being the two abstract

projections. Say that an element $q \in P$ represents the pair (a, b) iff $p_1(q) = a$ and $p_2(q) = b$. Like with abstract operation spaces, these general products can over-represent or under-represent the ordered pairs from A and B. In general, there could be two distinct elements $q, q' \in P$ representing the same ordered pair (a, b), in the sense that they both have a as the first projection, and both have b as the second projection: $p_1(q) = p_1(q') = a$ and $p_2(q) = p_2(q') = b$. Or a given ordered pair (a, b) could fail to be represented by any element of P: there is no q such that $p_1(q) = a$ and $p_2(q) = b$. What is special about about the set theoretic product $(A \times B, \pi_1, \pi_2)$ is there is a unique element q of $A \times B$ for any two elements $a \in A$ and $b \in B$, such that $\pi_1(q) = a$ and $\pi_2(q) = b$ (namely $q = (a, b)$). We can give this an abstract treatment in terms of a suitable notion of homomorphism. Intuitively, a homomorphism between two general products must map an element q to an element $h(q)$ that has the same two projections as q does (i.e. $h(q)$ should represent the same ordered pair as q does).

Definition 17.37 (General product)**.** *A general product of A and B is a tuple (P, p_1, p_2) with $p_1 : P \to A$ and $p_2 : P \to B$.*

A homomorphism of general products $h : (P, p_1, p_2) \to (P', p'_1, p'_2)$ is a function $h : P \to P'$ such that $p'_i(h(q)) = p_i(q))$ for $i = 1, 2$ and $q \in P$.

A homomorphism $h : P \to P'$ is an isomorphism iff it has an inverse $g : P' \to P$ such that $g \circ h = 1_P$ and $h \circ g = 1_{P'}$.

Remark 17.1. The notion of a homomorphism $h : (P, p_1, p_2) \to (P', p'_1, p'_2)$ can be restated in the primitives of category theory as an equation of arrows: $p'_i \circ h = p_i$ for $i = 1, 2$.

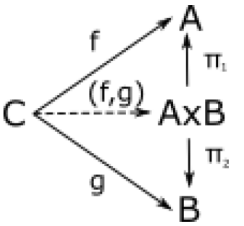
When a general product (P, p_1, p_2) of A and B represent all ordered pairs in A and B, one can always find a homomorphism from any other general product into (P, p_1, p_2): if q is an element of some other general product that represents (a, b) we know q can be mapped to any element of P that represents (a, b). When a general product never over-represents an ordered pair—there is at most one element of P representing the ordered pair (a, b)—then there is at most one element that a homomorphism can map an element of another general product to. Thus iff a general product has exactly one representation of each ordered pair, there is a unique homomorphism into it from any other general product:

Definition 17.38 (Product)**.** *A general product (P, p_1, p_2) A and B is a product of A and B iff there is exactly one homomorphism from any other general product into (P, p_1, p_2).*

Exercise 17.26. *Show that $(A \times B, \pi_1, \pi_2)$ is a product of A and B.*

Exercise 17.27. *Show that any two products of A and B are isomorphic.*

It follows from our definition of a product that given any set C and functions $f : C \to A$ and $g : C \to B$, there is a unique homomorphism of general products (thinking of (C, f, g) as a general product) which we will call $(f, g) : C \to A \times B$ such that the following diagram commutes:



Of course (f, g) may be defined explicitly as the function $c \mapsto (fc, gc)$, but our abstract definition ensures that something behaving like (f, g) exists for an arbitrary product (P, p_1, p_2), even if it is not a set of ordered pairs.

The reader may have observed that our definition of general product, and thus product, makes sense in any category $\mathcal{C}$.

Definition 17.39. *A general product of two objects A and B in a category $\mathcal{C}$ is a tuple (P, p_1, p_2) consisting of an object and a pair of arrows $p_1 : P \to A$ and $p_2 : P \to B$.*

A homomorphism between general products of A and B, (P, p_1, p_2) to (P', p'_1, p'_2), is an arrow $h : P \to P'$ such that $p'_i \circ h = p_i$ for $i = 1, 2$.

A product of A and B is a general product of A and B, (P, p_1, p_2) such that such that there is exactly one homomorphism into it from any other general product.

Exercise 17.28. *Show that the earlier definitions of the product of two modalized sets and M-sets also satisfies our categorical definition. (Observe that the categorical definition does not define the product of two objects uniquely, but only up to an isomorphism of general products.)*

Before we move on, observe that both our definition of an exponential object (full and functional abstract operation space), and our definition of product had a similar form: we defined a notion of homomorphism between abstract functional spaces/general products and defined the exponential object/product to be something such that there is exactly one homomorphism into it from any other object. This suggests a general definition

Definition 17.40 (Terminal and initial object)**.** *A terminal object a category $\mathcal{C}$ is an object T of the category such that for any other object A of $\mathcal{C}$, there is a unique arrow $h : A \to T$.*

Dually, an initial object a category $\mathcal{C}$ is an object I of the category such that for any other object A of $\mathcal{C}$, there is a unique arrow $h : I \to A$.

Exercise 17.29. *Which sets are terminal objects in the category* Set*? Which set is initial? What about the category of M-sets and modalized sets?*[i]

Exercise 17.30. *Show that the class of general products and general product homomorphisms is a category. Explain how a product is a terminal object of this category.*

Exercise 17.31. *Show that the class of abstract operation spaces and their homomorphisms form a category. Explain how an exponential object is a terminal object of this category.*

Now a useful definition:

Definition 17.41 (Cartesian Category)**.** *A category $\mathcal{C}$ is cartesian iff it contains a terminal object any two objects of $\mathcal{C}$ has a product.*[8]

Convention 17.1. *We will follow the convention of writing $(A \times B, \pi_1, \pi_2)$ for the product of two objects since (as we effectively proved in Exercise 17.26) any two products are isomorphic (as general products, and thus as objects of $\mathcal{C}$).*

Whenever C is an object and $f : C \to A$ and $g : C \to B$, we will write $(f, g) : C \to A \times B$ for the unique homomorphism of general products from (C, f, g) to $(A \times B, \pi_1, \pi_2)$.

If $f : A \to C$ and $g : B \to D$, then $f \circ \pi_1 : A \times B \to C$ and $g \circ \pi_2 : A \times B \to D$. We write $(f \times g) : A \times B \to C \times D$ for $(f \circ \pi_1, g \circ \pi_2) : A \times B \to C \times D$.

Exercise 17.32. *Show that in the category* Set*, our convention of using $(f \times g)$ accords with our previous explicit definition $(f \times g) = (a, b) \mapsto (fa, gb)$.*

Now observe that the only distinctively set-theoretic notion we used in our definition of an exponential object in Set was the use of the set-theoretic product in defining the notion

of application, $app : F \times A \to B$ for an abstract operation space. Thus we may define the notion of an exponential object in any cartesian category.

Definition 17.42. *An abstract operation space (F, app) from A to B in a category $\mathcal{C}$ consists of an object F and an arrow $app : F \times A \to B$.*

A homomorphism of abstract operation spaces from A to B, (F, app) and (F', app'), is an arrow $h : F \to F'$ such that that $h \circ app = app' \circ (h \times 1_A)$.

An abstract operation space (F, app) from A to B is full (functional, an exponential object) iff there is at least one (at most one, exactly one) homomorphism of abstract operation spaces from any other abstract operation space to it.

Definition 17.43 (Cartesian Closed Category)**.** *A cartesian category is cartesian closed if every objects A and B have an exponential.*

Convention 17.2. *We will write $A \Rightarrow B$ for an exponential object of A and B (they are the same up to an isomorphism of abstract operation spaces, which will also be an isomorphism of $\mathcal{C}$).*

Example 17.20. *The category of M-sets and the category of modalized sets are both Cartesian closed, as we showed in the previous section that the exponential of M-sets or modalized sets satisfies the categorical definition of an exponential object.*

It is possible to interpret the full λ-language (and indeed, the richer system that also includes product types) in any cartesian closed category in such a way that $\beta\eta$ is validated. The full details of this interpretation take us beyond the scope of this book; see Mitchell (1996), Section 7.2 or Lambek and Scott (1988). For technical reasons, the interpretation assigns arrows from the category to sequents that are derivable in the Curry typing system of Chapter 10.

Theorem 17.3. *Let $\mathcal{L}(\Sigma)$ be a λ-language with product types and a singleton type 1, $\mathcal{C}$ a cartesian closed category and for each type σ let be A^σ be an object of $\mathcal{C}$ such that*

- $A^{\sigma\to\tau} = A^\sigma \Rightarrow A^\tau$
- $A^{\sigma\times\tau} = A^\sigma \times A^\tau$
- $A^1 = T$

Then there exists an interpretation function $[\![\cdot]\!]$ mapping a derivable sequent $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ to an arrow $[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau]\!] : A^{\sigma_1} \times \cdots \times A^{\sigma_n} \to A^\tau$, subject to the constraint that sequents that are related by a permutation of the variables and with $\beta\eta$ equivalent terms on the right get mapped to the same arrow.

17.6 Actions

We'll close the chapter by briefly discussing a final example of a cartesian closed category which unifies the two cartesian closed categories that have been the focus of this chapter.

First we note that the notion of a category is a generalization of a (non-pointed) Kripke frame $(W, \leq)$.

Definition 17.44 (Preorder category)**.** *A* preorder category $\mathcal{F}$ *is a category such that there is at most one arrow between any two objects. For any objects w and v, $|Hom_{\mathcal{F}}(w, v)| \leq 1$.*

Every preorder category determines a Kripke frame $(W, \leq)$ by setting W to be the set of objects of the category, and stipulating that $w \leq v$ iff there exists an arrow $h : w \to v$.

Comprehension Check 17.2. *Check that $\leq$ is transitive and reflexive.*

Conversely, every Kripke frame determines a preorder category: the objects of the category are the worlds and there is a unique arrow between two objects iff the accessibility relation, $\leq$, holds between them.

Comprehension Check 17.3. *Check that the composition and identity laws are guaranteed to hold in this definition.*

A category, however, is more general than a Kripke frame. In a Kripke frame, a world can be accessible to another in at most one 'way'. By contrast, a category allows one world (i.e., object of the category) to access another in multiple different 'ways', represented by different arrows $h, g : w \to v$ with the same domain and codomain. One way in which this extra generality can manifest itself is that once we introduced the analogue of a modalized set over a category, one can have several counterpart functions $i_{wv}, j_{wv} : A_w \to A_v$ for different arrows $i, j : w \to v$.

A category is also a generalization of a monoid.

Definition 17.45 (Monoid category)**.** *A* monoid *category is a category with exactly one object.*

Every monoid category determines a monoid $(M, \circ, 1)$, and vice versa. The elements of M are the arrows from the unique object of the monoid category to itself, $M := Hom(A, A)$, $\circ$ is arrow composition, and 1 is 1_A the identity arrow. Kripke frames and monoids thus correspond to categories that are degenerate with respect to the arrows and the objects respectively.

We next introduce the notion of an *action* on a category $\mathcal{C}$. We'll first introduce it as a generalization of a modalized set, but we can also think of it as a generalization of an M-set. We'll refer to the objects of $\mathcal{C}$ as worlds, and use the letters we've previously used for worlds, w, v, x, y, z, to range over them. We'll use letters like i, j, k to range over arrows between worlds. An action A consists of a domain A_w for each world w, and for each way for w to access v, $i : w \to v$, a counterpart function $i^A_{wv} : A_w \to A_v$. As usual, we will omit the superscript A when no ambiguity is present.

Definition 17.46 (Action)**.** *Suppose that $\mathcal{C}$ is a category with objects W and homomorphisms $Hom(w, v)$ for each $w, v \in W$. An* action*, $(A_\cdot, \cdot^A_\cdot)$ over $\mathcal{C}$ is defined .*

- *A set, A_w, for each world $w \in W$.*
- *For each arrow $i : w \to v$, function, $i_{wv} : A_w \to A_v$ such that:*
  - *$(1_w)_{ww} : A_w \to A_w$ is the identity function*
  - *Whenever $i : x \to y$ and $j : y \to z$, $(j \circ i)_{xz} = j_{yz} \circ i_{xy}$ (where $\circ$ denotes arrow composition on the LHS and function composition on the RHS).*

Note that when $\mathcal{C}$ is a preorder category, our definition is essentially a notationally equivalent to our definition of a modalized set. Note, however, that when $\mathcal{C}$ is a monoid category, the notion of an action over $\mathcal{C}$ collapses into the notion of an M-set. Since there is only one object in a monoid, w, there is only one set A_w, and we can omit the subscript. For each $i : w \to w$, $\mu(i, a)$ can then be defined as $i_{ww}(a)$

Comprehension Check 17.4. *The reader should check that this is an M-set.*

Like modalized sets and M-sets, there is a natural notion of a homomorphism between actions

Definition 17.47 (Homomorphism of actions)**.** *If A and B are actions, a homomorphism $h : A \to B$ is a binary $f : W \times \bigcup_{w \in W} A_w \to \bigcup_{w \in W} B_w$ such that:*

- $f(w, \cdot) : A_w \to B_w$ *whenever for every* $w \in W$.
- $f(y, (i^A_{xy} a)) = i^B_{xy}(f(x, a))$ *whenever* $i : x \to y$ *and* $a \in A_x$.

Observe again that our definition closely mirrors the definition of a homomorphism between modalized sets.

Comprehension Check 17.5. *Convince yourself that when $\mathcal{C}$ is a monoid category, the definition of an action homomorphism is equivalent to the notion of a homomorphism of M-sets.*

The class of actions over a fixed category $\mathcal{C}$ with action homomorphisms forms a category in its own right. In fact, it is a cartesian closed category, so that all of the notions we defined in this chapter (abstract operation space, exponential, and structure) can all be realized in the category of actions. Describing much of this involves repeating definitions we have already encountered in the case of M-sets and modalized sets. We'll end the chapter by simply defining the exponential of two actions, although we will not attempt to prove that it satisfies the definition of an exponential (the details of which can be found in any textbook on category theory).[9]

Definition 17.48 (Exponential)**.** *If A and B are actions over $\mathcal{C}$, their exponential $(A \Rightarrow B)$ is defined as follows:*

- $(A \Rightarrow B)_w$ *is the set of all binary functions f such that*
  - *For each object v, f maps an arrow $i : w \to v$ and an element $a \in A_v$ to an element $f(i, a) \in B_v$. I.e., f determines a function in $Hom(w, v) \times A_v \to B_v$*
  - $f(i \circ j, i^A_{xy} a) = i^B_{yz}(f(j, a))$ *for all $j : x \to y$, $i : y \to z$*
- *For any $i : x \to y$, $(i^{A \Rightarrow B}_{xy} f) = f(j \circ i, a)$ for every $j : y \to z$ and $a \in A_z$.*

Endnotes

1. See Mitchell and Moggi (1991). Modalized sets are related to the modalized domains in Bacon (2018c), and are equivalent as categories.
2. The term 'frame' for a preordered set comes from the Kripke semantics for intuitionistic logic. In modal logic 'frame' often refers to a set equipped with an arbitrary relation.
3. See Cresswell and Hughes (1996). There are several connections here: the theorems of **S4** on every interpretation at every world of a frame if and only if that frame is transitive and reflexive. Moreover, **S4** is complete with respect to the class of transitive reflexive Kripke frames: any non-theorem of **S4** can be invalidated in some transitive reflexive Kripke frame.
4. Russell (1918), p. 19.
5. This description of μ can be made into a precise inductive definition if necessary: $\mu(i, c) = i(c)$ for each constant $c \in \Sigma$, $\mu(i, (MN)) = (\mu(i, M)\mu(i, N))$ and $\mu(i, \lambda x.M) = \lambda x.\mu(i, M)$.
6. See Bacon (2019b), Section 3.2.

7. See Bacon (2019b), Section 3.2. This notion is also found in Goodman (2018b).
8. An equivalent way of saying this is that $\mathcal{C}$ has n-ary product for each n, where an n-ary product of $A_1, \ldots, A_n$ is an object such that there is a unique product homomorphism into it from any general n-ary product, $(f_1 : C \rightarrow A_1, \ldots, f_n : C \rightarrow A_n)$. The terminal object can be thought of as the 0-ary product.
9. Readers interested in exploring the application of actions in the model theory of Classicism should see Bacon and Dorr (forthcoming), Section 3, and Appendices C–E.

Hints for exercises

[i] **Hint:** When determining which set is initial in the category of sets, the reader should pay attention to the official definition of a function in set-theory as a set of ordered pairs.

18

The model theory of classicism

In this chapter, we will investigate the model theory of Classicism. In Section 18.1, we will outline a class of models of Classicism based on the modalized structures of Chapter 17 and show that Classicism is sound (Section 18.2) and complete (Section 18.4 with respect to this class of structures. In between the soundness and completeness results, we discuss the special case of 'standard' models and discuss their incompleteness in general, and their completeness with respect to the propositional modal logic **S4** (Section 18.3). Section 18.5 turns to disjunction and coherence properties of extensions of Classicism introduced in Section 8.3. Section 18.6 introduces a powerful method for establishing that extensions of Classicism are coherent.

In this chapter, the word 'model' will be used exclusively for the modal models of Classicism defined in Section 18.1. Since in Classicism it is possible to define all the logical notions in Λ from this basis from the smaller logical signature Λ^-, taking only $\rightarrow$ and $\forall_\sigma$ as primitive, we shall work exclusively in the smaller signature in this chapter.

While the model theory of this chapter is based on modalized sets, it is possible to develop a model theory for Classicism based on the other structures we encountered in Chapter 17: the substitution structures of Section 17.2 (see Bacon (2019b), Section 5.3 and Bacon and Dorr (forthcoming), Section 3), and the more general notion of an action structure of Section 17.6 that includes both model theories as special cases (Bacon and Dorr (forthcoming), Section 3). These models are not covered here for reasons of space, however many of the independence results mentioned in Section 8 that are not proven in this chapter can be established using these further model techniques; the reader interested in the details can consult the appendices of Bacon and Dorr (forthcoming).[1]

18.1 Modal models of classicism

In this section, we will introduce a model theory for Classicism over certain sorts of modalized structures.

Let's begin by recapping key definitions from Section 17.1. Recall that a pointed Kripke frame consists of a set W equipped with a transitive reflexive relation $\leq$ on W along with a designated world $@ \in W$. Throughout when we refer to a Kripke frame we will always mean a pointed Kripke frame. A modalized set is a collection of sets D_w, indexed by worlds $w \in W$, equipped with a family of counterpart functions i_{wv}, mapping the representative of an individual in D_w to its representative in D_v.

Remark 18.1. The reason we are using pointed Kripke frames, rather than the more traditional notion of a frame $(W, \leq)$ used in the study of normal modal logics, has to do with the

DOI: 10.4324/9781003039181-23

Table 18.1 Definitions of key model elements

$k(y,a)(z,b)$	=	$i_{yz}a$
$s(y,f)(z,g)(w,a)$	=	$f(w,a)(w,g(w,a))$
$\text{if}(w,p)(v,q)$	=	$(W \setminus i_{wv}p) \cup q$
$\text{all}_\sigma(w,f)$	=	$\{v \in W \mid v \in f(v,a) \text{ for every } a \in D_v^\sigma\}$
$\text{eq}_\sigma(w,a)(v,b)$	=	$\{u \mid i_{wu}a = i_{vu}b\}$

fact that we have not built closure under necessitation into our definition of a higher-order logic. Pointed Kripke frames let us model modal logics that are not closed under necessitation. We will call an extension of Classicism that *is* closed under necessitation a *normal* extension.

The models we develop in this chapter are based on concrete modalized applicative structures (over some frame $\mathcal{F} = (W, \leq, @)$) from Definition 17.11. This means that each type σ is associated with a modalized set D^σ: D_w^σ represents the set of entities of that type that exist at w. Moreover, in concrete structures each functional type $D^{\sigma\to\tau}$ is a subdomain of the full exponential space $D^\sigma \Rightarrow D^\tau$. This means that $D_w^{\sigma\to\tau}$ consists of a set of homomorphisms $f : D^\sigma{\uparrow}w \to D^\tau{\uparrow}w$—i.e. a set of homomorphisms between the modalized sets you get from D^σ and D^τ by restricting these modalized sets to the worlds $\geq w$ (and discarding the domains at other worlds). When $v \geq w$, the counterpart at v of a homomorphism f from the domain at w is similarly obtained by restricting f to the worlds $\geq w$ (we called this restriction process 'truncation by w' in either case).

- $D_w^{\sigma\to\tau} \subseteq \{f \mid f : D^\sigma{\uparrow}w \to D^\tau{\uparrow}w \mid f \text{ a homomorphism}\}$.
- When $f \in D_x^{\sigma\to\tau}$ and $z \geq y$, $(i_{xy}f)(z,a) = f(z,a)$ (i.e. $i_{xy}f = f{\uparrow}y$)

(A homomorphism, $f : A \to B$, between modalized sets A and B, is a function that maps a pair (w,a) where $a \in A_w$ to a value in B_w and commutes in the obvious way with the counterpart functions: $i_{wv}f(w,a) = f(v, i_{wv}a)$.)

In the models, we will let D^e be any modalized set whatsoever, however we insist that the propositional domain D^t be a subdomain of the modalized set $B(\mathcal{F})$, from Section 17.1, where $\mathcal{F}$ is the frame of the structure. This means that the propositional domain at the world w contains sets of worlds that are accessible to w. When $v \geq w$, we may shift a set of worlds accessible to w to a set of worlds accessible to v by truncation: i.e. discarding the worlds in the set not accessible to v.

- $(D^t)_x \subseteq P(W{\uparrow}x)$.
- $i_{xy}p = p{\uparrow}y$.

Here, as before, we write $p{\uparrow}w$ for the set $\{v \in p \mid v \geq w\}$ when $p \subseteq W$ and $w \in W$.

Finally, we require that these models contain the elements listed in Table 18.1 that ensure the model contains interpretations of the combinators and logical operations.

We summarize all of this with the following definition. Here we write $A \leq B$ when A_w is contained in B_w for each w, and the counterpart functions for A are the result of restricting the counterpart functions for B (as in Definition 17.10).

Definition 18.1 (Modal model)**.** *A* modal model $\mathbf{M}$ *of a signature* $\Sigma \supseteq \Lambda^{-}$ *is a triple* $(\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$ *where* $\mathcal{F} = (W, \leq, @)$ *is a pointed Kripke frame, and* $D^{\cdot}$ *consists of a type-indexed collection of modalized sets* D^{σ} *and* $[\![\cdot]\!]$ *an interpretation function defined on* Σ *such that:*

1. $D^{t} \leq B(\mathcal{F})$.
2. $D^{\sigma\to\tau} \leq D^{\sigma} \Rightarrow D^{\tau}$.
3. *The combinators, connectives, quantifiers and identity belongs to* D^{σ}_{v} *for every world* v *and* σ *of appropriate type:*
   1. $k \in D^{\sigma\to\tau\to\sigma}_{@}$ *where* $k(y,a)(z,b) := i_{yz}a$ *where* $z \geq y \geq @$, $a \in D^{\sigma}_{y}$, $b \in D^{\tau}_{z}$
   2. $s \in D^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to(\sigma\to\rho)}_{@}$ *where* $s(y,f)(z,g)(w,a) := f(w,a)(w,g(w,a))$ *where these dummy variables range so that* $w \geq z \geq y \geq @$, $f \in (D^{\sigma} \Rightarrow (D^{\tau} \Rightarrow D^{\rho}))_{y}$, $g \in (D^{\sigma} \Rightarrow D^{\tau})_{z}$ *and* $a \in D^{\sigma}_{w}$.
   3. $\mathrm{if} \in D^{t\to t\to t}_{@}$ *where* $\mathrm{if}(w,p)(v,q) := (W \setminus i_{wv}p) \cup q$
   4. $\mathrm{all}_{\sigma} \in D^{(\sigma\to t)\to t}_{@}$ *where* $all_{\sigma}(w,f) = \{v \in W \mid v \in f(v,a) \text{ for every } a \in D^{\sigma}_{v}\}$.
   5. $\mathrm{eq}_{\sigma} \in D^{\sigma\to\sigma\to t}_{@}$ *where* $eq_{\sigma}(w,a)(v,b) = \{u \mid i_{wu}a = i_{vu}b\}$.
4. $[\![c]\!] \in D^{\sigma}_{@}$ *for each* $c \in \Sigma^{\sigma}$, $[\![\forall_{\sigma}]\!] = \mathrm{all}_{\sigma}$, *and* $[\![\to]\!] = \mathrm{if}$.

We write $v_{@} : D^{t} \to \{0,1\}$ *for the function:*

$v_{@}(p) = 1$ *iff* $@ \in p$.

and we say a formula A *is* true *in a model iff* $v_{@}([\![A]\!]^{g}_{@}) = 1$ *for all assignments* g.

Note that in the definition of truth, we used the notation $[\![A]\!]^{g}_{@}$ from Definition 17.13: in a concrete modalized applicative structure (including any modal model) the interpretation of λ-terms can be assigned recursively by assigning to each expression at each world w a denotation $[\![M]\!]^{g}_{w}$ in the domain of that world D^{σ}_{w}.

- $[\![c]\!]^{g}_{w} = i_{@w}[\![c]\!]$ for any logical or non-logical constant $c \in \Sigma$
- $[\![x]\!]^{g}_{w} = g(x)$
- $[\![MN]\!]^{g}_{w} = [\![M]\!]^{g}_{w}(w, [\![N]\!]^{g}_{w})$
- $[\![\lambda x.M]\!]^{g}_{w} = (v,a) \mapsto [\![M]\!]^{(i_{wv}\circ g)[a/x]}$

These denotations at each world are related in the natural way by the counterpart functions: $i_{wv}[\![A]\!]^{g}_{w} = [\![A]\!]^{i_{wv}\circ g}_{v}$ (Lemma 17.1).

Remark 18.2. The definition of a model here closely resembles the model theory used by Leon Henkin (1950) to obtain completeness results for Extensionalism. Henkin's models were based on the applicative structures from Example 14.1: $A^{t} = \{0,1\}$, but as in a Henkin structure, you let $A^{\sigma\to\tau} \subseteq A^{\sigma} \to A^{\tau}$ be some set of functions from A^{σ} to A^{τ}. Because Henkin did not require all functions to belong to $A^{\sigma\to\tau}$ one has to manually stipulate that the interpretations of the quantifiers, equality, truth-functional connectives, and the combinators belong to the model. This style of model was generalized to higher-order modal logic by Gallin (1975), although his logic included the **S5** principle.

Let's begin with a couple of immediate observations. First, note that our notion of model is a special case of a model of higher-order logic according to Definition 15.1 of Chapter 15. Recall that any modalized structure is an ordinary applicative structure, where the domain at type σ is identified with the extension of the modalized set at the designated world, $D^\sigma_{@}$, and application $\mathrm{App}(f,a)$ in a modalized structure is defined as $f(@,a)$. It can also be checked that with the valuation identified with $v_@$ we obtain a model. The clause for the quantifier, for instance, is proved as follows:

$v_@(\mathrm{App}(\mathrm{all}_\sigma,f)) = 1$ iff
$@ \in v_@(all_\sigma(@,f))$ iff
$@ \in \{v \in W \mid v \in f(v,a) \text{ for all } a \in D^\sigma_v\}$ iff
$@ \in f(@,a)$ for all $a \in D^\sigma_@$ iff
$v_@(\mathrm{App}(f,a)) = 1$ for every $a \in D^\sigma_@$.

The clause for the conditional—that $v_@(\mathrm{if}(p)(q)) = 1$ iff $v_@(p) = 0$ or $v_@(q) = 1$—is similarly straightforward. It follows immediately from Theorem 15.1 that the axioms of H are true in a model and the rules preserve truth in a model.

Proposition 18.1. *The set of formulae true in any modal modal is a theory: it contains the axioms of* H *and is closed under the rules of* H.

The second observation is that we can leverage the above clauses to introduce the notion of a formula being *true at* a world (relative to an assignment) allowing us to reason about the semantic values of formulas in a more natural way.

Definition 18.2 (Truth at a world)**.** *A formula A is* true at *w, relative to g, written* $w, g \Vdash A$, *iff* $w \in [\![A]\!]^g_w$.

The semantic clauses above then guarantee the following

Proposition 18.2. *In any modal model, the following biconditionals hold:*

- $w, g \Vdash A \to B$ *iff* $w, g \nVdash A$ *or* $w, g \Vdash B$
- $w, g \Vdash \forall_\sigma x A$ *iff* $w, g[a/x] \Vdash A$ *for all* $a \in D^\sigma_w$.
- $w, g \Vdash \Box A$ *iff, for all* $v \geq w$, $v, i_{wv} \circ g \Vdash A$
- $w, g \Vdash a =_\sigma b$ *iff* $[\![a]\!]^g_w = [\![b]\!]^g_w$.

Comprehension Check 18.1. *The reader should check for themselves that the first two clauses follow from the definition of a model and* $\Vdash$.

In the following exercise, you will establish the final clause, and a version of the penultimate clause involving $\Box_\top$ (i.e. $\lambda p.p = \top$) instead of $\Box$ (since $\Box =_{t\to t} \Box_\top$ is a theorem of Classicism, once we have proved the soundness of Classicism one can also conclude that the penultimate clause holds too).

- $w \in [\![a =_\sigma b]\!]^g_w$ iff $[\![a]\!]^g_w = [\![b]\!]^g_w$
- $w \in [\![A = \top]\!]^g_w$ iff $v \in [\![A]\!]^g_v$ for every $v \geq w$.

First observe that

$w \in [\![\forall_{\sigma\to\tau} Z(Za \leftrightarrow Zb)]\!]^g_w$ iff

$w \in all_{\sigma\to\tau}(w, [\![\lambda Z.(Za \leftrightarrow Zb)]\!]^g_w)$ iff

$w \in [\![\lambda Z(Za \leftrightarrow Zb)]\!]^g_w(w,f)$ for all $f \in D^{\sigma\to\tau}_w$ iff

$w \in [\![Za \leftrightarrow Zb]\!]^{g[f/Z]}_w$ for all $f \in D^{\sigma\to\tau}_w$

Exercise 18.1.

a. *Define a function f by letting, for each $v \geq w$, and $d \in D^\sigma_v$, $f(v,d) = \{u \geq v \mid i_{wu}(a) = d\}$. Check that f is a homomorphism from $D^\sigma \uparrow w \to D^\tau \uparrow w$ and show that $f \in D^{\sigma\to\tau}_w$.*[i]

b. *Using the observation in the text above and part a, prove that $w, g \Vdash a =_\sigma b$ (i.e. $w \in [\![a =_\sigma b]\!]^g_w$) iff $[\![a]\!]^g_w = [\![b]\!]^g_w$*

c. *Show that $w \in [\![A = \top]\!]^g_w$ iff $v \in [\![A]\!]^g_w$ for every $v \geq w$.*

d. *Using Lemma 17.1 (i.e. $i_{wv}[\![A]\!]^g_w = [\![A]\!]^{i_{wv}\circ g}_v$), conclude that $w, g \Vdash \Box A$ iff, for all $v \geq w$, $v, i_{wv} \circ g \Vdash A$.*

18.2 Soundness of modal models

To every class of models, there corresponds a theory consisting of those formulas that are true in every model in that class.

Definition 18.3. *The theory of a class of models $\mathcal{C}$ in a language $\mathcal{L}(\Sigma)$ is the set of formulas that are true in every model $\mathbf{M} \in \mathcal{C}$.*

Given a model $\mathbf{M} = (\mathcal{F}, D^\cdot, [\![\cdot]\!])$ it is sometimes convenient to call the first two components, which determine the interpretation of the logical fragment of a language, a *higher-order frame*.

Definition 18.4 (Higher-order frame)**.** *If $\mathbf{M} = (\mathcal{F}, D^\cdot, [\![\cdot]\!])$ is a model then $\mathcal{G} := (\mathcal{F}, D^\cdot)$ is the* frame *of* $\mathbf{M}$*. A higher-order frame is thus defined as a pair satisfying the first three clauses of Definition 18.1. We also say* $\mathbf{M}$ *is* based *on the frame $\mathcal{G}$.*

The logic *of a class of higher-order frames $\mathcal{C}$ is the theory of the class of models based on frames in $\mathcal{C}$.*

The distinction between models and frames comes from modal logic. In that context, the frame determines the interpretation of the logical operations, $\Box$, $\to$ and so on, and the model augments that with an interpretation of the non-logical constants. The theory of a class of models need not be a logic—it need not be closed under the rule of substitution—but the theory of a class of frames is always closed under the rule of substitution.

Exercise 18.2. *Prove that the theory of a class of higher-order frames is always a logic.*

Comprehension Check 18.2. *Why is the theory of a class of models not necessarily a logic?*

Definition 18.5. *We also will often make use of the following phrases:*

1. *A sentence is valid in a higher-order frame $\mathcal{G}$ if and only if it is true in every model over that frame.*
2. *A sentence (set of sentences) is satisfiable in a frame iff it is (they are) true in some model over the frame.*

3. *A sentence is valid in a class of frames iff it is valid in every frame in the class.*
4. *A logic is characterized by a class of frames iff it is the logic of that class of frames.*
5. *The logic of a class of Kripke frames $\mathcal{C}$ is the logic of the class of higher-order frames $(\mathcal{F}, D)$ with $\mathcal{F} \in \mathcal{C}$ (a similar notion can be defined from the logic of a class of non-pointed Kripke frames).*

Exercise 18.3. *Show that a logic* **L** *is satisfiable in a frame if and only if it is valid in the frame.*

The models we have introduced are *sound and complete* with respect to Classicism. Classicism is the logic of the class of all models, and more generally, for any set of sentences X, X proves A (i.e. A belongs to the smallest theory containing Classicism and X) if and only if every model of X is a model of A. In this section, we prove soundness, which is the left-to-right direction of this implication.

Recall that Classicism may be characterized as the smallest extension of H that is closed under the rule of equivalence:

The Rule of Equivalence If $\vdash Ax_1 \ldots x_n \leftrightarrow Bx_1 \ldots x_n$ then $\vdash A =_{\sigma_1 \to \ldots \sigma_n \to t} B$, provided $x_1 \ldots x_n \notin FV(A) \cup FV(B)$

We have already observed in Proposition 18.1 that, because modal models are models in the sense of Chapter 15, the axioms of H must be true in modal models, and its rules must be preserved. It thus remains to show that the rule of equivalence is also preserved.

Theorem 18.1 (Soundness)**.** *Classicism is sound for modal models.*

Proof. Say that a formula A is *globally satisfied* in a model $\mathbf{M} = (\mathcal{G}, [\![\cdot]\!])$ iff $w \in [\![A]\!]^g_w$ for every world w and assignment g for w. Where $\mathcal{G} = (\mathcal{F}, D)$ and $\mathcal{G}\!\uparrow\! w = ((W \uparrow w, \leq, w), D \uparrow w)$, this is equivalent to saying that $v_w([\![A]\!]^g) = 1$ for every variant of the original model with a different designated world, $\mathbf{M} = (\mathcal{G}\!\uparrow\! w, i_{@_w}[\![\cdot]\!])$, and every assignment g for w. Since we previously observed that modals models, including models of the form $(\mathcal{G}\!\uparrow\! w, [\![\cdot]\!])$, are higher-order models in the sense of Chapter 15, we can immediately infer that all the theorems of H are globally satisfied.

We can also show that the globally satisfied formulas of a model are closed under the rule of equivalence. Suppose $Ax_1 \ldots x_n \leftrightarrow Bx_1 \ldots x_n$ is globally satisfied where $x_1, \ldots, x_n \notin FV(A) \cup FV(B)$. We now try to show that $[\![A]\!]^g_w = [\![B]\!]^g_w$ for any w and assignment g for w, ensuring the global satisfaction of $A = B$.

Let w be any world. Suppose we apply $[\![A]\!]^g_w$ to some arguments $(v_1, a_1), \ldots, (v_n, a_n)$ where $v_1 \leq \ldots \leq v_n$ and $a_i \in D^{\sigma_i}_{v_i}$, to yield an element of $D^t_{v_n}$, i.e. a subset of $W \uparrow v_n$. We will show that this must yield the same result as applying $[\![B]\!]^g_w$ to those arguments. So we show $u \in [\![A]\!]^g_w(v_1, a_1) \ldots (v_n, a_n)$ (where $u \geq v_n$) if and only if $u \in [\![B]\!]^g_w(v_1, a_1) \ldots (v_n, a_n)$.

Since the semantic clause for application is defined relative to representatives that belong to the same world, it will be helpful to shift the function $[\![A]\!]^g_w$ and all of these arguments up to the topmost world, u, so we can use the fact that $Ax_1 \ldots x_n \leftrightarrow Bx_1 \ldots x_n$ is globally satisfied.

But Proposition 17.1 tells us that we can do this. Given $v_1 \leq \ldots \leq v_n \leq u$,

$$[\![A]\!]^g_w(v_1, a_1) \ldots (v_n, a_n) = [\![A]\!]^g_w(v_n, i_{v_1 v_n} a_1)(v_n, i_{v_2 v_n} a_2) \ldots (v_n, a_n)$$

Also, because $i_{v_n u}$ commutes with homomorphisms (in general $i_{wv}(f(w, a_1) \ldots (w, a_n)) = f(v, i_{wv} a_1) \ldots (v, i_{wv} a_n)$), we can apply $i_{v_n u}$ to both sides, on the LHS using the fact that $i_{v_n u}$ is

defined as the$\uparrow u$ of a proposition:

$$[\![A]\!]^g_w(v_1, a_1) \dots (v_n, a_n))\uparrow u = [\![A]\!]^g_w(u, i_{v_1 u} a_1)(u, i_{v_2 u} a_2) \dots (u, i_{v_n u} a_n)$$

Clearly u is the LHS of the first equation iff it's in the LHS of the second. Because on the RHS $[\![A]\!]^g_w$ takes all its arguments from domains at world u, truncating $[\![A]\!]^g_w$ at u and applying then yields the same results:

$$(i_{wu}[\![A]\!]^g_w)(u, i_{v_1 u} a_1)(u, i_{v_2 u} a_2) \dots (u, i_{v_n u} a_n)$$

By Lemma 17.1, $i_{wu}[\![A]\!]^g_w = [\![A]\!]^{i_{wu} \circ g}_u$. So the above is the same as

$$[\![A x_1 \dots x_n]\!]^{(i_{wu} \circ g)[i_{v_1 u} a_1 / x_1, \dots, i_{v_n u} a_n / x_n]}_u$$

Now we can finally apply our assumption that $A\bar{x} \leftrightarrow B\bar{x}$ is universally satisfied to conclude that u is in the above set iff it belongs to

$$[\![B x_1 \dots x_n]\!]^{(i_{wu} \circ g)[i_{v_1 u} a_1 / x_1, \dots, i_{v_n u} a_n / x_n]}_u$$

And thus, by the same reasoning as above, $u \in [\![B]\!]^g_w(v_1, a_n) \dots (v_n, a_n) \uparrow u$. Thus $[\![A]\!]^g_w(v_1, a_n) \dots (v_n, a_n) = [\![B]\!]^g_w(v_1, a_n) \dots (v_n, a_n)$. □

18.3 Standard models, modal completeness and higher-order incompleteness

We have yet to show that there are any structures meeting our definition of a modal model. Constructing structures that meet the criteria is in general quite non-trivial because the criteria are to some extent circular. For instance, in order for the all_σ homomorphism to belong to the model, the propositional domain must contain $\mathrm{all}_\sigma f$ for every homomorphism f representing a property in the model. On the other hand, the space of homomorphisms representing properties depends on what sets of worlds we have included in the propositional domain.

However, certain kinds of modal models are 'full': their functional domains contain as many homomorphisms as possible at each world—they are full modalized function spaces of the form $A \Rightarrow B$—and their propositional domains contain as many propositions as possible at each world—they are all of $B(\mathcal{F})$. Any structure that is full will automatically meet the requirement that $\mathrm{if}, \mathrm{all}_\sigma, \mathrm{eq}_\sigma, s$ and k belong to the model, and so will automatically satisfy the criteria for being a model.

Definition 18.6 (Fullness). *A model* $\mathbf{M} = (\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$ *is*

- functionally full *iff* $D^{\sigma \to \tau} = D^\sigma \Rightarrow D^\tau$ *for every type* σ *and* τ
- propositionally full *iff* $D^\iota = B(\mathcal{F})$.
- standard *iff it is both propositionally and functionally full.*

Functional fullness here corresponds to what we called 'quasi-fullness' in Section 17.1. We have shown, in Exercise 17.6 that the s and k operations, belong to quasi-full structures at the relevant types.

Exercise 18.4. *Suppose that the model* $(\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$ *satisfies all the criteria in Definition 18.1 for being a model except the for the third condition. Show that if* $(\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$ *is functionally and propositionally full, then it automatically satisfies the third condition: it will contain* all_σ, if *and* eq_σ *(s and k have been shown previously).*

Note that the underlying modalized structure of a standard model is completely determined by the pointed Kripke frame they are based on and the modalized set, D^e, representing the individuals.

Because standard models are easy to construct they are useful for establishing many of the independence results we have stated in this book.

Exercise 18.5. *Show that one cannot derive the* B *principle,* $A \rightarrow \Box\Diamond A$, *in Classicism by constructing a standard model in which it fails (you may appeal, without proof, to Proposition 18.2).*[2]

Exercise 18.6. *Show that one cannot derive* BF^e *in Classicism by constructing a standard model in which it fails. Explain why this means the Functionality schema is not a theorem of Classicism either.*

In fact, Exercise 18.5 can be generalized: any schema of propositional modal logic that is not a theorem of S4 can be refuted in a standard model. That is to say, the propositional modal fragment of Classicism is exactly S4: although there are certainly models of Classicism which contains many axiom schemas that are not contained in S4, such as the Brouwerian axiom B, these further axioms cannot be derived from Classicism alone.

Let's make this precise. Consider the higher-order signature Σ that has an infinite stock of non-logical propositional constants, $P_0, P_1, \ldots$. The propositional modal fragment of $\mathcal{L}(\Sigma)$ is the smallest set of formulas containing P_i, and containing $\neg A$, $(A \rightarrow B)$ and $\Box A$ whenever it contains A and B.

Theorem 18.2 (Completeness of S4)**.** *A sentence of the propositional modal fragment of* $\mathcal{L}(\Sigma)$ *is a theorem of Classicism if and only if it is a theorem of* S4.

The following proof assumes some basic familiarity with propositional modal logic.

Proof. We previously showed that every theorem of S4 is a theorem of Classicism. Moreover, by standard results in modal logic (see, for instance, Cresswell and Hughes (1996) p.120) any non-theorem of S4, C, may be refuted at a world @ in a Kripke model over an S4 frame $(W, \leq, [\![\cdot]\!])$. For any modalized set for the individuals D, we have a corresponding standard modal model $((W, \leq, @), D, [\![\cdot]\!])$, using the same refuting world and interpretation function for the propositional letters. Using our earlier notation, $w, g \Vdash A$, for $w \in [\![A]\!]^g_w$, and omitting the g (A is closed) we obtain the following clauses (by Proposition 18.2:

- $w \Vdash A \rightarrow B$ iff $w \nVdash A$ or $w \Vdash B$.
- $w \Vdash \neg A$ iff $w \nVdash A$.
- $w \Vdash \Box A$ iff $v \Vdash A$ for all $v \geq w$.

This means that for sentences in the propositional modal fragment, $w \in [\![A]\!]^g_w$ iff A is true at w in the Kripke model $(W, \leq, [\![\cdot]\!])$. And so, in particular, $@ \notin [\![C]\!]^g_@$ since C is not true at @ in the Kripke model $(W, \leq, [\![\cdot]\!])$. Thus $v_@([\![C]\!]^g_@) = 0$. As we previously observed, the theorems of Classicism are true in any modal model, so Classicism does not prove C. □

One might wonder if Classicism too is complete with respect to standard models. This is not so: there are many claims that are valid in standard models which cannot be derived from Classicism.[3]

Exercise 18.7. *In Exercise 18.6, we showed that* BF^e *is not true in all standard models. Show, by contrast, that* BF^t *is valid with respect to the class of standard models.*

Exercise 18.8. *Show that Rigid Comprehension is valid with respect to the class of standard models.*

Exercise 18.9. *Show that the Strong Leibniz Biconditionals are valid with respect to the class of standard models.*

Could we obtain a system that is complete with respect to standard models by strengthening Classicism somehow, e.g. by adding as axioms these further validities explored in the above exercises?

This turns out to be impossible. For reasons relating to Gödel's incompleteness theorems, the logic of the class of functionally full models is highly non-recursive.

Theorem 18.3 (Gödel's first incompleteness theorem)**.** *No recursively axiomatizable theory in the signature of arithmetic which includes Peano arithmetic is both complete and consistent (contains any sentence or its negation, but never both).*

The incompleteness of the logic of standard Fregean models of Example 15.1—full Henkin models in which $A^t = \{0, 1\}$—is well-known. But it also carries over to the present context where we are considering $\sigma \to \tau$ domains that are only 'full' with respect to the category of modalized sets (and so certain set-theoretic functions are omitted from all inner domains $D_w^{\sigma\to\tau}$). Discussions of the incompleteness of the higher-order logic of the class of Fregean models can be found in many places (see, for instance, Shapiro (1991)).

Let us briefly explain the issue. The incompleteness phenomena is not really distinctive to Fregean models, but rather any class of models of higher-order logic that include at least one model with an infinite domain, and which are 'extensionally full': have $\sigma \to t$ properties coextensive with arbitrary subsets of the σ domain, and similar properties for relations.

Definition 18.7 (Extensional Fullness)**.** *A Leibnizian model* $(A^{\cdot}, \mathrm{App}^{\cdot}, [\![\cdot]\!], v)$ *of higher-order logic (in the sense of Chapter 15) is* extensionally full *for relations of type* $\sigma_1 \to \ldots \sigma_n \to t$ *iff, for every subset* $S \subseteq A^{\sigma_1} \times \ldots \times A^{\sigma_n}$ *there exists an element* $d \in A^{\sigma_1\to\ldots\to\sigma_n\to t}$ *such that, for all* $a_1 \in A^{\sigma_1}, \ldots, a_n \in A^{\sigma_n}$, $v(\mathrm{App}\ldots(\mathrm{App}(d, a_1), \ldots), a_n)) = 1$ *iff* $(a_1, \ldots, a_n) \in S$. *(A non-Leibnizian model is* extensionally full *at type* σ *if every such S that is also closed under Leibniz equivalence (the relation* $v(\mathrm{App}(\mathrm{App}([\![=_\sigma]\!], a), b)) = 1$*) has the same property.)*

Recall from our discussion of incompleteness in Section 15.6 that there is a predicate $\mathrm{Nat} : (e \to e \to t) \to t$ definable in pure higher-order logic stating that a binary relation $<: e \to e \to t$ orders the individuals in its domain in a natural number sequence: there is a least element, a_0, a second element a_1, which is the $<$-least element different from a_0 and so on.[4] It can be shown that when a model is extensionally full, the only elements of the $e \to e \to t$ domain satisfying $\mathrm{Nat}\, X$ will have as their extension a set of ordered pairs that forms a natural number sequence. So for any arithmetical truth $A(<)$ (i.e. true sentence of $\mathcal{L}(\{<\})$), $\forall_{e\to e\to t} X(\mathrm{Nat}\, X \to A(X))$ will be valid in any class of extensionally full models. Conversely, if there is a model of $\exists_{e\to e\to t} X \mathrm{Nat}\, X$, then $\forall_{e\to e\to t} X(\mathrm{Nat}\, X \to A(X))$ is valid only if $A(<)$ is an arithmetical truth. But, as we noted in Section 15.6, the arithmetical truths are not recursively enumerable, so the validities of any class of extensionally full models that includes at least one infinite model will not be recursively enumerable either.

Exercise 18.10. *Show that every standard modal model* $\mathbf{M} = (\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$*, where* $\mathcal{F} = (W, \leq, @)$*, is extensionally full for properties of type* $\sigma \to t$ *by showing that if* $S \subseteq D^{\sigma}_{@}$ *then the function* $f(w, a) = \{v \geq w \mid i_{wv}a = i_{@v}b$ *for some* $b \in S\}$ *is a homomorphism and has S as its extension at the actual world:* $a \in S$ *iff* $v_{@}(f(@, a)) = 1$.

While functional fullness tends to guarantee extensional fullness, relaxing functional fullness or even extensional fullness on its own will not restore axiomatizability. The logic of propositionally full models is not recursively axiomatizable either: roughly the validities in the propositionally quantified modal fragment of the language are not recursively axiomatizable because the propositional quantifiers can simulate quantification over arbitrary sets of world propositions (see Fine (1970), Proposition 6).

18.4 Completeness of modal models

While we cannot obtain completeness with respect to *standard* models, there is a completeness theorem for arbitrary modal models.

Theorem 18.4 (Soundness and Completeness)**.** *Classicism is sound and complete with respect to modal models. A theory containing Classicism in a language* $\mathcal{L}(\Sigma)$ *is consistent if and only if it is true in some modal model.*

Soundness we have proved in Section 18.2. The proof strategy for the completeness portion of the proof can be broken up into three separate ideas. First we consider extending the signature Σ to a larger signature Σ_{∞} with infinitely many new constants. Using Theorem 15.3 from Chapter 15, we construct negation complete witness complete extensions of the theory in question in fragments of this language (recall that a theory is negation complete if it contains any sentence or its negation, and witness complete if, whenever it contains $\exists_{\sigma}F$, it contains Fa for some term $a : \sigma$). Next we follow a standard method for proving completeness in modal logic, and construct a 'canonical frame' from these complete sets of sentences: the set of worlds W of this frame are the complete sets of sentences, and $w \leq v$ iff, whenever a sentence $\Box A \in w$, $A \in v$. Next we wish to construct a model over this Kripke frame. To do this we first describe the modalized term structure of this Kripke frame and use that to construct an isomorphic model. The propositions that exist at w are subsets of W that are defined by a closed sentence of the language w is constructed from, sets of the from $p_A = \{v \in W \mid A \in v\}$ where A involves only constants appearing in w. Similarly, we only let the functions that exist at w, $D^{\sigma \to \tau}_{w}$, be functions that correspond to closed $\sigma \to \tau$ terms of the language w is formulated in. The aforementioned isomorphism is used to spell out the sense in which a function corresponds to a term.

Let T be a consistent theory extending Classicism in a language $\mathcal{L}(\Sigma)$ with Σ countable. Let $\mathcal{L}(\Sigma_{\infty})$ be the result of adding infinitely many constants to $\mathcal{L}(\Sigma)$ at each type. We will be interested in sets of closed sentences w with the following property:

> w is a max con set iff it contains the closed theorems of Classicism in some language $\mathcal{L}(\Sigma_w)$, and is negation and witness complete relative to $\mathcal{L}(\Sigma_w)$, where $\Sigma \subseteq \Sigma_w \subseteq \Sigma_{\infty}$, and is such that $\Sigma^{\sigma}_{\infty} \setminus \Sigma^{\sigma}_{w}$ is infinite for every type σ. ($\mathcal{L}(\Sigma_w)$ thus always denotes the minimal language that contains all the formulas in w.)

Our model will be based on a special pointed Kripke frame, $\mathcal{F} = (W, \leq, @)$, called a *canonical frame* for T. We will begin by defining and studying this.

Definition 18.8 (Canonical frame)**.** *The canonical frame for a theory T extending Classicism is defined as follows*

- @*, our root world, is some max con set containing T.*[5]
- $W = \{w \geq @ \mid w \text{ is a max con set}\}$.
- $w \leq w'$ *iff whenever* $\Box A \in w$, $A \in w'$. *(Note that this implies that* $\mathcal{L}(\Sigma_w) \subseteq \mathcal{L}(\Sigma_{w'})$*).*

Here is a standard lemma concerning the canonical frame that will be useful later.[6]

Proposition 18.3. $\Box A \in w$ *iff* $A \in w'$ *for all* $w' \geq w$.

Proof. The left-to-right direction is trivial. Suppose that $A \in w'$ for all $w' \geq w$. It follows that A is derivable from the set $\{B \mid \Box B \in w\}$ in H, for otherwise $\{B \mid \Box B \in w\} \cup \{\neg A\}$ would be consistent and extend to a negation and witness complete consistent set w', and by construction it would be such that $w' \geq w$ and $A \notin w'$. Since derivations are finite, there is a finite set of sentences $B_1, \ldots, B_n$, with $\Box B_k \in w$, such that $B_1 \wedge \ldots \wedge B_n \rightarrow A$ belongs to H. So $\Box(B_1 \wedge \ldots \wedge B_n \rightarrow A)$ is a theorem of Classicism, and belongs to w, and by definition $\Box B_1, \ldots, \Box B_n \in w$ so $\Box A \in w$ as required. □

Each world w of the canonical frame induces an equivalence relation on terms of being provably identical in w. For each term $M \in \mathcal{L}(\Sigma_w)$ let:

$$[M]_w = \{N \in \mathcal{L}^\sigma(\Sigma_w) \mid M =_\sigma N \in w\}.$$

Observe that because w contains the necessitations of any identities it contains, if $w \leq v$ then $[M]_w \subseteq [M]_v$. The equivalence classes $[M]_w$ together comprise a modalized structure: the term structure, defined below. While the term structure is not itself the structure part of a modal model (because the functional domains at each world of a modal model are comprised of homomorphisms, rather equivalence classes of terms), our strategy will be to construct a model isomorphic to the term structure.

Definition 18.9 (The modal term structure)**.** *The modal term structure is a modalized structure over the canonical frame* $(W, \leq, @)$*. It is defined as follows*

- $T^\sigma_w = \{[M]_w \mid M \in \mathcal{L}(\Sigma_w), FV(M) = \emptyset\}$
- $i_{wv}([M]_w) = [M]_v$
- $\mathrm{App}(w, ([M]_w, [N]_w)) = [MN]_w$.

We extend it to a λ*-interpretation by setting* $(\llbracket c \rrbracket_T)_w := [c]_w$

Exercise 18.11. *Prove that the modal term structure is quasi-functional and contains combinators.*[ii]

Exercise 18.12. *Let* $\llbracket \cdot \rrbracket$ *be the interpretation of* Σ *in T given by* $\llbracket c \rrbracket = [c]_@$*. Prove that if M is a term, and g an assignment mapping each variable* $x_i : \sigma$ *to an element* $[a_i] \in T^\sigma_w$*, then* $\llbracket M \rrbracket^g_w = [M[a_1/x_1 \ldots a_n/x_n]]_w$.

We now have the ingredients we need to define a modal model. First we define the modalized set of propositions. Recall that this must be a subdomain of $B(\mathcal{F})$:

$D^t_w = \{p_{[A]_w} \mid A \in \mathcal{L}^t(\Sigma_w), A \text{ closed}\}$
$i_{wv}(p_{[A]_w}) = p_{[A]_w} \uparrow v = p_{[A]_v}$

where $p_{[A]_w} := \{v \geq w \mid [A]_v \subseteq v\} = \{v \geq w \mid A \in v\}$.[7]

As noted above, if $A \in \mathcal{L}(\Sigma_w)$ and $w \leq v$ then $A \in \mathcal{L}(\Sigma_v)$ (because if $A \in \mathcal{L}(\Sigma_w)$ then $\Box A \in \mathcal{L}(\Sigma_w)$). To show i_{wv} is a well-defined function, note that if $[A]_w = [B]_w$ then $A =_t B \in w$, and so $\Box(A =_t B) \in w$, and since $w \leq v$, $A =_t B \in v$, i.e. $[A]_v = [B]_v$.

Next, we set $D^e = T^e$, i.e.

- $D^e_w = T^e_w = \{[a]_w \mid a \in \mathcal{L}^e(\Sigma_w)\}$
- $i_{wv}[a]_w = [a]_v$

Finally the functional domains are defined in tandem with a pair of mutually inverse isomorphisms, $j^\sigma : D^\sigma \to T^\sigma$ and $h^\sigma : T^\sigma \to D^\sigma$, between the modal term structure and the structure we are defining.

- $j^t(w, p_{[A]_w}) = [A]_w$, $h(w, [A]_w) = p_{[A]_w}$
- $j^e(w, [a]_w) = h^e(w, [a]_w) = [a]_w$
- $j^{\sigma\to\tau}(w,f)$ is defined and identical to $[M]_w$ iff, for all $v \geq w$

$$f(v, h^\sigma(v, [N]_v)) = h^\tau(v, [MN]_v) \text{ (i.e. } = h^\tau(v, \mathrm{App}(v, ([M]_v, [N]_v)))).$$

- $h^{\sigma\to\tau}(w, [M]_w)$ is the homomorphism $f : D^\sigma \uparrow w \to D^\tau \uparrow w$ defined by $f(v, h^\sigma(v, [N]_v)) = h^\tau(v, [MN]_v)$
- $D^{\sigma\to\tau}_w$ consists of those homomorphisms $f : D^\sigma \uparrow w \to D^\tau \uparrow w$ such that $j^{\sigma\to\tau}(w,f)$ is defined.

First we verify that j and h are well-defined and mutual inverses.

Proposition 18.4. *j^σ and h^σ are well-defined and mutual inverses: $j^\sigma \circ h^\sigma = id_{T^\sigma}$, and $h^\sigma \circ j^\sigma = id_{D^\sigma}$.*

Proof. This is trivial at type e. To establish that j^t is well-defined it suffices to show that if $p_{[A]_w} = p_{[B]_w}$ then $[A]_w = [B]_w$. Suppose the antecedent holds, so that for any maximally consistent set $v \geq w$, $A \in v$ iff $B \in v$, and so $A \leftrightarrow B \in v$ for all $v \geq w$. By Lemma 18.3, $\Box(A \leftrightarrow B) \in w$ and thus $A =_t B \in w$, which means finally that $[A]_w = [B]_w$. Clearly, if j^t is well-defined it is an inverse of h^t.

Suppose for induction that $h^\sigma, h^\tau, j^\sigma$ and j^τ are well-defined and mutual inverses.

It is clear from the form of the definition of $h^{\sigma\to\tau}$ and $j^{\sigma\to\tau}$ that they are mutual inverses provided they are well-defined. $h^{\sigma\to\tau}$ is well-defined provided all possible arguments to $f(v,\cdot)$ are of the form $h^\sigma(v, [N]_v)$, which they are by the fact that $h^\sigma(v,\cdot)$ is surjective (this follows from the fact that j^σ and h^σ are inverses). By definition, $j^{\sigma\to\tau}$ is totally defined on $D^{\sigma\to\tau}$. It is also uniquely defined. Suppose two terms, M and M', satisfy the condition for their equivalence class being the value of $j^{\sigma\to\tau}(w,f)$. Thus, for all $v \geq w$ and $N \in \mathcal{L}(\Sigma_v)$, $f(v, h^\sigma(v, [N]_v)) = h^\tau(v, [MN]_v) = h^\tau(v, [M'N]_v)$. It suffices to show that $[M]_w = [M']_w$, or equivalently, $M =_{\sigma\to\tau} N \in w$. By the injectivity of $h^\tau(v,\cdot)$ for each $v \geq w$ we know $[MN]_v = [M'N]_v$ for every $N : \sigma$, since we already knew that $h^\tau(v, [MN]_v) = h^\tau(v, [M'N]_v)$

for every $N : \sigma$. Thus $MN =_\tau M'N \in v$ for any $N : \sigma$. Because v is witness complete, we also have $\forall_\sigma x(Mx =_\tau M'x) \in v$, for every $v \geq w$ (or else we would obtain a contradiction by letting N be the witness for $\exists_\sigma x(Mx \neq_\tau M'x)$). So by Proposition 18.2, $\Box\forall_\sigma x(Mx =_\tau M'x) \in w$, and since Modalized Functionality is a theorem of Classicism, $M =_{\sigma\to\tau} M' \in w$ as required. □

Lastly, we define an interpretation function. For each constant $c \in \Sigma^\sigma$ we let

- $[\![c]\!] = h^\sigma(@, [c]_@)$.

We have thus defined all the components of a modal model $((W, \leq, @), D^\cdot, [\![\cdot]\!])$. It remains to show it is indeed a model.

Proposition 18.5. *$((W, \leq, @), D^\cdot, [\![\cdot]\!])$ is a modal model.*

Proof. We must show that the model contains operations k, s, if, all_σ, and eq satisfying the conditions in Definition 18.1. Indeed, it will turn out that these operations are given, respectively, by $h(@, [K^{\sigma\tau}]_@)$, $h(@, [S^{\sigma\tau\rho}]_@)$, $h(@, [\to]_@)$, $h(@, [\forall_\sigma]_@)$ and $h(@, [\lambda xy.\forall_{\sigma\to t}X(Xx \leftrightarrow Xy]_@)$. Most of this is routine. Cases of K and S follow from the previously established fact that h is a homomorphism of quasi-functional modalized structures. The case of if is straightforward, so I'll just show the cases of all_σ and eq_σ.

To show that $\mathrm{all}_\sigma \in D_@^{(\sigma\to t)\to t}$ we will show that $\mathrm{all}_\sigma = h^{(\sigma\to t)\to t}(@, [\forall_\sigma]_@)$. By definition of h, $h^{(\sigma\to t)\to t}(@, [\forall_\sigma]_@) = g$ where:

$$g(w, h^{\sigma\to t}(w, [F]_w)) = h^t(w, [\forall_\sigma F]_w)$$

It suffices to show that $h^t(w, [\forall_\sigma F]_w) = \mathrm{all}_\sigma(w, h^{\sigma\to t}(w, [F]_w))$. By definition of h^t, the former set is is $p_{[\forall_\sigma F]_w}$, i.e.

$$\{v \geq w \mid \forall_\sigma F \in v\}$$

and the latter set is $\{v \geq w \mid v \in h^{\sigma\to t}(w, [F]_w)(v, d) \text{ for all } d \in D_v^\sigma\}$. Now each $d \in D_v^\sigma$ is of the form $h^\sigma(v, [a]_v)$, and by the definition of h, $h^{\sigma\to t}(w, [F]_w)(v, h^\sigma(v, [a]_v)) = h^t(v, [Fa]_v) = \{u \geq v \mid Fa \in u\}$. So the second set my be written as follows:

$$\{v \geq w \mid Fa \in v \text{ for all } a \in \mathcal{L}^\sigma(\Sigma_v)\}$$

We can now show these two sets are identical. If v is in the former set: $\forall_\sigma F \in v$. Then $Fa \in v$ for all $a \in \mathcal{L}(\Sigma_v)$, by UI. Conversely if v is in the latter set, then $\forall_\sigma F \in v$ since v is witness complete (if it were not, then by negation completeness $\exists_\sigma x.\neg Fx \in v$, and by witness completeness $\neg Fa \in v$ for some a, which given the consistency of v contradicts our assumption).

Let's now show that $h(@, [=_\sigma]_@) = eq$. We will use the fact that all elements of D are of the form $h(x, [a]_x)$, and evaluate both operations by applying them to arguments of this form. Applying the former to arguments of this form yields the following: $h(@, =)(w, h(v, [a]_w))(v, h(v, [b]_v)) = h^t(v, [a = b]_v) = \{u \geq v \mid a =_\sigma b \in u\}$. Apply eq to the same arguments: $\mathrm{eq}(w, h(v, [a]_w))(v, h(v, [b]_v)) = \{u \mid i_{wu}h(w, [a]_w) = i_{vu}h(v, [b]_v)\}$. Since h^σ is a homomorphism, $i_{wu}h(w, [a]_w) = h(u, i_{wu}[a]_w) = h(u, [a]_u)$, and similarly $i_{wu}h(w, [b]_w) = h(u, [b]_u)$. Since $h(u, \cdot)$ is injective $h(u, [a]_u) = h(u, [b]_u)$ if and only if $[a]_u = [b]_u$, if and only if $a =_\sigma b \in u$. Thus $\{u \mid i_{wu}h(w, [a]_w) = i_{vu}h(v, [b]_v)\} = \{u \mid a =_\sigma b \in u\}$, as required. □

Proposition 18.6. *Every element of T is true in* $\mathbf{M} = ((W, \leq, @), D^{\cdot}, [\![\cdot]\!])$.

Proof. Suppose $A \in T$ is a sentence of $\mathcal{L}(\Sigma)$. We want to show that $@ \in ([\![A]\!]_{\mathbf{M}})_@$. Because $h : T \to D$ is a homomorphism of applicative structures that preserves combinators, and since $h(@, [\![c]\!]^T) = h(@, [c]_@) = [\![c]\!]_{\mathbf{M}}$ for each constant c, it follows that $([\![A]\!]_{\mathbf{M}})_@ = h(([\![A]\!]_T)_@)$.

By Exercise 18.12, $([\![A]\!]_T)_@ = [A]_@$ and $h([A]_@) = \{w \in W \mid A \in w\}$. Since $A \in T \subseteq @$, it follows that $@$ is in this set, and thus that $@ \in ([\![A]\!]_{\mathbf{M}})_@$ as required. □

18.5 The disjunction and coherence properties in extensions of classicism

A modal logic, L, is said to have the disjunction property iff, whenever, $\Box A_1 \vee \Box A_2 \vee \ldots \vee \Box A_n \in \mathbf{L}$ then for some i with $1 \leq i \leq n$ $A_i \in \mathbf{L}$.[8] There is an analogous property for extensions of Classicism with respect to the modal operator $\Box$.

Definition 18.10 (The Disjunction Property)**.** *Suppose that* $\mathbf{L}$ *is a higher-order logic containing Classicism in a logical signature* Σ*. Then* $\mathbf{L}$ *has the disjunction property iff, whenever* $\Box A_1 \vee \Box A_2 \vee \ldots \vee \Box A_n \in \mathbf{L}$ *then for some* i *with* $1 \leq i \leq n$ $A_i \in \mathbf{L}$ *for any formulas* $A_1, \ldots, A_n$ *of* $\mathcal{L}(\Sigma)$.

In Classicism, the disjunction property is no stronger than its $n = 2$ case: If $\Box A \vee \Box B \in \mathbf{L}$ then either $A \in \mathbf{L}$ or $B \in \mathbf{L}$. In the following exercise, you will show how to get the $n = 3$ case from the $n = 2$ case, but the argument can clearly be turned into an inductive argument establishing the disjunction property for arbitrary n.

Exercise 18.13. *Using the fact that Classicism contains all instances of the* **S4** *principle, show that the instance of the disjunction property with* $n = 3$ *can be derived from the* $n = 2$ *instance.*

An equivalent way of stating the disjunction property for a logic $\mathbf{L}$ is this: whenever $A_1, A_2, \ldots, A_n$ are consistent formulas of $\mathbf{L}$ then $\Diamond A_1 \wedge \Diamond A_2 \wedge \ldots \Diamond A_n$ is also consistent in $\mathbf{L}$. An interpretation of $\mathbf{L}$ in which this latter conjunction is true approximates an interpretation in which $\Diamond$ means 'consistent with $\mathbf{L}$', and so dually, $\Box$ means 'a logical truth of $\mathbf{L}$' (with respect to the sentences $A_1, \ldots, A_n$).

Indeed, there is a stronger property that a modal logic can have, *coherence*, in which, roughly speaking, $\mathbf{L}$ has an interpretation in which $\Box A$ means A is a logical truth of $\mathbf{L}$.[9] The official definition of coherence for modal logics (in, e.g., Meyer (1971)) does not extend naturally to contexts, like in Classicism, where $\Box$ is not a primitive logical constant, but it is equivalent to the consistency of the set $\mathbf{L} \cup \{\Diamond A \mid A \text{ is consistent in } \mathbf{L}\}$, and so similarly has a natural analogue for extensions of Classicism.

Definition 18.11 (The Coherence Property)**.** *Suppose that* $\mathbf{L}$ *is a higher-order logic containing Classicism in a logical signature* Σ*. Then* $\mathbf{L}$ *has the coherence property iff the higher-order theory determined by the following set*

$$\mathbf{L} \cup \{\Diamond A \mid \neg A \notin \mathbf{L}, A \textit{ closed}\}$$

is consistent. $\mathbf{L}$ *has the coherence property with respect to a class of models (frames) for* $\mathbf{L}$ *iff the above set has a model in the class (has a model over a frame in the class).*

In this section and the next, we will be especially interested in interpretations of higher-order logic in which broad necessity, $\Box$, expresses a kind of logical necessity: a propositional

operator that, roughly speaking stands to reality as the notion of logical truth stands to language. This can be partially articulated by a constraint on the truths of an interpreted modal language $\mathcal{L}$, higher-order or otherwise, containing an operator $\Box$ that is purported to express logical necessity. Let us suppose that the logical truths in a given signature are represented by a logic **L**. Then the operator expressed by '$\Box$' can be thought of as standing to reality as the logic **L** stands to language when, for any closed sentence A of the language:

Logical Necessity '$\Box A$' is true if and only if $A \in \mathbf{L}$.

We also need a constraint to the effect that the logical truths are in fact true: every member of **L** is true. Coherent logics are of special interest because they support an interpretation of broad necessity with this feature:

Exercise 18.14. *Suppose that* **L** *is a coherent extension of Classicism. Show, using the soundness and completeness result of the previous section, that there is a modal model* $\mathbf{M} = (\mathcal{F}, D^{\cdot}, [\![\cdot]\!])$ *in which every sentence of* **L** *is true such that, for every closed sentence A:*

$\Box A$ *is true in* **M** *if and only if* $A \in \mathbf{L}$.

Models in which $\Box$ behaves like a notion of logical necessity, in the sense spelled out above, validate a number of recombinatorial principles. For instance, if R is a non-logical binary predicate, then any claim about R that is logically consistent (according to **L**) must be broadly possible. For instance, in any consistent logic extending Classicism, the claim that R is symmetric, $R = \lambda xy.Ryx$, is consistent.[10] So in the model of $\Box$ as logical necessity, $\Diamond(R = \lambda xy.Ryx)$ must be true. Often logics are consistent with much more exotic claims about R; for instance, one can state that R has inaccessible order type using logical expressions alone, so if this claim is consistent with **L** the model in question will deem this to be a broad possibility for R too. So on interpretations of a language in which Logical Necessity is satisfied, the intepretations of the non-logical constants are extremely modally malleable. It follows that, even in a language in which $\Box$ expresses the desired kind of logical necessity, not every interpretation of the non-logical constants of this language would yield an interpretation satisfying our constraint, Logical Necessity. Suppose the language had two non-logical predicates, M denoting the property of being a man, and B ('bachelor') denoting the conjunction of being unmarried and a man. In any consistent logic $\exists_e x.(Bx \wedge \neg Mx)$ is consistent in that logic, but clearly it should not be possible for there to be something that is an unmarried man and also not a man.[11] Only certain properties and relations are suitable to be the interpretations of non-logical constants, if we want to validate Logical Necessity—entities that stand to reality roughly as the non-logical constants stand to language: entities that are logically simple, as opposed to entities which are logical constructions from simple properties (like being unmarried and a man). We have called such a language 'logically perfect' following Russell (recall Definition 11.1). So understood, Logical Necessity articulates a sort of logical recombinatorialism, in the sense of Lewis (1986), pp. 87–88, concerning the logically simple, fundamental properties and relations that were characteristic of logical atomists of the time such as Wittgenstein and Russell.

It turns out that the disjunction property and coherence properties coincide for extensions of Classicism:

Exercise 18.15. *Suppose that* **L** *is coherent. Show that* **L** *has the disjunction property.*

Because proofs in H are necessarily finite, a set is consistent if all of its finite subsets are. So one can also prove the following:

Exercise 18.16. *Suppose that* **L** *has the disjunction property. Show that* **L** *is coherent.*

Because we have set-wise completeness for logics with respect to the class of modal models that validate that logic, we know that a logic is coherent if and only if it is coherent with respect to the class of all of its models. Note that a logic **L** can also be the logic of a class of models $\mathcal{C}$ that is properly included in the class of all of its models. In this case, a set of sentences can be consistent in **L** (i.e. contain **L** and be contained in a non-trivial theory) while failing to have a model in $\mathcal{C}$. In this case 'having a model in $\mathcal{C}$' is a more demanding notion of consistency for sets of sentences than simply being consistent, despite coinciding with the ordinary notion of consistency with respect to sentences.

Example 18.1. *Let* **L** *be the logic of all standard models. The result of extending* **L** *with the claims 'every one-to-one relation on all the individuals is onto' and, for each n, the claim 'there are at least n individuals' is consistent and has a modal model but does not have a standard modal model.*

Exercise 18.17. *Say that a logic* **L***, characterized by a class of models* $\mathcal{C}$*, is compact with respect to* $\mathcal{C}$*, iff a set of sentences has a model in* $\mathcal{C}$ *whenever every finite set of sentences has a model in* $\mathcal{C}$*. Show that if* **L** *has the disjunction property, and is characterized by and compact with respect to a class of models* $\mathcal{C}$*, then it is coherent with respect to* $\mathcal{C}$*.*

Remark 18.3. There is a variant notion of coherence for consequence relations, $\vdash$, in which it is required that $\{A \mid \vdash A\} \cup \{\Diamond A \mid \nvdash \neg A\} \nvdash \bot$. The disjunction property may be defined analogously, so that $\vdash \Box A \vee \Box B$ only if $\vdash A$ or $\vdash B$. In this case, one needs to assume that $\vdash$ is compact, in the sense that a set is consistent whenever every finite subset is, in order to show that the disjunction property follows from the coherence property. Without compactness the coherence property is strictly stronger.

The following exercise provides us with the existence of a coherent logic and also illustrates that coherence is a signature-dependent property.

Exercise 18.18. *Recall that in a Fregean model* $A^t = \{0, 1\}$*, and* $A^{\sigma\to\tau} = A^\sigma \to A^\tau$*. In this question, you may assume without proof that the set of sentences* **L** *true no matter how the non-logical constants are interpreted over a given Fregean model is an extension of Classicism, and that* $\forall_t p(p \leftrightarrow \Box p) \in \mathbf{L}$*.*

a. Let $\mathbf{L}^\Lambda$ *be the set of logical formulas of* $\mathcal{L}(\Lambda)$ *that are true in a given Fregean model. Show that* $\mathbf{L}^\Lambda$ *is coherent.*

b. Let Σ *be a logical signature containing a non-logical constant* $P : t$*. Let* $\mathbf{L}^\Sigma$ *be the set of formulas of* $\mathcal{L}(\Sigma)$ *that are true in every extension of the previous model interpreting the new non-logical constants in* Σ*. Show that* $\mathbf{L}^\Sigma$ *is not coherent.*

Could the coherence and disjunction properties be extended to higher-order logics that are not extensions of Classicism? Coherence and the disjunction property can only be formulated with respect to languages that contain a modal operator $\Box$ that could plausible express a kind of necessity. Different theories of granularity can also admit logical definitions of operators that play roughly the role that $\Box$ plays in Classicism, but we have no general guarantee that a higher-order logic admits a definition of a modally well-behaved operator.

In Classicism, every necessity sentence, $\Box A$, is equivalent to an identity, namely $A =_t \top$, and every identity $A =_t B$ is equivalent to a necessity sentence, $\Box(A \leftrightarrow B)$. So we may formulate properties that are equivalent, for extensions of Classicism, to the disjunction

and coherence properties in terms of identity. But because they are formulated in terms of identity alone, they are sensible notions in any higher-order logic.

Definition 18.12 (The $=$-Disjunction Property)**.** *Let* $\mathbf{L}$ *be a higher-order logic.* $\mathbf{L}$ *has the* $=$*-disjunction property iff, whenever* $M_1 =_{\sigma_1} N_1 \vee \ldots \vee M_n =_{\sigma_n} N_n \in \mathbf{L}$ *then* $M_i =_{\sigma_i} N_i \in \mathbf{L}$ *for some* $1 \leq i \leq n$*.*

Suppose that $\mathbf{L}$ is an extension of Classicism with the disjunction property, and $M_1 =_{\sigma_1} N_1 \vee \ldots \vee M_n =_{\sigma_n} N_n \in \mathbf{L}$. By the necessity of identity we can infer $\Box M_1 =_{\sigma_1} N_1 \vee \ldots \vee \Box M_n =_{\sigma_n} N_n \in \mathbf{L}$ and by the disjunction property it follows that $M_i =_{\sigma_i} N_i \in \mathbf{L}$ for some $1 \leq i \leq n$. Conversely suppose that $\mathbf{L}$ is an extension of Classicism with the $=$-disjunction property, and $\Box A_1 \vee \ldots \vee \Box A_n \in \mathbf{L}$. It follows that $A_1 = \top \vee \ldots \vee A_n = \top \in \mathbf{L}$ and so by the $=$-disjunction property $A_i = \top \in \mathbf{L}$ for some i, and thus $A_i \in \mathbf{L}$ for some i.

Definition 18.13 ($=$-Coherent Logic)**.** *Suppose that* $\mathbf{L}$ *is a higher-order logic in a logical signature* Σ*. Then* $\mathbf{L}$ *has the coherence property iff the higher-order theory is determined by the following set*

$$\mathbf{L} \cup \{M \neq_\sigma N \mid M \neq_\sigma N \notin \mathbf{L}, M, N \textit{ closed}\}$$

is consistent.

Like in the modal case, the $=$-coherence property is equivalent to the $=$-disjunction property.

Exercise 18.19. *Show that a logic that is an extension of Classicism is* $=$*-coherent iff it is coherent.*

$=$-coherence and the $=$-disjunction property no longer express the idea that a logic is consistent with an interpretation of a modal operator as logical necessity. However, they do express the idea that the logic is consistent with a very fine-grained conception of entities. In a model of $\mathbf{L}$ satisfying $\mathbf{L} \cup \{M \neq_\sigma N \mid \neg M =_\sigma N \notin \mathbf{L}, M, N \text{ closed}\}$, any distinctness between two terms that is not simply ruled out by logic alone (logic according to $\mathbf{L}$) is true. This ensures the principles we called $\mathbf{L}$-Distinctness and $\mathbf{L}$-Identity in Section 13.2, and so is an important property for those interested in the consistency of structured theories of propositions. Again, for reasons we encountered previously, this idea is sensitive to the language it is being formulated in. In a logically imperfect language, the logical complexity of an entity denoted by a constant is not visible in the language, and so is not the sort of thing purely linguistic notions like logic could be sensitive to: there could be true identities between terms that cannot be proven from logic alone (for instance 'to be a bachelor is to be an unmarried man', $B =_{e \to t} \lambda x.(Ux \wedge Mx)$). But the constraint may be satisfied in a logically perfect language, in which case it articulates the idea that there is at most one way to build an entity from the fundamental entities modulo the identities required by logic.

18.6 Coalesced sums

In this section, I will outline a general strategy for establishing that a higher-order logic has the disjunction property or is coherent. It's instructive to consider the strategy in the simpler context of propositional modal logics. Suppose that $\mathbf{T}$ is the theory of a class of pointed Kripke models $\mathcal{C}$, and for any two models in the class there is another model in $\mathcal{C}$—the 'coalesced sum'—that has copies of the two original models along with a new designated

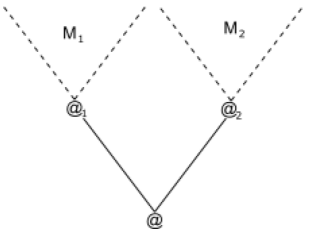


Figure 18.1 The coalesced sum of two pointed Kripke models M_1 and M_2

world, @, that sees every world (i.e. sees each world in the two copies and itself, see Figure 18.1). The truth values of the propositional letters in the coalesced sum are determined by their truth values in each copy as prescribed by the original two models, and the truth values at @ may be assigned however we wish. Doing this guarantees that the interpretations of any letter in the copies can be obtained from their interpretation in the coalesced sum by truncating it about the designated world of the relevant copy. It can also be checked that the truth value of arbitrary formulas in either copy will agree with their truth values in the original pair of models relative to worlds that are in those models.

If **L** is the logic of a class of models like this, then it must have the binary disjunction property (and thus by Exercise 18.13, the full disjunction property). For if A is satisfiable at the designated world $@_1$ in a model in $\mathcal{C}$, and B is also satisfiable at the designated world $@_2$ in a model in C, then $\Diamond A \wedge \Diamond B$ is true at @ in the coalesced sum model, which by assumption is also in $\mathcal{C}$. For in that model @ sees a world, namely $@_1$, at which A is true, and also sees a world, namely $@_2$, at which B is true. It follows that if neither A nor B are in **L**, $\Diamond\neg A \wedge \Diamond\neg B$ is consistent, and so $\Box A \vee \Box B$ is not in **L** either. This is just the disjunction property in contrapositive form.

A similar strategy can be used to show the coherence of a logic or theory. The idea here is to enumerate all the consistent sentences of the logic, $C_1, C_2, C_3, \ldots$ and build a large coalesced sum this time with an infinite number of component models, one for each consistent sentence. Every consistent sentence of the logic will be possible at the new base world @ of the coalesced sum. And conversely, if the coalesced sum is still a model of the logic in question, then every thing possible at the base world will be consistent with the logic.

The strategy is essentially the same in the case of higher-order logic. One can describe a similar construction on modal models, and the interpretations of the constants are chosen so that their truncations by component models match the interpretations in those models. For constants c of type t—i.e. a propositional letter—we use exactly the same strategy described above. A propositional letter gets assigned a set of worlds p in the coalesced model in such a way that when you truncate it by one of the worlds from one of the component models the resulting set, $p \uparrow w$, agrees with the set of worlds assigned to that letter by the component model. (It follows that we only have two possible ways of interpreting the letter, that disagree only on whether the new designated world @ is included in the interpretation or not.) When c has functional type, we similarly chose a homomorphism that has the component interpretations of c has truncations. Recall that the truncation of a homomorphism $f \in D_@^{\sigma\to\tau}$ by w, written $f\uparrow w$, is obtained by simply restricting f to arguments of the form (v, a) where $v \geq w$.

Let us try to make this precise. We'll begin by defining the coalesced sum of some pointed Kripke frames

Definition 18.14 (Coalesced sum of Kripke frames)**.** *Suppose that $\mathcal{F}^i = (W^i, \leq^i, @_i)$ are a family of higher-order frames indexed by $i \in I$. We may assume, without loss of generality, that $W^i \cap W^j = \emptyset$ when $i \neq j$. Their coalesced sum $\nabla_{i\in I}\mathcal{F}^i = (W, \leq, @)$ is defined as follows:*

- $W := \bigcup_i W^i \cup \{@\}$.
- $\leq := \bigcup \leq^i \cup \{(@, w) \mid w \in \bigcup_i W^i \cup \{@\}\}$.
- $@$ *is a new world not appearing in any* W_i.

We will call each frame $\mathcal{F}^i$ a component of the coalesced sum $\nabla_{i\in I}\mathcal{F}^i$.

Definition 18.15 (Coalesced sum of models)**.** *Suppose that for each $k \in K$, $\mathbf{M}_k = (W_k, \leq_k, D_k, @_k, [\![\cdot]\!]^k)$ are modal models. The coalesced sum of these models, $\nabla_{k\in K}\mathbf{M}_k = ((W, \leq, @), D^{\cdot}, [\![\cdot]\!])$, is defined as follows. The frame is the coalesced sum of frames $\nabla_{k\in K}(W_k, \leq_k, @_k)$.*

- *For every $w, v \in W_k$ with $w \leq v$ and type σ, $D^\sigma_w = (D_k)^\sigma_w$, and $i^\sigma_{wv} := (i_k)^\sigma_{wv}$.*
- $D^e_@ := \Pi_k D^e_{@_k}$, $i_{@w}a = i_{@_k w}\pi_k a$ *when $w \in W_k$. $i_{@@}$ is the identity function.*
- $D^t_@ := \{p \subseteq W \mid p{\uparrow}@_k \in D^t_{@_k}$ *for every $k \in K\}$*
- $D^{\sigma\to\tau}_@ := \{f : D^\sigma \to D^\tau \mid f{\uparrow}@_k \in (D_k)^{\sigma\to\tau}_{@_k}$ *for every $k \in K\}$.*
- $[\![c]\!](w, a_1)\dots(w, a_n) := [\![c]\!]_k(w, a_1)\dots(w, a_n)$ *if $w \in W_k$ and*
- $[\![c]\!](@, a_1)\dots(@, a_n) := \bigcup_k [\![c]\!]_k(@_k, i_{@@_k}a_1)\dots(@_k, i_{@@_k}a_n)$ *when c has relational type, $([\![c]\!]_k)_{k\in K} \in \Pi_k D^e_{@_k}$ otherwise.*

We have chosen, for concreteness, to identify individuals with sequences of individuals from the component frames. Informally, a sequence $(a_i)_{i\in I}$ corresponds to the individual a_j in D^e_j. Other choices are possible without fundamentally changing the construction to follow.

The clause for the interpretations of constants requires some explanation. We are appealing to Corollary 17.1 telling us that homomorphisms are defined by their behaviour on arguments drawn from the domains of a single world (by Proposition 17.1, $[\![c]\!](w_1, a_1)\dots(w_n, a_n)$ is equivalent to $[\![c]\!]_k(w_n, i_{w_1 w_n}a_1)\dots(w_n, i_{w_n w_n}a_n)$ whose arguments are all drawn from the domains of w_n). We have chosen the interpretations of the constants in such a way that their extensions at world $v \in W_i$ in $\nabla_k\mathbf{M}_k$ matches their extension at the world v in $\mathbf{M}_i$. At the designated world $@$ we have made the extensions of the constants empty. This choice, again, was made for concreteness: one can set the extensions of the relational constants at the designated world to be anything whatsoever.

Proposition 18.7. *$\nabla_k\mathbf{M}_k$ is a modal model.*

Proof. Clearly $D^t \leq B(\nabla_{k\in K}\mathcal{F}_k)$, and $D^{\sigma\to\tau} \leq D^\sigma \Rightarrow D^\tau$. To show that all_σ, if, k and s are in the relevant domains of the coalesced sum, we must simply show that their truncations to the world $@_k$ are in the model $\mathbf{M}_k$ for each k. The truncation of all_σ, $\mathrm{all}{\uparrow}@_k$, for instance, is just the universal operation for $\mathbf{M}_k$, which must belong to $(D_k)^{\sigma\to\tau}_{@_k}$ because $\mathbf{M}_k$ is itself a model. The same reasoning applies to if, k and s. □

Definition 18.16. *A class of modal models $\mathcal{C}$ is closed under κ-sized coalesced sums iff whenever $(\mathbf{M}_k)_{k\in K}$ are a family of frames in $\mathcal{C}$ and $|K| = \kappa$, then $\nabla_{k\in K}\mathbf{M}_k \in \mathcal{C}$.*

Now we are in a position to prove our central results.

Theorem 18.5. *Let* **T** *be a theory in a countable signature:*

1. *If* **T** *is characterized by a class of models, $\mathcal{C}$, that is closed under binary coalesced sums, then* **T** *has the disjunction property.*
2. *If* **T** *is characterized by a class of models, $\mathcal{C}$, that is closed under countable coalesced sums, then* **T** *is coherent with respect to $\mathcal{C}$ (and therefore also coherent simpliciter).*

Proof. We show 2. The proof of 1 is similar and left as an exercise.

Suppose that **T** and $\mathcal{C}$ are as in the statement of the theorem. Since **T** is characterized by $\mathcal{C}$ every consistent sentence C_k of **T** is true in some model $\mathbf{M}_k = ((W_k, \leq_k, D_k), @_k, [\![\cdot]\!]_k) \in \mathcal{C}$. Since there are countably many consistent sentences A_k we may assume that the variable k ranges over some countable index set K. Now let $\nabla_{k\in K}\mathbf{M}_k$ be a coalesced sum of these these models, which we know also belongs to $\mathcal{C}$. Since **T** is characterized by $\mathcal{C}$, every theorem of **T** is true in $\nabla_{k\in K}\mathbf{M}_k$.

Note that $[\![c]\!]_{@_k} = i_{@@_k}[\![c]\!]_@ = ([\![c]\!]_k)_{@_k}$ for each non-logical and logical constant, and the recursive clauses for extending $[\![\cdot]\!]$ and $[\![\cdot]\!]_k$ to complex expressions are the same so that $[\![\cdot]\!]_{@_k} = ([\![\cdot]\!]_k)_{@_k}$. So, for any of the enumerated consistent sentences, C_k, we have $i_{@@_k}[\![C_k]\!]_@ = [\![C_k]\!]_{@_k} = ([\![C_k]\!]_k)_{@_k}$. Moreover, since C_k is true in $\mathbf{M}_k$, $@_k \in ([\![C_k]\!]_k)_{@_k}$. Thus $@_k \in i_{@@_k}[\![C_k]\!]_@ = [\![C_k]\!]_@ \uparrow @_k$, and finally $@_k \in [\![C_k]\!]_@$. In Exercise 18.1, we showed that $w \in [\![\Box A]\!]_@$ iff $v \in [\![A]\!]_@$ for every $v \geq w$; dually, since we know there is a world $\geq @$ in $[\![C_k]\!]_@$ (namely $@_k$), we know that $@ \in [\![\Diamond_\top C_k]\!]$. Thus the result of a prefixing a $\Diamond$ to any consistent sentence of **T** will be true in $\nabla_k\mathbf{M}_k$. □

Exercise 18.20. *Show that if* **L** *is characterized by a class of general frames, $\mathcal{C}$, that is closed under binary coalesced sums, then* **L** *has the binary disjunction property.*

We immediately get a number of corollaries from this result. We'll divide these up according to logics that are recursively axiomatizable and those that are not. Firstly, we have:

Corollary 18.1. *Classicism is coherent.*

Proof. Since Classicism is characterized by the class of all modal models, and this class is automatically closed under coalesced sums, Classicism is coherent. □

Corollary 18.2. *Classicism plus any of the 64 combinations of the following principles is coherent*

1. *Weak Leibniz Biconditionals/*$\Box$*Weak Leibniz Biconditionals.*
2. *Actuality/*$\Box$*Actuality.*
3. *Barcan formula/*$\Box$*Barcan formula.*

Proof. It suffices to show that a coalesced sum of frames for any one of these principles is also a frame for one of these principles. It follows immediately that a coalesced sum of frames for a combination of these principles is also a frame for that combination of principles.

We will tackle the trickiest case, the Weak Leibniz Biconditionals, and leave the remaining cases as exercises. Suppose $\nabla_k\mathbf{M}_k = ((W, \leq, @), D^{\cdot}, [\![\cdot]\!])$ is a coalesced sum of models $\mathbf{M}_k = ((W_k, \leq_k, @_k)(D_k)^{\cdot}, [\![\cdot]\!]_k)$ with $k \in K$, of the Weak Leibniz Biconditionals. It follows from

the way we defined $D^t_{@}$ that a proposition $p \subseteq W$ is in the domain at @ when $p{\uparrow}@_k \in (D_k)^t_{@}$ for every $k \in K$. Since $\{@\}{\uparrow}@_k = \emptyset \in (D_k)^t_{@}$ for every $k \in K$, it follows that $\{@\}$ is in the propositional domain at @. So: if $p \in D^t_{@}$ is non-empty then it is either $\{@\}$, in which case p satisfies the formula $\exists_t x(\mathrm{WWorld}\, x \wedge x \leq y)$ at @ witnessed by assigning x to $\{@\}$. Or else $p{\uparrow}@_k$ is non-empty for some $k \in K$. So there is some element $q \subseteq W_k$ that exists at $@_k$ such that assigning x and y to q and $p{\uparrow}@_k$ respectively satisfies $\mathrm{WWorld}\, x \wedge x \leq y$ at $@_k$.

We will show that q is a weak world at @: i.e. $@, g \Vdash \Diamond x \wedge \forall_t z(\Box(x \to z) \vee \Box(x \to \neg z))$ where $g(x) = q$ and $g(y) = p$. The first conjunct holds: since $@_k, g \Vdash \Diamond x$, there exists $v \geq @_k$ (and thus $\geq @$) such that $v, g \Vdash x$.

To show $@, g \Vdash \forall_t z(\Box(x \to z) \vee \Box(x \to \neg z))$, let $r \in D^t_{@}$ and $w \geq @$. Since q is a weak world at $@_k$, $(@_k, g[q/x] \Vdash \mathrm{WWorld}\, x)$ it follows that either r is true at every world $w \geq @_k$ at which q is true (i.e. $w, g[r/z] \Vdash x \to z$ for every $w \geq @_k$) or false at every such world (i.e. $w, g[i_{@@_k} r/z] \Vdash x \to z$ for every $w \geq @_k$). Since $q \subseteq W_k$, q is false at every world outside W_k, it follows that follows that either r is true at every world $w \geq @$ at which q is true, or that r is false at every world $w \geq @$, at which p is true. Thus we have shown that $@, g \Vdash \mathrm{WWorld}\, x$. It is clear that $@, g \Vdash x \leq y$ as well, since every q world is in W_k and makes p true.

It is now straightforward to prove that □Weak Leibniz Biconditionals is also preserved under coalesced sums. For if □Weak Leibniz Biconditionals is true in each of the component frames, then in the coalesced sum, it is true at every world apart from @, and by the reasoning above it is also true at @, so □Weak Leibniz Biconditionals is also true in the coalesced sum. □

The proof that Actuality and □Actuality are preserved under coalesced sums is a simpler instance of the above reasoning.

Exercise 18.21. *Show that Actuality, and thus □Actuality, are preserved under coalesced sums.*

For the Barcan formula and its □ one can appeal to the fact that they are equivalent to the principle Functionality.

Exercise 18.22. *Show that Functionality and □Functionality are closed under coalesced sums.*

Thus Classicism plus any combination of Actuality, □Actuality, Functionality, □Functionality, The Weak Leibniz Biconditionals and □The Weak Leibniz Biconditionals are all coherent.

Many principles of propositional modal logic are preserved under coalesced sums as well. Here are a couple of examples:

Proposition 18.8. *The result of adding to Classicism any combination of the McKinsey axiom, the Grzegorczyk axiom and their necessitations is coherent:*

M $\forall_t p(\Box\Diamond p \to \Diamond\Box p)$

Grz $\forall_t p(\Box(\Box(p \to \Box p) \to p) \to p)$

The logics we have considered above are obtained by adding some axioms to Classicism (i.e. taking the smallest higher-order logic containing Classicism and those further axioms) and so are by definition recursively axiomatizable. In Section 15.6, we suggested that the true higher-order logic will not be recursively axiomatizable so it is worth noting that we can also obtain coherence results for non-recursively axiomatizable logics in the same way. For instance, since standard models are extensionally full, the logic of a class of standard higher-order frames that contains at least one frame with infinitely many individuals will not be recursively axiomatizable, for as we observed earlier the truths of arithmetic can be

encoded as validities in such class of frames. Such logics can still be coherent, however, if the class of frames in question is closed under coalesced sums.

In particular, we have that the logic of all standard models is coherent:

Corollary 18.3. *The logic of standard models is coherent with respect to the class of standard models.*

Exercise 18.23. *a. Show that a coalesced sum of propositionally full models is propositionally full.*

b. Show that a coalesced sum of functionally full models is functionally full.

c. Explain why it follows that the logic of standard models is coherent with respect to the class of standard models.

We noted previously that the reason that the logic of a class of frames encodes the arithmetical truths in its validities is that they are extensionally full. More generally, *no* class of standard models that contains at least one model with an infinite domain at some type will have a recursively axiomatizable theory.

Corollary 18.4.

1. *The logic of all extensionally full models is coherent with respect to that class.*
2. *The logic of all extensionally full models of Boolean Completeness is coherent with respect to that class.*

Proof. For part 1 we must check that coalesced sums of extensionally full models is extensionally full. In fact, the coalesced sum of any class of models is extensionally full, even if the components are not. For instance, for any $X \subseteq D^{\sigma}_{@}$ of the coalesced sum, let $f(w, a) = \emptyset$ if $w \neq @$ and $f(@, a) = \{@\}$ iff $a \in X$.

For part 2, we need to check that Boolean Completeness holds in the coalesced sum if it holds in the components. To illustrate, we will show the type t instance of Boolean Completeness. Suppose that $f \in D^{t \to t}_{@}$ and has extension $X \subset D^{t}_{@}$. Define $X \uparrow @_k = \{p \uparrow @_k \mid p \in X\}$. By the extensional fullness of $\mathbf{M}_k$ it follows there is a $t \to t$ operator $g_k \in (D_k)^{t \to t}_{@_k}$ with $X \uparrow @_k$ as its extension, and since Boolean Completeness hold in the $\mathbf{M}_k$, there is a proposition $p_{@_k} \in (D_k)^{t}_{@_k} = D^{t}_{@_k}$ which is the meet of $X \uparrow @_k$ according to $\mathbf{M}_k$. The join of X at $@$ is the union of the $p_{@_k}$ if $@$ doesn't belong to any member of X and is the union of $p_{@_k}$ with $\{@\}$ if $@$ does belong to some member of X. □

Let's end with some remarks about how generally applicable this result is. Firstly, observe that the Kripke frame of a coalesced sum of higher-order frames is always non-symmetric: $@$ sees each $@_k$, but no $@_k$ sees $@$. Thus no class of symmetric frames can be closed under coalesced sums. And since Brouwer's principle B corresponds to symmetry, it seems like our method of proving coherence does not apply to logics including B. Indeed, we can see that coherent logics containing B are quite rare:

Proposition 18.9. *Suppose that* **L** *is a higher-order logic extending Classicism containing* B*, and there exists a sentence A such that* $\Box A$ *and* $\Box \neg A$ *are consistent in* **L**. *Then* **L** *is not coherent.*

Proof. Assume **L** is coherent. Since $\Box A$ and $\Box \neg A$ are consistent in **L** it follows that $\Diamond \Box A$ and $\Diamond \Box \neg A$ must be jointly consistent in **L**. But since the logic contains B we have $\Diamond \Box A \to A$, and $\Diamond \Box \neg A \to \neg A$, these cannot be jointly consistent. □

Whether there is a sentence A such that $\Box A$ and $\Box\neg A$ are consistent in the logic will in general be signature dependent, but it is a very easy condition to satisfy due to the closure of logics under the rule of substitution. For instance, it is satisfied by any consistent logic extending Classicism in a signature with a constant with a relational type $R : \sigma_1 \to \ldots \to \sigma_n \to t$. In that case $\Box\exists x_1 \ldots x_n.Rx_1 \ldots x_n$ and $\Box\neg\exists x_1 \ldots x_n.Rx_1 \ldots x_n$ are both consistent in $\mathbf{L}$. For if $\neg\Box\exists x_1 \ldots x_n.Rx_1 \ldots x_n \in \mathbf{L}$ then by the rule of substitution $\neg\Box\exists x_1 \ldots x_n.(\lambda y_1 \ldots y_n.\top)x_1 \ldots x_n$ and thus $\neg\Box\exists x_1 \ldots x_n.\top$ is in $\mathbf{L}$, meaning that L is not consistent (and thus not coherent). Similarly, if $\neg\Box\neg\exists x_1 \ldots x_n.Rx_1 \ldots x_n \in \mathbf{L}$ then $\neg\Box\neg\exists x_1 \ldots x_n.(\lambda y_1 \ldots y_n.\bot)x_1 \ldots x_n \in \mathbf{L}$ and thus $\Diamond\exists x_1 \ldots x_n.\bot \in \mathbf{L}$ meaning that $\mathbf{L}$ is inconsistent.

It is also satisfied by any consistent logic containing B in a signature containing two non-logical constants of the same type, $c, c' : \sigma$ which also doesn't prove $\forall_\sigma xy.(x =_\sigma y)$—that there is only one object of type σ. Suppose that $\neg\Box(c =_\sigma c')$ belonged to the logic. Then by the rule of substitution $\neg\Box(c =_\sigma c)$ would belong to the logic, and it would be inconsistent (and thus not coherent). So $\Box c =_\sigma c'$ is consistent with the logic. Similarly, suppose $\neg\Box(c \neq_\sigma c')$, or $\Diamond c =_\sigma c'$ is in the logic. Then by the necessity of identity $\Diamond\Box c =_\sigma c'$ is in $\mathbf{L}$ and thus $c =_\sigma c'$. By the rule of substitution, we can get $x =_\sigma y$ and by Gen we have that $\forall_\sigma xy.x =_\sigma y$ is in the logic contradicting our assumption that it was not.

In Exercise 18.18, we showed that the logic of a one world frame—a Fregean model—in the purely logical signature is coherent, and since it is a one world frame it validates B. However, apart from edge cases like these, logics containing B are generally not coherent.

Apart from B, what other principles fail to be preserved under coalesced sums? Although Rigid Comprehension is always true in standard models, and thus is preserved under coalesced sums of standard models, it is not preserved under coalesced sum of arbitrary models. As we previously observed, a coalesced sum is always extensionally full. Given a property f that exists at @ in a coalesced sum, we might hope to find a rigid property coextensive with it as follows. Let $X = \{a \in D^e_{@} \mid @ \in f(@, a)\}$ be the extension of f: we might try to find, for each $@_k$, a rigid property g_k existing at $@_k$ that is coextensive with $\{i_{@@_k}a \mid a \in X\}$ and then set our rigid property to be a function g whose extension at @ is X, and such that $i_{@@_k}g = g_k$ for each index i. However, we have no reason to believe that the relevant g_k exist: although we have assumed that each model $\mathbf{M}_k$ makes Rigid Comprehension true, we have no guarantee that there are *any* properties coextensive with $\{i_{@@_k}a \mid a \in X\}$ at $@_k$ (since general frames are not always extensionally full), and so the truth of Rigid Comprehension at $@_k$ doesn't ensure that we can find a rigid property coextensive with X there either.

Indeed, there is a general argument (adapted from Goodsell (2022)) that there cannot by any recursively axiomatizable coherent extension of Classicism and Rigid Comprehension.

Proposition 18.10. *Let $A(<)$ be any arithmetical sentence (i.e. sentence whose only non-logical constant is $<$). Then the following is a theorem of Classicism + Rigid Comprehension:*

$$\Box\forall_{e\to e\to t}X(\mathrm{Nat}\, X \to A(X)) \vee \Box\forall_{e\to e\to t}X(\mathrm{Nat}\, X \to \neg A(X))$$

The details of this argument are too involved to give here, but can be found in Goodsell (2022). I will, in broad strokes, outline some of the difficulties in proving this and how they may be overcome. First, by an argument of Dedekind, one can prove (in H) that any two natural number structures $X, Y : e \to e \to t$ are isomorphic, and thus make the same arithmetical sentences true: $\forall_{e\to e\to t}XY(\mathrm{Nat}\, X \wedge \mathrm{Nat}\, Y \to (A(X) \leftrightarrow A(Y))$ whenever $A(<)$ is arithmetical. This is provable in H, and thus its necessitation is also a theorem of Classicism,

and so ensures that arithmetical sentences cannot vary their truth value across natural number structures that exist at the same broad possibility. To get our theorem, however, we need to ensure that arithmetical sentences have the same truth values in natural number structures at *different* broad possibilities. There are several obstacles to proving this however. For one thing, how do we compare actual natural number structures X with merely possible natural number structures made out of completely new individuals (which may not actually exist)? They evidently can't be compared actually, because some of the relevant individuals do not exist yet. They cannot be compared at a non-actual possibility where the new structure exists either: even if X is rigid, we have no guarantee that X is still a natural number structure at the relevant non-actual worlds since distinct individuals in Xs domain may have become identical. The trick is to find a natural numbers structure in a higher type consisting of things we know have not collapsed. In particular, the finite cardinality quantifiers *n Fs are G* are pairwise distinct at any possibility where it is possible that there are infinitely many objects, and thus any world with a type e natural number structure. So we can show that any possible natural numbers structure is isomorphic with a single natural numbers structure made from the cardinality quantifiers. The cardinality quantifiers will have retained their distinctness at any other possibility where there is a natural number structure, and so can serve as a constant point of comparison at any such possibility. Finally, Rigid Comprehension ensures that the actual finite cardinality quantifiers necessarily exhaust all the finite cardinality quantifers. This allows us to prove that this particular natural number structure made from cardinality quantifiers makes the same arithmetical sentences true at any world where they form a natural numbers structure (i.e. are distinct from one another). Observe, finally, that if it is merely possible that there are natural number structures, then the cardinality quantifiers are actually distinct and so actually form a natural numbers structure. (And if it is not possible that there are natural numbers structures then the theorem holds vacuously.)

Of course, a standard assumption in mathematics is that there are natural numbers that are well-ordered by some relation $<$: $e \to e \to t$. In the following definition, we will help ourselves to the notion of an *arithmetical truth*: an arithmetical sentence $A(<)$ that is true. Tarski (1936a) has shown that this notion can be made mathematically precise.[12]

Definition 18.17. *A higher-order logic* **L** *is* arithmetically sound *iff, whenever* $\forall_{e\to e\to t}X(\mathrm{Nat}\ X \to A(X)) \in \mathbf{L}$*, the sentence* $A(<)$ *is an arithmetical truth.*

Observe that if a logic proves there are only finitely many individuals then it is not arithmetically sound since $\forall_{e\to e\to t}X(\mathrm{Nat}\ X \to A(X))$ can be proven in the logic to be vacuously true for any formula $A(X)$. (This is to be expected since the arithmetical truths entail there are infinitely many things, namely the natural numbers.)

We can now conclude the following:[13]

Proposition 18.11. *Suppose that* **L** *is a recursively axiomatizable logic extending Classicism + Rigid Comprehension. Suppose also that* **L** *is arithmetically sound. Then* **L** *is not coherent.*

Proof. Suppose for contradiction that **L** is coherent and satisfies the conditions of the proposition.

Since **L** is recursively axiomatizable and arithmetically sound, there is an arithmetical sentence, e.g. the consistency statement for **L**, $\mathrm{Con}_{\mathbf{L}}(X)$, such that **L** does not prove $\forall_{e\to e\to t}X(\mathrm{Nat}\ X \to \mathrm{Con}_{\mathbf{L}}(X))$. Thus $\exists_{e\to e\to t}X.(\mathrm{Nat}\ X \wedge \neg\, \mathrm{Con}_{\mathbf{L}}(X))$ is consistent in **L**.

Since by assumption $\mathbf{L}$ is coherent it is consistent, and so $\mathrm{Con}_{\mathbf{L}}(<)$ is an arithmetical truth. Since $\mathbf{L}$ is arithmetically sound it does not prove $\forall_{e\to e\to t}X(\mathrm{Nat}\ X \to \neg\,\mathrm{Con}_{\mathbf{L}}(X))$, and thus $\exists_{e\to e\to t}X.(\mathrm{Nat}\ X \wedge \mathrm{Con}_{\mathbf{L}}(X))$ is also consistent. But then, by coherence we have that $\Diamond\exists_{e\to e\to t}X.(\mathrm{Nat}\ X \wedge \mathrm{Con}_{\mathbf{L}}(X) \wedge \Diamond\exists_{e\to e\to t}X.(\mathrm{Nat}\ X \wedge \neg\,\mathrm{Con}_{\mathbf{L}}(X)$ must be consistent in $\mathbf{L}$, contradicting Proposition 18.10. □

Endnotes

1. While the logic of the models of Classicism based on substitution structures is a strict strengthening of Classicism, it follows from the results in this chapter that Classicism is sound and complete with respect to the action models in virtue of having the modal models as special cases.
2. Consider a two world Kripke frame where $@ \leq w$ but $w \not\leq @$.
3. Proofs that these claims are not derivable in Classicism can be found in Bacon and Dorr (forthcoming) Appendix D.
4. More precisely $\mathrm{Nat}\ X$ says that a relation X is (i) non-empty $\exists_e yz.Xyz$, (ii) serial, $\forall_e yz.Xyz$, (iii) a well-order, $\forall_{e\to t}Z\exists_e y.(Zy \wedge \forall_e x(Xxy \to \neg Zx))$, and moreover any other relation $Y : e \to e \to t$ that satisfies (i)-(iii) and has the X-least individual in its field is such that X is contained in Y.
5. We know that T may be extended to a witness complete negation complete set by Theorem 15.3.
6. For the propositional modal logic version of this lemma, see e.g. Cresswell and Hughes (1996), Lemma 6.4.
7. These expressions are equivalent because $A \in v$ iff $[A]_v \subseteq v$ when $w \leq v$.
8. See Lemmon (1977).
9. This notion was introduced independently by Meyer (1971) and Fine (unpublished). Fine introduces an extension of the disjunction property, that says that if $A_0 \vee \Box A_1 \vee \ldots \vee \Box A_n \in \mathbf{L}$ and A_0 is a propositional formula, then for some $0 \leq i \leq n$, $A_i \in \mathbf{L}$, and shows that any coherent logic must have the extended disjunction property. In fact, the extended disjunction property is equivalent to coherence; see Bacon and Fine (2023), Proposition 37.
10. If $R \neq \lambda xy.Ryx \in \mathbf{L}$ then, since logics are closed under substitution, $\lambda yz.\top \neq \lambda xy(\lambda yz.\top)yx \in \mathbf{L}$, but this is inconsistent in Classicism.
11. If $\neg\exists_e x.(Bx \wedge \neg Mx) \in \mathbf{L}$ then by the rule of substitution $\neg\exists_e x.((\lambda x.\top)x \wedge \neg(\lambda x.\bot)x) \in \mathbf{L}$ and so $\mathbf{L}$ is inconsistent.
12. One way to make this precise is to appeal to a background set theory, such as ZFC, and define the relationship of a sentence being true in the set-theoretic structure $(\mathbb{N}, <)$. However, it is possible to avoid the detour through set theory in defining arithmetical truth. Observe that when $A(<)$ is an arithmetical sentence then $A(<)$ is a sentence involving quantifiers from a fixed finite set of higher-order quantifiers. So satisfaction and truth may be defined for this sublanguage in a full higher-order language. The basic idea behind this definition is also essentially due to Tarski, although Tarski was working in a higher-order logic that assumed Extensionalism so some complications of Tarski's original definition would be required. (Observe that arithmetical sentences are usually identified with first-order sentences in a signature including primitives for 0, successor, addition, as well as the ordering $<$. However, since we are defining relations corresponding to addition and successor from the only primitive $<$ using second-order quantification, arithmetical sentences in our sense will generally be second-order, not first-order.)
13. This observation is also due to Goodsell.

Hints for exercises

[i] **Hint:** Use the fact that eq is in the model.

[ii] **Hint:** For the second part it suffices to show that $\mathrm{App}(w, \mathrm{App}(w, (i_{@w}[K^{\sigma\tau}]_@, [a]_w)), [b]_w) = [a]_w$ for every $w \in W$ and $a \in \mathcal{L}^\sigma(\Sigma_w), b \in \mathcal{L}^\tau(\Sigma_w)$, and to show a similar property of $S^{\sigma\tau\rho}$.

Part V

Appendices

Appendix A

The Curry-Howard isomorphism

Natural deduction systems for typing terms closely mirror natural deduction systems for propositional logics. We will investigate this connection in this chapter, and explore some of its applications.

A.1 Implicational propositional logics

In what follows we will focus on *implicational* propositional logics.

Definition A.1. *The language of implicational propositional logic consists of an infinite collection of propositional letters, $p_1, p_2, \ldots$ Every letter is a formula, and $A \to B$ is a formula whenever A and B are.*

In addition to propositional letters with subscripts, we will also use the letters p, q and r for ease of readability. We apply the same bracketing conventions to formulas as we do to types. We will, for instance, write $A \to B \to C$ instead of $(A \to (B \to C))$.

An implicational logic is a set of sequents $A_1, \ldots, A_n \vdash B$ where $A_1 \ldots A_n$ and B are implicational formulae. Table A.1 summarizes a collection of rules from which one can derive all the valid sequents of the implicational fragment of intuitionistic propositional logic. Removing Weakening provides an axiomatization of the implicational fragment of *relevant logic*, removing Contraction gives the implicational fragment of *affine logic*, removing both yields the implicational fragment of *linear logic*, and removing also permutation yields the implicational fragment of *ordered logic*.

A proof of a sequent in this system behaves much like a proof of a sequent in our Curry typing system. Try the following exercises to familiarise yourself with this these systems.

Exercise A.1.

a. Derive the sequent $\vdash A \to A$

b. Derive the sequent $\vdash A \to B \to A$

c. Derive the sequent $\vdash (A \to A \to B) \to A \to B$

d. Derive the sequent $\vdash (A \to B \to C) \to B \to A \to C$

e. Derive the sequent $\vdash (A \to B) \to (C \to A) \to C \to B$

f. Derive the sequent $\vdash (A \to B) \to (B \to C) \to A \to C$

g. For each of the above, make note of which logics—relevant, affine, linear or ordered—they can be derived in.

All the tricks you learned for proving sequents in the Curry system apply here too. For instance, if you are trying to prove something of the form $\vdash A \to B$, first try to prove $A \vdash B$.

DOI: 10.4324/9781003039181-25

Table A.1 Natural deduction for propositional logic

$\dfrac{}{A \vdash A}$	**Identity**	$\dfrac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B}$	**Generalized modus Ponens**
$\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}$	**Exchange**	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$	**Conditional proof**
$\dfrac{\Gamma \vdash B}{A, \Gamma \vdash B}$	**Weakening**	$\dfrac{\Gamma \vdash A \to B}{\Gamma, A \vdash B}$	**Reverse conditional proof**
$\dfrac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}$	**Contraction**		

Notice that the principles Conditional Proof and Reverse Conditional Proof are converses of one another. When a rule goes both ways like this, we may notate it as follows:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

It turns out that in the presence of Identity and Generalized Modus Ponens, Reverse Conditional Proof is an admissible rule (in the sense spelled out in Definition 10.3). This means it is strictly speaking redundant, although it does make proving things easier.

Exercise A.2. *Show that Reverse Conditional Proof is an admissible rule in any system closed under Identity and Generalized Modus Ponens.*

In systems without Identity, however, Reverse Conditional Proof may not be admissible. We will call the result of removing Identity from relevant, affine, linear or ordered logic, subrelevant, subaffine, sublinear and subordered logic.

You will have noticed, of course, that we have labeled these propositional logics using the same labels as for the various substructural type theory. The reason is this: if you take a sequent of type theory, $x_1 : \sigma_1 \ldots x_n : \sigma_n \vdash M : \tau$, and simply delete the colons and the λ-terms preceding the types, you get a sequent of the propositional logic. The sequents you arrive at after performing this operation are rather special: the formulas appearing in these sequents will involve only two propositional letters, e and t. This is because the type theory we have been considering thus far only has two base types, e and t. However, it is straightforward to generalize our definition of a type so that there is a base type, p_k, for every propositional letter p_k. The definition of a signature and a Curry system assigning terms to these news types proceeds in exactly the same manner as in Chapter 10.

This establishes a sort of correspondence going from a type theory to a propositional logic.

Definition A.2. *Suppose $\Gamma \vdash M : \sigma$ is a sequent of type theory (in a type system with infinitely many base types, $p_1, p_2, p_3, \ldots$). Then we write $|\Gamma \vdash M : \sigma|$ for the propositional sequent you get by deleting all λ-terms and colons from Γ and $M : \sigma$.*

If Σ is a signature, we write $|\Sigma|$ for the set of types of constants in Σ.

Notice that we use Greek letters like σ, τ and ρ as terms standing for arbitrarily complex types, and latin letters A, B, C, etc. for arbitrarily complex formulae. Thus both σ and A could stand for the $p \to q$. We use the former to indicate it should be thought of as a type, and the latter to indicate that we are thinking of it as a formula, but we will not otherwise distinguish the formula from the type.

To illustrate how this correspondence works, try the following exercise that concerns the sequent version of Modus Ponens, $p \to q, p \vdash q$, in propositional logic.

Exercise A.3. *Derive the sequent* $X : p \to q, y : p \vdash Xy : q$ *in the full Curry type system with infinitely many base types. Now prove the propositional sequent* $|X : p \to q, y : p \vdash Xy : q|$ *using the intuitionistic system (Table A.1).*

Under this correspondence, we see that our typing rules—Application, Abstraction, et cetera—correspond to rules of the propositional logic. Consider, for instance, the typing rule Weakening:

$$\frac{\Gamma \vdash M : \tau}{\Gamma, x : \sigma \vdash M : \tau}$$

If we remove the terms and colons, like $M :$ and $x :$, and replace σ with A and τ with B, and writing $|\Gamma|$ for the result of performing these replacements to the typing judgments in Γ, we get:

$$\frac{|\Gamma| \vdash B}{|\Gamma|, A \vdash B}$$

Under this correspondence, Abstraction corresponds to Conditional Proof, Concretion to Reverse Conditional Proof, Application to Generalized Modus Ponens, and the remaining structural rules Identity, Weakening, Permutation and Contraction to the identically named principles.

The rule Constants is signature dependent and does not correspond to any of the propositional rules we have outlined above. But we might want to augment our propositional logics so that they can accommodate extra premises. Given a set $|\Sigma|$ of additional premises, we can posit a further rule, corresponding to the typing rule Constants

$$\frac{A \in |\Sigma|}{\vdash A}\ \text{Premise}$$

This completes the analogy.

By putting these observations together we can prove the following proposition:

Proposition A.1. *If you can derive* $\Gamma \vdash M : \sigma$ *in intuitionistic, relevant, affine, linear, or ordered type theory with a signature* Σ*, then you can derive* $|\Gamma \vdash M : \sigma|$ *in the corresponding propositional logic with premises in* $|\Sigma|$.

This proposition is true because for each inference rule between sequents in the typing system, there is a corresponding inference between propositional sequents. I leave the details to the reader.

Let's consider an application of this proposition. Notice that some types do not contain any closed terms. For instance, in $\mathcal{L}(\emptyset)$, there are no closed terms (i.e. combinators) of type t. We could reason this out as follows. First we may assume, without loss of generality, that it is in β-normal form. Now it is a fact about β-normal forms that they have one of the following forms: (i) $PM_1 \ldots M_n$ where P is either a constant or a variable and $M_1 \ldots M_n$ are in β-normal form, or (ii) $\lambda x.M$ where M is in β-normal form. (i) is ruled out because there are no constants in this signature, and P cannot be a variable if it is a closed term. (ii) is ruled out because terms beginning with a λ always have a functional type, and t is a base type.

However, this argument was quite labour-intensive and doesn't generalize easily. What if you have constants in the signature, or we were asking about a functional type, like $(t \to t) \to t$? One application of Proposition A.1 is that it gives us a recipe for immediately telling if a type contains a closed term. Let's work through the example of $(t \to t) \to t$. Suppose that there is a closed term of this type in the empty signature: we can derive $\vdash M : (t \to$

$t) \to t$ in our sequent system. By Proposition A.1 that means we can derive the sequent $\vdash (p \to p) \to p$ in intuitionistic logic. But now we can apply our knowledge of propositional logic to determine that $(p \to p) \to p$ is not a theorem of intuitionistic propositional logic. Specifically, we know that classical logic is sound with respect to classical truth tables. Since intuitionistic logic is properly contained in classical logic, it is also sound (but not complete) for classical truth tables. But it is easy to construct a truth table falsifying $(p \to p) \to p$: simply give p the truth value *false*. So $\vdash (p \to p) \to p$ is not an intuitionistally derivable sequent, and so $\vdash M : (t \to t) \to t$ cannot be derivable for *any* term M. We can also apply this method to show things about languages with non-empty signatures. For instance, we can see that having a constant $F : e \to t$ does not allow one to form closed terms of type t. This time we observe that we cannot prove p in intuitionistic logic even with the help a non-logical premise $q \to p$. This can be seen by means of a truth table that makes p and q false.

Above we appealed to the soundness of intuitionistic logic with respect to classical truth tables. However, there are some classical tautologies, like $((t \to e) \to t) \to t)$, which are not intuitionistic theorems so that we would need to appeal to a more general model theory to establish non-theoremhood. We'll outline a more general semantics like this in Section B.1. It is worth noting, however, that the need to use semantic methods beyond classical truth tables is quite rare in practice.

Exercise A.4. *For each of the following types, either show that it contains no combinators, according to the full type system, or find a combinator of that type:*

(i) $e \to t$

(ii) $((t \to t) \to t) \to t$

(iii) $(e \to t) \to t$

Exercise A.5. *Consider the λ-language $\mathcal{L}(\{c\})$ in one constant, $c : (t \to t) \to t$. Find a type involving only ts which does not contain any closed terms.*

Exercise A.6. *Devise a finite signature in which there are closed terms in every type.*

Going in the other direction—from a sequent of propositional logic to a typing sequent—is not as smooth, as there are typically many typing sequents that correspond to the same propositional sequent. For instance, if $\Gamma \vdash M : \sigma \to \tau$ is a derivable sequent, and if x doesn't appear in Γ then $\Gamma \vdash \lambda x.Mx : \sigma \to \tau$ is also a derivable sequent. But removing the term annotations (and substituting Greek for Latin letters in a consistent way) results in the same propositional sequent. Here the two sequents type a pair of η-equivalent terms. One might have thought that the correspondence would become one-one once one ignores distinctions between $\beta\eta$-equivalent terms: that if a propositional sequent can be associated with two different sequents typing different terms, those terms will be $\beta\eta$-equivalent. But this is not the case either:

Exercise A.7. *Derive two sequents $\vdash M : (t \to t) \to t \to t$ and $\vdash N : (t \to t) \to t \to t$ in the full type system in the empty signature where M and N are not $\beta\eta$-equivalent.*

Another source of non-uniqueness is that two constants might have the same type. Thus, for instance, $| \vdash c_1 : p \to q| = | \vdash c_2 : p \to q|$, but the typing sequents are clearly distinct.

Propositional sequents thus do not contain enough information to determine a unique typing sequent. However, we can, in special cases, associate each *derivation* in this propositional logic with a unique typing sequent. To illustrate this, delete all the λ-terms and colons

from the two typing derivations you created in Exercise A.7. The result will be two different propositional derivations of the same formula of propositional logic. Moreover, you can sort of see how the two derivations encode information about how the two (now deleted) λ-terms were built.

This correspondence from propositional proofs to λ-terms (or more generally, type sequents) is another important aspect of the Curry-Howard isomorphism: λ-terms can be thought of as a certain compact sort of representation of natural deduction proofs for propositional logic. It also allows us to prove a sort of converse to Proposition A.1. Namely, if it's possible to derive a propositional sequent then there *exists* a term M which can be prefixed to the conclusion, and variables $x_1 \dots x_n$ which can be prefixed to the premises, to make a derivable typing sequent. The idea is simply to take the derivation of the propositional sequent, find the term that corresponds to that derivation and use it as M.

Proposition A.2. *If $A_1 \dots A_n \vdash C$ is a derivable sequent of intuitionistic (relevant, affine, linear, ordered) propositional logic with premises in $|\Sigma|$, then there exists a term M in variables $x_1 \dots x_n$ such that $x_1 : A_1 \dots x_n : A_n \vdash M : C$ is derivable in the corresponding type theory with signature Σ.*

We will now describe in more detail the correspondence between propositional proofs and type sequents.

It should be noted that the association of propositional proofs with type sequents is not entirely unique. The uniqueness is only up to choice of variables: sequents that can be obtained from each other by relabeling variables will be considered the same for this purpose. It is also possible to associate propositional logics with premises (with the rule Premises) to type systems for languages in signatures containing constants in exactly the premise types. Uniqueness will therefore fail if there are several constants with the same type, as a one line derivation of a premise A using the rule Premises could be associated with any type sequent of the form $\vdash c : A$. To avoid this we just associate some canonical constant of the correct type for each premise in a propositional derivation with additional premises.

Thus we begin with an operation f associating premises to particular constants in Σ that have the type of that premise. We will then define recursively an operation, F, which maps derivations in propositional logic to typing sequents. It should be observed, as we construct F, that the last line of D is $|F(D)|$. (That is, if $F(D) = \Gamma \vdash M : \sigma$, then the last line of the derivation D will be $|\Gamma \vdash M : \sigma|$.)

If D is just an instance of Premise, $\vdash A$, then F maps D to $\vdash f(A) : A$. If D is an instance of Identity, $p \vdash p$, it is mapped to the sequent $x : p \vdash x : p$ (note that that this choice is unique up to relabeling of the variable x).

We suppose, for a recursive definition, that $F(D)$ is already defined, and state what F does to the result of adding an application of Conditional Proof, Reverse Conditional Proof, Permutation, Weakening, or Contraction to the end of D. For Generalized Modus Ponens, one assumes that Fs action on two derivations D_1 and D_2 is defined, and we state what it does to the derivation obtained by combining them using Application.

Conditional Proof. Suppose for induction that $F(D) = \Gamma, x : \sigma \vdash M : \tau$, and we have a derivation D' constructed from D using Conditional Proof:

$$\dfrac{\begin{array}{c}\vdots\\ D\\ \vdots\\ \Delta, A \vdash B\end{array}}{\Delta \vdash A \to B}\ \text{Conditional Proof}$$

Then we set $F(D') = \Gamma \vdash \lambda x.M : \sigma \to \tau$. Note that assuming the last line of the derivation D is $|F(D)| = |\Gamma, x : \sigma \vdash M : \tau|$ then the last line of D' is $|\Gamma \vdash \lambda x.M : \sigma \to \tau|$ which is just $|F(D')|$.

Permutation. Suppose that $F(D) = x_1 : \sigma_1, \ldots, x_k : \sigma_k, x_{k+1} : \sigma_{k+1}, \ldots, x_n : \sigma_n \vdash M : \tau$ and D' is obtained from D by applying Permutation to k and $k+1$th premises as follows

$$\dfrac{\begin{array}{c}\vdots\\ D\\ \vdots\\ A_1 \ldots A_k, A_{k+1}, \ldots A_n \vdash C\end{array}}{A_1 \ldots A_{k+1}, A_k, \ldots A_n \vdash C}\ \text{Permutation}$$

Then we set $F(D')$ to be $x_1 : \sigma_1, \ldots, x_{k+1} : \sigma_{k+1}, x_k : \sigma_k, \ldots, x_n : \sigma_n \vdash M : \tau$.

The other cases are described similarly

- Suppose $F(D) = \Gamma \vdash M : \sigma \to \tau$, and D' continues D with an application of Reverse Conditional Proof. Then $F(D') = \Gamma, x : \sigma \vdash Mx : \tau$ where x is new (i.e. doesn't appear in Γ).
- Suppose $F(D) = \Gamma \vdash M : \tau$, and D' continues D with an application of Weakening. Then $F(D') = x : \sigma, \Gamma \vdash M : \tau$ where x is new.
- Suppose $F(D) = \Gamma, x : \sigma, y : \sigma, \Delta \vdash M : \tau$, and D' continues D with an application of Contraction. Then $F(D') = \Gamma, z : \sigma, \Delta \vdash M[z/x][z/y] : \tau$ where z is new.
- Suppose $F(D_1) = \Gamma \vdash M : \sigma \to \tau$ and $F(D_1) = \Delta \vdash N : \sigma$, and D joins D_1 and D_2 with an application of Generalized Modus Ponens. Then $F(D) = \Gamma, \Delta \vdash MN : \tau$

Exercise A.8.

a. Find a derivation of $p \vdash (p \to q) \to q$, and apply F to it.

b. Find a derivation of $p \to q \vdash (q \to r) \to (p \to r)$ and apply F to it.

c. Find a derivation of $p, q \vdash p$ and apply F to it.

Remark A.1. Curry, Howard, Lambek, Lauchli, and others, have elaborated on this connection between type theory and proof theory in various different ways. In fact, there are many different aspects to this correspondence. Apart from the simple correspondence between types and formulas outlined above, there are also connections between proofs in a natural deduction system and λ-terms (relating to the operation F defined above), and some proof theorists use λ-terms as a compact notation for representing natural deduction proofs. This links the disciplines of proof theory and type theory, allowing one to appeal to theorems in one area to demonstrate things about the other. One important example of this is that the correspondence shows us that various normalization procedures on natural deduction proofs turn out to correspond to the operation of $\beta\eta$-reduction on λ-terms. We will look

Table A.2 Axioms for propositional logics

I	$A \to A$	S	$(A \to B \to C) \to (A \to B) \to A \to C$
B	$(B \to C) \to (A \to B) \to A \to C$	W	$(A \to A \to B) \to A \to B$
B′	$(A \to B) \to (B \to C) \to A \to C$	K	$A \to B \to A$
C	$(A \to B \to C) \to B \to A \to C$		

at another application like this below, where we demonstrate that there is no combinatory basis for the ordered λ-terms by showing the corresponding propositional logic isn't finitely axiomatizable.

Lambek (1986) also showed that there is a correspondence between proofs/λ-terms and arrows in a Cartesian closed category (recall Definition 17.41), linking the disciplines of type theory, proof theory and category theory. Category theory as subsequently played a big role in the study of type theory in computer science.

Remark A.2. The Curry-Howard correspondence also provides a useful perspective on semantic ideas seen in various areas of philosophy and logic, from the modeling of non-classical logics such as intuitionistic logic and relevant logic, to truth-maker semantics for classical logic.

The most straightforward connection is through the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic. According to that interpretation, a sentence is considered true when has at least one proof. Since we are focusing on simple implicational logics, a proof can simply be represented by a λ-term (as we have just established). According to the BHK interpretation, a proof of a conditional, $A \to B$, is a function that takes a proof of A and produces a proof of B. This is, of course, what a term of type $A \to B$ does as well: it combines with a type A term/proof to form a type B term/proof. (It's worth noting that the BHK interpretation also extends to conjunctions and disjunctions, and there are more sophisticated type theories with product and coproduct types which correspond with this part of the interpretation as well.) Other semantics for non-classical propositional logics drawing on Fine (1974) and Urquhart (1972) can be seen through a similar lens: we will explore some of these below. In truth-maker semantics, a sentence is true when it has a 'truth-maker' instead of a proof. Under this analogy, we should think of λ-terms as models of truth-makers. Truth-makers in Kit Fine's work (see e.g. Fine (2017)) inherit some of the order-theoretic properties of models of the untyped λ-calculus as studied in domain theory (see, e.g. Hindley and Seldin (2008), Chapter 16).

A.2 Combinatory languages and Hilbert systems

Earlier we saw that the full λ-language and its various restrictions, in the form of a Curry type system, correspond to propositional logics formulated in natural deduction. Hilbert systems are another way of formulating propositional logics. In this section, we will see that combinatory languages correspond to Hilbert systems.

Natural deduction systems favour rules over axioms. By contrast, a Hilbert system will typically have only one rule, Modus Ponens, and are individuated instead by their axioms. Table A.2 considers some possible axioms which can be added.

In the context of natural deduction, Generalized Modus Ponens was used as a name for an inference between sequents: from $\Gamma \vdash A \to B$ and $\Delta \vdash A$ infer $\Gamma, \Delta \vdash B$. Here the rule

Modus Ponens, by contrast, is an inference between *sentences*: from A and $A \to B$ you may infer B. (We will explain later why this rule is less 'general' than the similarly named rule in natural deduction.) A Hilbert system is defined by a set of axioms and through Modus Ponens as the only form of inference, determines a notion of derivability between sentences:

Definition A.3 (Hilbert systems)**.** *A Hilbert system for an implicational logic consists of a finite set of sentences, H, called the* axioms.

We write $A_1 \dots A_n \vdash_H B$ to mean that B is derivable from assumptions $A_1 \dots A_n$ using Modus Ponens. This means there is a sequence of sentences, t, such that any sentence in t is either (i) identical to one of $A_1 \dots A_n$, (ii) a substitution instance of an axiom $A \in H$, or (iii) follows from earlier sentences in the sequence by Modus Ponens: there are sentences A and $A \to B$ earlier in the derivation, and the sentence in question is B.

We will follow the convention of writing a sequence of letters consisting of the standard axioms (see Table A.2) to represent the Hilbert system consisting of those axioms. Thus, for instance, BCI will represent the Hilbert system consisting of the axioms B, C and I.

To say that you may use any substitution of an axiom simply means that the schematic letters A, B, C et cetera can be replaced by any sentences you like. For instance $(p \to q) \to p \to q$ is in instance of I. A single sentence may be instances of many axioms: $(p \to p \to q) \to p \to p \to q$ is an instance of both I and C, for example. Readers who have not encountered Hilbert systems before, or consider themselves rusty, are invited to try the following exercises:

Exercise A.9. *Show the following*

a. $A, A \to B, B \to C \vdash_\emptyset C$

b. $A \to B, B \to C \vdash_B A \to C$

c. $A \to B, B \to C \vdash_{SK} A \to C$

d. $A \to B \vdash_{SK} (C \to A) \to C \to B$

e. $A \vdash_{CI} (A \to B) \to B$

f. $\vdash_{BC} (A \to B) \to (B \to C) \to A \to C$

g. $\vdash_{SK} A \to A$

It is worth relating the notion of a Hilbert system to the notion of a propositional *logic*:

Definition A.4 (Logic)**.** *A logic,* $\mathbf{L}$*, is a set of implicational sentences* $\mathbf{L}$ *that is closed under Modus Ponens and the rule of uniform substitution: whenever $A \in \mathbf{L}$ and $A \to B \in \mathbf{L}$ then $B \in \mathbf{L}$, and whenever $A \in \mathbf{L}$, $A' \in \mathbf{L}$ where A' is any result of uniformly replacing sentence letters in A with other sentences.*

We write $\Gamma \vdash_{\mathbf{L}} A$ when A belongs to the smallest set of sentences $\mathbf{L} \cup \Gamma$ and closed under Modus Ponens.

Every Hilbert system H determines a unique logic $\mathbf{L}_H$: the set of sentences A such that $\vdash_H A$. Or, equivalently, $\mathbf{L}_H$ may be defined as the smallest set containing every substitution instance of the axioms that is closed under Modus Ponens.

Exercise A.10. *Show that $\Gamma \vdash_H A$ iff $\Gamma \vdash_{\mathbf{L}_H} A$*

However the converse need not hold. Some theories are not of the form $\mathbf{L}_H$ for some Hilbert system H because $\mathbf{L}$ is too complicated to be captured by a finite set of axioms.

We must now pause to talk about an important notational point. The symbol $\vdash$ is being overloaded here. Earlier we considered a system in which the basic units were

sequents, notated with the $\vdash$ symbol. There we introduced metalogical relations between and properties of sequents (the notion of a 'derivable sequent' for instance). In a Hilbert system the basic objects we are reasoning about are not sequents, but *sentences*. And we are now using $\vdash$ to represent a particular metalogical relation between those sentences. The symbols thus have very different meanings. Unfortunately it is standard practice to use the same symbol for both, so we shall just have to make do and tread cautiously.

One particularly unfortunate aspect of this notational overload is this. One can sometimes set up correspondences between Hilbert systems and natural deduction systems in which they can be thought of as different formulations of the same logical content. But the notational overload suggests a particularly simple form this correspondence might take which simply doesn't work: namely that for certain natural deduction systems there is a Hilbert system such that the sequent $A_1 \ldots A_n \vdash B$ is derivable in the former iff $A_1 \ldots A_n \vdash B$ in the Hilbert system—i.e. there is a derivation of B from $A_1 \ldots A_n$. The following exercise shows that for natural deduction systems lacking some of the structural rules, no such Hilbert system can exist.

Exercise A.11. *Show that the rules of Identity, Permutation, Weakening and Contraction for implicational logic are all vacuously true when* $\vdash$ *is interpreted as the metalogical relation of Hilbert consequence.*

A version of the Curry-Howard isomorphism exists for Hilbert systems. Observe that the names of our axioms in Table A.2 are simply the names of the combinators they are the types of. Those who attempted the last two exercises of Section A.9 may also have noticed that there is a close correspondence between those derivations and our attempts, in Chapter 3, to define some combinators out of others. Recall how we showed that the I combinator could be defined from S and K, as SKK with appropriate type superscripts. Using Latin letters for types, we firstly found an S of type $(A \to (B \to A) \to A) \to (A \to (B \to A)) \to A \to A$ and a K of type $A \to (B \to A) \to A$, and combined them to make a term SK of type $(A \to B \to A) \to A \to A$. Then we combined that with a different K of type $A \to B \to A$ to form SKK of type $A \to A$. We could notate this process as follows:

1. $S : (A \to (B \to A) \to A) \to (A \to (B \to A)) \to A \to A$ (instance of S)
2. $K : A \to (B \to A) \to A$ (instance of K)
3. $SK : (A \to (B \to A)) \to A \to A$ (from 1 and 2 by application)
4. $K : A \to (B \to A)$ (instance of K)
5. $SKK : A \to A$ (from 3 and 4 by application)

Now suppose we delete the terms and colons in the above, and replace the word 'application' with 'modus ponens'. We get a derivation of $A \to A$ in the Hilbert system SK:

1. $(A \to (B \to A) \to A) \to (A \to (B \to A)) \to A \to A$ (instance of S)
2. $A \to (B \to A) \to A$ (instance of K)
3. $(A \to (B \to A)) \to A \to A$ (from 1 and 2 by Modus Ponens)
4. $A \to (B \to A)$ (instance of K)
5. $A \to A$ (from 3 and 4 by Modus Ponens)

Exercise A.12. *Repeat the above exercise for the penultimate part of question A.9.*

These examples generalize. Whenever you can define a combinatory term of type A from combinators $X \subseteq \{B, B', C, I, W, S, K\}$, then one can show $\vdash_X A$.

What about inferences with premises, like $A \to B, B \to C \vdash_B A \to C$? Notice that if you have the B combinator, and two constants or variables of type $A \to B$ and $B \to C$ respectively, you may construct a term of type $A \to C$:

1. $c : A \to B$ (constant)
2. $d : B \to C$ (constant)
3. $B : (B \to C) \to (A \to B) \to A \to C$ (instance of B)
4. $Bd : (A \to B) \to A \to C$ (from 3 and 2 by application)
5. $Bdc : A \to C$ (from 4 and 1 by application).

If we delete the terms and colons, replace the words 'application' with 'modus ponens' and 'constant' with 'assumption' we get the desired Hilbert derivation.

1. $A \to B$ (assumption)
2. $B \to C$ (assumption)
3. $(B \to C) \to (A \to B) \to A \to C$ (instance of B)
4. $(A \to B) \to A \to C$ (from 3 and 2 by Modus Ponens)
5. $A \to C$ (from 4 and 1 by Modus Ponens).

Finally, just as the λ-terms can be thought of as ways of coding a natural deduction proof, the combinatory term may be thought of as coding a Hilbert proof. For instance the term SK, with superscripts properly inserted, instructs you to apply Modus Ponens to instances of S and K (where the instances are determined by their superscripts), and thus $(SK)K$ instructs you to append another instance of K to this derivation and use Modus Ponens.

We can reduce these observations to a theorem. If $X \subseteq \{B, B', C, I, K, W, S\}$, we write $|X|$ to denote the Hilbert system made from the corresponding axioms, e.g. $|\{B, C, I\}|$ for **BCI**.

Theorem A.1 (Curry-Howard for Combinatory languages)**.** *Let Σ be a signature and $X \subseteq \{B, B', C, I, K, W, S\}$. Then $|\Sigma| \vdash_{|X|} A$ if and only if M is a term of type A in $CL(X, \Sigma)$*

Proof. For the left to right direction, we show, by induction on term structure, that for every term M of type A in $CL(X, \Sigma)$, A is derivable from $|\Sigma|$ in the Hilbert system $|X|$.

Base case: if M is a constant in Σ of type A, then $A \in |\Sigma|$ so $|\Sigma| \vdash_{|X|} A$. If M is an instance of a combinator in X of type A, then A is a substitution instance of an axiom of $|X|$ so that $|\Sigma| \vdash_{|X|} A$.

Step: Suppose MN has type B, and that M and N are terms of type $A \to B$ and A. By inductive hypothesis we may assume that $|\Sigma| \vdash A \to B$ and $|\Sigma| \vdash A$. By concatenating the proofs of $A \to B$ and A and adding a final application of Modus Ponens, we get a proof of B from $|\Sigma|$.

The right-to-left direction we show by induction that if d is a derivation of A from $|\Sigma|$, there is a term $M : A \in CL(X, \Sigma)$.

Base: The empty derivation is a derivation of no sentence, so the hypothesis holds vacuously.

Step: suppose the hypothesis holds for any derivation of length $\leq n$. Suppose d is a derivation of length $n+1$ that ends with an axiom instance A of $|X|$. Then there is an identically named combinator from X with the type A that is term of $CL(X, \Sigma)$.

Similarly, if d ends with an assumption in $|\Sigma|$, then there is a constant $c \in \Sigma$ in the language $CL(X, \Sigma)$.

If d ends with an application of Modus Ponens from earlier sentences $A \to B$ and A, we can truncate the derivation to obtain two derivations d_1 and d_1 of $A \to B$ and A. By inductive hypothesis there are terms $M : A \to B, N : A \in CL(X, \Sigma)$, and so (MN) is also in $CL(X, \Sigma)$ and is of type B, as required. □

A.3 Correspondences between Hilbert and natural deduction systems

Let's return to the aforementioned correspondence between Hilbert and natural deduction systems. As cautioned earlier, we should not expect, for a given natural deduction system S, to find Hilbert system H such that $A_1 \ldots A_n \vdash_H B$ exactly when $A_1 \ldots A_n \vdash B$ is a derivable sequent of S. $\vdash_H$ stands for a metalogical relation between sentences for which substantive structural rules like contraction, weakening and permutation automatically hold.

In a Hilbert system, one derives sentences not sequents, and in a natural deduction system, one derives sequents not sentences. A more reasonable expectation would be to find a correspondence between sentences of a Hilbert system and sequents of a natural deduction system.

Definition A.5 (Translating from sequents to sentences)**.** *Given a propositional sequent $A_1, \ldots, A_n \vdash B$ we define its sentential translate as follows:*

$$(A_1, \ldots, A_n \vdash B)^* = A_1 \to A_2 \to \ldots \to A_n \to B$$

Exercise A.13. *Find sequents that map to each of the axioms in Table A.2 .*

Note, however, that this translation is not invertible. Distinct sequents can be mapped to the same sentence: for instance $A, B \vdash C$ and $A \vdash B \to C$ both map to the same sentence $A \to B \to C$.

Exercise A.14. *Describe two mappings, $+$ and $-$, from implicational sentences to sequents such that (i) A^+ has the maximum number of premises possible given the constraint that $A^{+*} = A$ and (ii) A^- has the minimum number of premises possible given the constraint that $A^{-*} = A$.*

Let us say that a sequent rule, such as

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Sigma \vdash C}$$

corresponds to the Hilbert inference $(\Gamma \vdash A)^*, (\Delta \vdash B)^* \vdash (\Sigma \vdash C)*$.

Exercise A.15. *Show that Conditional Proof and Reverse Conditional Proof correspond to derivable inferences in any Hilbert system.*

One particular correspondence is worth examining: the Hilbert principle that corresponds to Generalized Modus Ponens is:

Generalized Modus Ponens (Hilbert version) $C_1 \to \ldots \to C_n \to A \to B, D_1 \to \ldots \to D_k \to A \vdash C_1 \to \ldots \to C_n \to D_1 \to \ldots \to D_k \to B$.

Modus Ponens is the special case of this rule where $n = k = 0$ (lending some more transparency to our naming conventions). Generalized Modus Ponens is important for the following reason: we will shortly see that some natural deduction systems do not correspond to finitely axiomatized Hilbert systems, and this can be traced to the limited power of Modus Ponens in comparison to Generalized Modus Ponens. The following exercise illustrates this:

Exercise A.16. *Let ordered logic be the set of sentences that are translates of sequents derivable in ordered natural deduction. Show that it can be finitely axiomatized by all instances of* I*, using Generalized Modus Ponens instead of Modus Ponens as the only rule of proof. (Hint: show by induction on natural deduction proofs, that if you can derive* $A_1 \dots A_n \vdash B$ *then you can prove* $(A_1 \to \dots \to A_n \to B)$ *from instances of Identity and Generalized Modus Ponens.)*

The system described in this exercise is not strictly speaking a Hilbert system, according to our definition, because it has a stronger rule of inference.

Finally, we shall say what it means for a natural deduction system, S, to correspond to a Hilbert system.

Definition A.6. *The system natural deduction S corresponds to a Hilbert system H precisely if the following two conditionals hold:*

- *If* $A_1 \dots A_n \vdash B$ *is a derivable sequent in S with the premise set* Γ*, then* $\Gamma \vdash_H A_1 \to \dots \to A_n \to B$.
- *If* $\Gamma \vdash_H C$ *then* $\vdash C$ *is a derivable sequent in S with the premise set* Γ.

Now that we have defined a reasonably natural correspondence between natural deduction systems and Hilbert systems, we can ask which of our natural deduction systems correspond to which Hilbert systems. If we allow for Hilbert systems with infinitely many axioms it is always possible, as the following exercise demonstrates.

Exercise A.17. *Let S be a natural deduction system. Let* $H := \{(A_1 \dots A_n \vdash B)^* \mid A_1 \dots A_n \vdash B$ *is derivable S from the empty premise set*$\}$*. Show that the Hilbert system H corresponds to S.*

Needless to say, this is hardly satisfying. Ideally we would like to find *finite* Hilbert systems for each of our main systems.

It turns out, however, that given the Curry-Howard isomorphism this is equivalent to a question we have already spent some time on: which classes of λ-terms can be given combinatory bases. This gives us opportunity to explore another example of how theorems about typed languages can translate into theorems about the propositional logics:

Theorem A.2.

- *Intuitionistic natural deduction corresponds to the Hilbert system* SK.
- *Relevant natural deduction corresponds to* BCIW.
- *Affine natural deduction corresponds to* BCK.
- *Linear natural deduction corresponds to* BCI.

Proof. We show the correspondence of linear natural deduction with BCI.

Suppose that $A_1 \dots A_n \vdash B$ is a derivable sequent in linear natural deduction with the premise set Γ. Then by the Curry-Howard isomorphism (Proposition A.1) there is a term M and variables such that $x_1 : A_1 \dots x_n : A_n \vdash M : B$ is derivable in linear type theory in a

signature such that $|\Sigma| = \Gamma$. Thus $\vdash \lambda x_1 \ldots x_n.M : A_1 \to \ldots \to A_n \to B$ is also derivable in that system by n applications of Abstraction. By Theorem 10.2, we know that $\lambda x_1 \ldots x_n.M$ is a linear term, and in Definition 9.16 we described a translation that maps $\lambda x_1 \ldots x_n.M$ to a combinatory term $P \in CL(\{B, C, I\}, \Sigma)$, with the same type as $\lambda x_1 \ldots x_n.M$, defined entirely from the combinators B, C and I. Finally, since we have a term P of type $A_1 \to \ldots \to A_n \to B$ in the signature Σ we can apply the Curry-Howard isomorphism between Hilbert systems and combinatory languages to conclude that $|\Sigma| \vdash_{BCI} A_1 \to \ldots \to A_n \to B$. That is, given our abbreviations, $\Gamma \vdash_{BCI} (A_1 \ldots A_n \vdash B)^*$ as required.

Conversely, suppose that $\Gamma \vdash_{BCI} A$. So by Curry-Howard (Hilbert/combinatory languages) there is some combinatory term P of type A in $CL(\{B, C, I\}, \Sigma)$, where $|\Sigma| = \Gamma$. P translates to closed a linear term M of the λ-calculus and thus by Theorem 10.2, the sequent $\vdash M : A$ is derivable in linear type theory with signature Σ. By the Curry-Howard isomorphism (natural deduction/λ-calculus) $\vdash A$ is a derivable sequent in propositional linear natural deduction with premise set $|\Sigma| = \Gamma$.

This argument is representative: the cases for intuitionistic, relevant and affine natural deduction proceed similarly. □

It's worth stressing how cumbersome this proof would have been without the Curry-Howard isomorphism. Since we may think of λ-terms as encoding a natural deduction derivation, and a combinatory term as encoding a Hilbert deduction, the translations from λ-terms to combinatory terms we appealed to in the proof can be thought of as a conversion procedure that takes a derivation in a natural deduction system of a sequent $A_1 \ldots A_n \vdash B$ and converts it to a derivation in a Hilbert system of the sentence $(A_1 \ldots A_n \vdash B)^*$. Of course, our job is partly made simpler by the fact that we have already done the work of describing this translation procedure between typed languages. Even so, the conversion would be quite fiddly to describe as an operation on our two-dimensional way of depicting derivations, and far less intuitive than our translation procedure between the λ-languages and combinatory languages.

A conspicuous omission from this list is a correspondence between ordered natural deduction and a Hilbert system. The reason we cannot prove that ordered natural deduction corresponds to a finitely axiomatized Hilbert system by the same method is simply this. In Section 9.8 we found a way of translating relevant, linear, and affine terms in terms of a finite collection of different sorts of combinators; however, we couldn't find a way of translating ordered terms in terms of a finite collection of combinators. The existence of this translation proved to be essential in the above argument.

Perhaps there is a different way of establishing a connection between ordered natural deduction and a finite Hilbert system. It turns out there isn't, but it's worth first dispelling an attractive but mistaken thought that might lead one to think otherwise. One might have thought, by simply glancing at the correspondences already established, that the C axiom corresponds to natural deduction systems with Permutation, I to systems with Identity, W to Contraction, K to Weakening and I to Identity. (Given our choice to treat premise combinations in sequents as associative—see Exercise 10.4, where one typed the B combinator, and the surrounding discussion—all of these systems include B.) By that reasoning, ordered natural deduction should correspond to the Hilbert system BI. But it doesn't: you cannot, for example, define the ordered combinator $\lambda XYZw.Xy(Zw) : (p \to q \to r) \to p \to (p \to q) \to p \to r$ from the B and I combinators alone, which, by the Curry-Howard isomorphism, means you cannot prove $(p \to q \to r) \to p \to (p \to q) \to p \to r$ in BI alone.

Exercise A.18. *Fill out the above: assuming $\lambda XyZw.Xy(Zw)$ cannot be defined from B and I, show that $(p \to q \to r) \to p \to (p \to q) \to p \to r$ is not a theorem of* BI.

I end this section with a conjecture:

Conjecture A.1. *No finitely axiomatized Hilbert system corresponds (in the sense of Definition A.6) to ordered natural deduction.*

If this conjecture holds we can use this fact about Hilbert systems to corroborate our suspicion, from Section 9.8, that there is no finitary combinatory basis for the ordered terms. This is a prime example of how the Curry-Howard isomorphism can be useful: well-known techniques for reasoning about propositional logics can be used to establish non-trivial facts about typed languages.

Appendix B

Definability semantics

In this appendix, we explore the Curry-Howard analogy from a semantic perspective. We will develop a particular model theory for non-classical propositional implicational logics and extend it to a model theory for general λ-languages that are defined by the substructural Curry type systems of Chapter 10.

B.1 Definability semantics and metaphysical definability

The Curry-Howard isomorphism distinguishes a particular class of implicational logics that correspond to substructural type theories. In Appendix A we studied the proof theory of these implicational logics. In this section, we will turn to their semantics. We will introduce a flexible class of models for implicational logic, based on early work by Urquhart on relevant logic.[1] In this framework one can find sound and complete semantics for the natural deduction systems one gets by dropping any combination of Weakening, Permutation and Contraction from Table A.1 .

The study of non-classical propositional logics and their semantics is large topic in its own right. But this is not what this book is primarily about, and our discussion will reflect those interests accordingly. The larger goal here is to find models of the substructural type theories described in earlier chapters. Because these type theories do not allow one to type things like the K, W or C combinators, we need to find natural constraints about what operations we let into the functional domains that exclude the semantic counterparts of these combinators. Through the Curry-Howard isomorphism we obtain suitable constraints from the semantics of the corresponding propositional logic. This connection will become clearer in Section B.6.

Definition B.1 (Frame)**.** *A* definability frame $\mathcal{F}$ *is a tuple* $(W, \circ, 0)$ *where*

- $0 \in W$
- $\circ$ *is a binary operation on* W *(i.e.* $\circ : W \times W \to W$*)*

subject to the following laws:

Unit $0 \circ x = x = x \circ 0$ *for every* $x \in W$

Associativity $x \circ (y \circ z) = (x \circ y) \circ z$ *for every* $x, y, z \in W$.

Notice that there is no difference between a definability frame and a monoid (see Section 17.2). However, in the context of monoids we used the numeral 1 for the unit of the monoid; here we use 0 as mnemonic for its role as the empty object of W.

DOI: 10.4324/9781003039181-26

Because of the associativity axiom we may write $x \circ y \circ z$ to unambiguously refer to either $(x \circ y) \circ z$ or $x \circ (y \circ z)$. By omitting the brackets we can take shortcuts in our reasoning that would strictly speaking require appeals to associativity. (I'll entrust it to the reader to keep track of where this is happening.)

Given a definability frame, it is possible to construct interpretations of implicational logic, which I'll simply call the *definability semantics* for propositional logic. The basic idea is this. The frame determines an abstract collection of indices W, that should be thought of concretely as bundles of simple constituents. What sort of bundles these are will depend on the application—for instance they might be sets, multisets or sequences of constituents depending on whether we are working with ordered logic, linear logic, etc. We write $x \Vdash A$ to mean that given the bundle of constituents x one can build something with type A. For instance, if x is a bundle consisting of a property and an individual, of types $e \to t$ and e respectively, then $x \Vdash t$ because one has the ingredients to build a proposition from this bundle. The notion of building an entity from a bundle of constituents is supposed to be related to the notion of metaphysical definability and exact definability ($\rhd$ and $\rhd^!$) explored in Section 13.1, and one may appeal to the same intuitions acquired there.

The reader should be warned that the interpretation of $\Vdash$ will vary slightly depending on whether one is modeling Weakening or not. Without Weakening $x \Vdash A$ corresponds to exact definability: you can build something of type A from x, but you have to use *all* of the constituents in x. When we are modeling logics including Weakening, it is interpreted to mean that you can build something of type A from ingredients in x, but you don't need to use all of them up.

Given two bundles of constituents, x and y, one can combine them into a larger bundle $x \circ y$ that contains them both and nothing more. Concretely, if x and y are sets, multisets or sequences, then $\circ$ is interpreted as, respectively, set union, multiset union, or sequence concatenation. Among the bundles is the *empty* bundle, 0, which adds nothing to another bundle—it is a unit of $\circ$ in the sense that $0 \circ x = x \circ 0 = x$ for every $x \in W$.

To see how we might obtain an interpretation of implicational logic with in a definability frame, let us consider what sorts of bundles of constituents build functional entities of type $A \to B$. Suppose f is a functional entity of type $A \to B$ and can be built from the constituents in x, and that a is an argument for f of type A that can be built from the bundle of constituents y. We know that the result of applying f to a to form something of type B, fa, can be built from $x \circ y$: the constituents in x and in y suffice to build the function and argument.

Remark B.1. Note that the order $x \circ y$ vs $y \circ x$ potentially matters. In Section 13.1, we outlined an interpretation of the building relation in which $\Gamma \rhd F$ and $\Delta \rhd a$ implied $\Gamma, \Delta \rhd Fa$ given the basic logic of building, but in which substantive principles would be needed to derive $\Delta, \Gamma \rhd Fa$ from the same assumptions. In the present notation, the assumption that x builds f and y builds a ensures $x \circ y$ builds fa but doesn't ensure that $y \circ x$ builds fa.

Thus, if you can define something of type $A \to B$ from x, and you can define something of type A from y then you can define something of type B from $x \circ y$:

$x \Vdash A \to B$ only if, for every y such that $y \Vdash A$, $x \circ y \Vdash B$.

This is a one-directional implication, but in fact we will adopt the converse of this principle as well, providing us with an inductive definition of the $\Vdash$ relation assuming we know how it is defined on the propositional letters. A more general semantics could be investigated in

which we only accept one direction of this principle, and treat it as a constraint, rather than a definition, of $\Vdash$. However we will not pursue this idea here. (This corresponds roughly to the distinction between full and non-full models of type theory.)

It should be emphasized that the interpretation of the indices and sentences developed here bears little resemblance with the interpretation of this sort of formalism put forward by relevant logicians and truth-maker semanticists in other contexts. Typically, an index $x \in W$ is thought of as a piece of information, a situation, a truth-maker or state.[2] The formulae of implicational logic are thought of as declarative sentences (as opposed to types) that are verified or falsified by the indices, providing an alethic interpretation of $\Vdash$. And $\circ$ is understood to be an operation that combines information, situations are truth-makers to make more specific or complete information, situations or truth-makers.

Let's formalize our observations with a definition

Definition B.2 (Model). *A model $\mathcal{M}$ is a tuple $(W, \circ, 0, [\![\cdot]\!])$ where $(W, \circ, 0)$ is a definability frame and $[\![\cdot]\!]$ is a function:*

$$[\![\cdot]\!] : \{p_0, p_1, \ldots\} \to P(W)$$

so that $[\![p_k]\!]$ is a set of indices for each letter p_k.

A model thus interprets the sentence letters in a frame by sets of indices, much like a model of modal logic would interpret a sentence by a set of worlds. Like in the modal case, we define the relation of being *true at* a world (or in our interpretation, being *definable from* a bundle of constituents) $x \Vdash A$, inductively as previous discussed.

Definition B.3 (The definability relation). *Given a model $\mathcal{M} = (W, \circ, 0, [\![\cdot]\!])$ we define the relation $x \Vdash A$ as follows*

$x \Vdash p_k$ if and only if $x \in [\![p_k]\!]$ for each sentence letter p_k.
$x \Vdash A \to B$ if and only if, for every $y \in W$, if $y \Vdash A$ then $x \circ y \Vdash B$.

Let's look at some examples of frames and models. Closest to our intended interpretation are the following examples of frames.

Example B.1 (The frame of sequences). *W is the set of finite sequences X^* over some set X, with $\circ$ the operation of sequence concatenation and 0 the empty sequence.*

Example B.2 (The frame of finite sets/multisets). *Let W be the set of finite sets (multisets) from some set X, with $\circ$ the operation of set union (multiset union), and 0 the emptyset $\emptyset$ (empty multiset $[]$).*

The interpretations of these examples were discussed above when X is thought of as a set of simple, or fundamental entities.

These examples provide a flatfooted way of connecting the semantics being explored here with the earlier discussion of metaphysical definability. However the abstract treatment of the indices we have adopted permits many other mathematical objects to play the role of the indices. This will give us plenty of flexibility later when we want to build models of type theory. Here is one example

Example B.3 (Constituent number). *Let $W = \mathbb{N}$, the set of natural numbers, where $\circ$ is addition, and 0 is the additive unit $0 \in \mathbb{N}$. Call this the additive frame over $\mathbb{N}$.*

A model over this frame would assign to each sentence letter a set of natural numbers.

We might think of the claim $n \Vdash$ as telling us that n constituents suffices to build something of type A. If you need n constituents build $f : A \to B$ and m constituents to build an particual argument $a : A$ then you need $n + m$ constiuents to build $fa : B$, so addition appears to be the right meaning for $\circ$ according to this interpretation. (There is, of course, another definability frame one can build over the natural numbers, where $\circ$ is multiplication and 0 is the multiplicative unit $1 \in \mathbb{N}$.)

Let us begin by stating a useful fact about evaluating

Proposition B.1. *$x \vdash A_1 \to \ldots \to A_n \to B$ if and only if, for every $y_1 \Vdash A_1, \ldots, y_n \Vdash A_n$, $(\ldots (x \circ y_1) \circ \ldots \circ y_n) \Vdash B$.*

Exercise B.1. *Prove Proposition B.1.*

Exercise B.2. *Show that*

a. $0 \Vdash A \to A$ *in any model.*

b. $0 \Vdash (B \to C) \to (A \to B) \to A \to C$ *in any model.*

c. In Exercise 10.4 we noted that the unithood of the empty type judgment and the associativity of our method of combining type judgments automatically allowed us to type the I and B combinators. Their types are explored in the previous two parts of the question: briefly describe what would happen if we dropped the unit or associativity constraints from our definition of a definability frame.

Exercise B.3.

a. Find a model $(W, \circ, 0, [\![\cdot]\!])$ and $x \in W$ over any frame of your choosing in which $x \Vdash (p \to p)$ does not hold.

b. Find a model $(W, \circ, 0, [\![\cdot]\!])$ and $x \in W$ over any frame of your choosing in which $x \Vdash (q \to r) \to (p \to q) \to p \to r$ does not hold.

B.2 Validity and frame conditions

We are now in a position to connect our semantics with the natural deduction and Hilbert systems of Chapter A. Our focus will mainly be the natural deduction systems, and more specifically the natural deduction systems containing Identity. We will be able to prove soundness and completeness results for the systems obtained by all eight combinations of Weakening, Permutation and Contraction.

Definition B.4 (Truth in a model and acceptance)**.** *A sentence A is* true *in a model $(W, 0, \circ, [\![\cdot]\!])$ iff $0 \Vdash A$.*

The model accepts *a sequent $A_1, \ldots, A_n \vdash B$ if and only if $(A_1 \ldots A_n \vdash B)^*$ is true in the model. In other words, if and only if $0 \Vdash A_1 \to \ldots \to A_n \to B$.*

Since a sequent cannot be true or false, I have used the word 'acceptance' for the semantic relation that holds between a sequent and a model.

Those familiar with modal logic should be familiar with the notion of frame validity: the property a sentence has when it is true in the frame no matter how the letters are interpreted. We have an analogous notion here:

Definition B.5 (Frame validity)**.** *A sentence A is valid in a definability frame $\mathcal{F} = (W, \circ, 0)$ if and only if A is true in every model $(W, \circ, 0, [\![\cdot]\!])$ based on this frame.*

A sequent is valid in a frame $\mathcal{F}$ if and only if it is accepted by every model based on that frame.

The basic modes of inference in natural deduction systems are sequent rules allowing one to infer a sequent from a set X of premise sequents. A sequent rule is valid when it preserves acceptance in every model.

Definition B.6 (Frame validity for rules)**.** *A rule with premises given by a set of sequents X and conclusion $A_1 \dots A_n \vdash B$ is valid in a definability frame if and only if $A_1 \dots A_n \vdash B$ is accepted by every model based on $\mathcal{F}$ that accepts every element of X.*

Typically X has cardinality 0, 1 or 2: Constants and Identity have no premises, Generalized Modus Ponens has two, and the other rules have only one premise

Exercise B.4. *Consider a rule whose premises are a finite sequence of sequents $s_1 \dots s_n$ and conclusion the sequent t. Say that a rule is schmalid in a frame $\mathcal{F}$ if and only if $s_1^* \to \dots \to s_n^* \to t^*$ is true in every model based on $\mathcal{F}$.*

a. Find a rule from Table A.1 that is valid but not schmalid.

b. For which n does validity and schmalidity coincide?

Exercise B.5. *Show that Generalized Modus Ponens is valid in every definability frame.*

Remark B.2. Given a class of frames we follow the usual convention of saying that a sentence, sequent or rule is valid in that class if it is valid in every frame in the class.

Proposition B.2. *Abstraction, Identity, Concretion and Generalized Modus Ponens are valid in any definability frame.*

Proof. Note that $(A_1, \dots, A_n, B \vdash C)^* = A_1 \to \dots \to A_n \to B \to C = (A_1, \dots, A_n, \vdash B \to C)^*$, so the premise and conclusions of Concretion and Abstraction are identical, and thus vacuous valid.

Identity is valid iff whenever $x \Vdash A$, $0 \circ x \Vdash A$. This holds because $0 \circ x = x$. The validity of Generalized Modus Ponens is left as an exercise. □

Corollary B.1 (Soundness)**.** *Any sequent derivable in the natural deduction system consisting of Abstraction, Identity, Concretion and Application is valid in every definability frame.*

Proof. By induction on derivation length. □

Notice, however, that neither Permutation, Contraction nor Weakening is valid in every frame. It is an instructive exercise to try and find small models that invalidate each of these inferences.

Exercise B.6. *In this question, you are asked to find counterexamples to Permutation, Contraction and Weakening respectively. When asked to find a model, you do not need to specify the interpretation of letters not appearing in the sequents indicated.*

a. Find a model which accepts the sequent $p, q \vdash r$ but not $q, p \vdash r$.

b. Find a model that accepts $p, p \vdash r$ but not $p \vdash r$

c. Find a model that accepts $p \vdash r$ but not $q, p \vdash r$.

There are, however, restricted class of definability frames which validate Permutation and Contraction. For instance the class of commutative frames, in which $x \circ y = y \circ x$ for

every $x, y \in W$ validates Permutation, and the class of idempotent frames, where $x \circ x = x$ validates Contraction.

Exercise B.7.

a. Show that Permutation is valid in commutative frames.

b. Show that Contraction is valid in idempotent frames.

These two classes of frames—the commutative and idempotent frames—are special because they are in fact *characterized* by the Permutation and Contraction rules respectively. For instance, Permutation is not only valid on commutative frames, but any frame whatsoever in which Permutation is valid is commutative. A similar relation holds between the validity of Contraction and idempotence.

Proposition B.3.

1. A definability frame is commutative if and only if Permutation is valid.

2. A definability frame is idempotent if and only if Contraction is valid.

Proof. We have shown the left to right directions of these claims in the previous exercise.

Suppose that Permutation is valid in $(W, \circ, 0)$. Let x and y be arbitrary points in W, and define a model by setting $[\![p]\!] = \{x\}$, $[\![q]\!] = \{y\}$ and $[\![r]\!] = \{x \circ y\}$. Clearly this model accepts the sequent $p, q \vdash r$, and since Permutation is valid it must also accept the sequent $q, p \vdash r$. Thus if $y' \Vdash q$ and $x' \Vdash p$ then $y' \circ x' \Vdash r$. Since y and x are the only indices that build q and r respectively, and $x \circ y$ the only index that builds r, it follows that $y \circ x = x \circ y$. So W must be commutative because x and y were arbitrary.

We leave part 2 to an exercise. □

Exercise B.8. *Show the second part of the previous proposition.*[i]

B.3 Logics with weakening

There is also a frame condition for Weakening, but unfortunately it is the condition that $W = \{0\}$:

Exercise B.9.

a. Show that Weakening is valid in a definability frame iff $x \circ z = x \circ y \circ z$ for all $x, y, z \in W$.

b. Show that this condition is simply equivalent to the condition that $x = 0$ for all $x \in W$, and that there is exactly one definability frame that satisfies this condition.

This means that we have no chance of finding a completeness theorem for logics containing weakening relative to a class of a frames: Weakening is valid only on the empty class and the singleton of the frame explored in the previous exercise, and these classes validate far more sequent rules than can be shown admissible from Weakening alone.

Our informal interpretation of the formalism also sheds some light on this phenomena. Metaphysical views that accept weakened properties subscribe to the idea that if you can build something from a bundle of constituents, you can also build it from that bundle plus a bunch of redundant constituents: a weakened property will simply throw away the redundant constituents with a vacuous λ-abstract. Without assuming commutativity, one can add new constituents to a bundle x in many ways: one can add them on the right to get $x \circ y$, or

on the left to get $y \circ x$, or put it somewhere in the middle. In general, if $x \circ z$ is a bundle, then $x \circ y \circ z$ is a way of adding y to that bundle, where the adding to the right and left are the limiting cases where $z = 0$ and $y = 0$ respectively. The trick to modeling Weakening, then, is to consider a more restricted notion of model over a frame which ensures that the letters have the property that if $x \circ z \in [\![p]\!]$ then $x \circ y \circ z \in [\![p]\!]$.

Definition B.7. *Let $(W, \circ, 0)$ be a definability frame, and let $X \subseteq W$. We will write $X\uparrow$ to denote the smallest set containing X and such that it contains $x \circ y \circ z$ whenever it contains $x \circ z$.*

We shall make this more precise as follows. Let $F(X) = \{x \circ y \circ z \mid y \in W, x \circ z \in X\}$. Let $F^1(X) = X$ and $F^{n+1}(X) = F(F^n(X))$. Then

$$X\uparrow := \bigcup_n F^n(X)$$

Definition B.8 (Persistent model)**.** *A model is* persistent *if and only if $[\![p]\!]\uparrow = [\![p]\!]$ for every propositional letter.*

Definition B.9 (P-validity)**.** *A sentence, sequent or sequent rule is p-valid in a definability frame $\mathcal{F}$ if and only if it is true, accepted or preserves acceptance for each persistent model over $\mathcal{F}$.*

The terminology of persistence comes from the Kripke semantics for intuitionistic logic, where the interpretation of propositional letters was required to be upwards closed under a certain pre-order on the worlds.

Definition B.10. *We define a transitive reflexive relation (a preorder) on W as follows*

- $x \leq y$ *iff* $y \in \{x\}\uparrow$

Exercise B.10. *a. Show that $X = X\uparrow$ if and only if X is closed under $\leq$: if $x \in X$ and $x \leq y$ then $y \in X$.*

b. Show that either of the above is equivalent to the claim that if $x \circ z \in X$ then $x \circ y \circ z \in X$ for any $x, y, z \in W$.

Lemma B.1. *If $x \leq x'$ and $y \leq y'$ then $x \circ x' \leq y \circ y'$.*

Exercise B.11. *Prove by induction that $F^n(\{x\}) \circ F^n(\{y\}) \subseteq F^{2n}(x \circ y)$, writing $X \circ Y$ for $\{u \circ v \mid u \in X, v \in Y\}$. Use this to establish Lemma B.1.*

We will call a sentence A persistent in a model if, whenever $x \circ z \Vdash A$, $x \circ y \circ z \Vdash A$.

Proposition B.4. *Every sentence in a persistent model is persistent. If $x \Vdash A$ and $x \leq y$ then $y \Vdash A$.*

Proof. Clearly this holds for the letters in a persistent model. Suppose A and B are persistent, and that $x \circ z \Vdash A \to B$. We want to show that $x \circ y \circ z \Vdash A \to B$. Suppose that $u \Vdash A$. We want to show that $x \circ y \circ z \circ u \Vdash B$. Since $x \circ z \Vdash A \to B$, we know that $x \circ z \circ u \Vdash B$. And since B is persistent we know that $x \circ y \circ z \circ u \Vdash B$. □

We are now in a position to show that Weakening is p-valid over any definability frame.

Proposition B.5. *Identity, Generalized Modus Ponens, Conditional Proof, Reverse Conditional Proof and Weakening are p-valid over any definability frame.*

Table B.1 Frame conditions for contraction and permutation with respect to p-validity

Idempotence$^-$	$x \circ x \leq x$
Commutativity$^-$	$x \circ y \leq y \circ x$

Proof. The p-validity of Identity, Generalized Modus Ponens, Conditional Proof and Reverse Conditional Proof following from the fact that they are valid, and thus preserve acceptance in all models.

Suppose that $A_1 \ldots A_k, A_{k+1} \ldots A_n \vdash C$ is accepted by a persistent model. We want to show that $A_1 \ldots A_k, B, A_{k+1} \ldots A_n \vdash C$ is also accepted, so suppose that $x_1 \Vdash A_1 \ldots x_n \Vdash A_n$ and $y \Vdash B$. Since the unweakened sequent is accepted by the model $x_1 \circ \ldots \circ x_n \Vdash B$. By associativity we may rebracket this as $(x_1 \circ \ldots \circ x_k) \circ (x_{k+1} \circ \ldots \circ x_n)$. But since B is persistent, $(x_1 \circ \ldots \circ x_k) \circ y \circ (x_{k+1} \circ \ldots \circ x_n) \Vdash B$, establishing that $A_1 \ldots A_k, B, A_{k+1} \ldots A_n \vdash C$ is accepted. □

When working with the notion of p-validity instead of validity, the frame conditions for Permutation and Contraction change. Clearly idempotence suffices for the p-validity of Contraction, because it suffices for the validity of Contraction, and every validity is a p-validity. Similarly Permutation is p-valid in any commutative frame, since it is valid in any commutative frame. But the p-validity of Contraction and Permutation correspond to in principle weaker constraints on definability frames than the validity of these rules.[3]

Exercise B.12. *Consider the definability frame where* $W = [0, 1]$, $\circ$ *is multiplication and* $0 := 1$.

a. Show that $x \leq y$ *for all* $x, y \in W$.

b. Show that Contraction is p-valid in this frame.

c. Show that Contraction is not valid in this frame.

d. Find a frame in which Permutation is p-valid but not valid.

Proposition B.6.

1. *Permutation is p-valid in a definability frame if and only if it is commutative*$^-$
2. *Contraction is p-valid in a definability frame if and only if it is idempotent*$^-$

Proof. I leave the proof that Permutation and Contraction are p-valid in commutative$^-$ and idempotent$^-$ frames to an exercise.

Suppose that Permutation is p-valid in $(W, 0, \circ)$, and suppose for contradiction that $y \circ x \not\leq x \circ y$. Choose a persistent interpretation where $[\![p]\!] = \{x\}\uparrow, [\![q]\!] = \{y\}\uparrow$ and $[\![r]\!] = \{x \circ y\}\uparrow$. If $x' \Vdash p$ and $y' \Vdash q$ then $x \leq x'$ and $y \leq y'$, so by Lemma B.1, $x \circ y \leq x' \circ y'$ and $x' \circ y' \Vdash r$. So $p, q \vdash r$ is accepted by this model. Since Permutation is p-valid it follows that $q, p \vdash r$ is accepted. Since $y \Vdash q$ $x \Vdash p$ $y \circ x \Vdash r$, i.e. $x \circ y \leq y \circ x$, a contradiction.

Now suppose, again for contradiction, that Contraction is p-valid and that $x \circ x \not\leq x$ for some $x \in W$. Now consider a definability frame where $[\![p]\!] = \{x\}\uparrow$ and $[\![q]\!] = \{x \circ x\}\uparrow$. Whenever $x' \Vdash p$ and $y' \Vdash p$, $x \leq x'$ and $x \leq y'$ so $x \circ y \leq x' \circ y'$ by Lemma B.1. So $x' \circ y' \Vdash q$, which means $p, p \vdash q$ is accepted. Since Contraction is p-valid, $p \vdash q$ is also accepted. Since $x \vdash p$, $x \vdash q$ which means $x \in \{x \circ x\}\uparrow$, i.e. $x \circ x \leq x$, another contradiction. □

Exercise B.13 (Optional)**.** *In this exercise we will explore a generalization of definability frames that do not impose the associativity constraint, that* $(x \circ y) \circ z = x \circ (y \circ z)$*. A persistent model is now one such that whenever* $x \circ z \in [\![p]\!]$*,* $(x \circ y) \circ z \in [\![p]\!]$ *and* $x \circ (y \circ z) \in [\![p]\!]$*.*

a. Show that B *is p-valid in definability frames satisfying the condition that* $x \circ (y \circ z) \leq (x \circ y) \circ z$

b. Show that B′ *is p-valid in definability frames satisfying the condition that* $x \circ (y \circ z) \leq (y \circ x) \circ z$

B.4 Completeness

We are now in a position to provide completeness theorems for each of our natural deduction systems containing Identity. Recall, again, that we use $\mathbf{ND}[]$ to stand for the system with Conditional Proof, Reverse Conditional Proof and Generalized Modus Ponens. We use I, P, C and W to stand for Identity, Permutation, Contraction and Weakening respectively, and insert them into the square brackets to obtain various strengthenings of $\mathbf{ND}[]$—for instance, $\mathbf{ND}[IPCW]$ refers to the full system.

1. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[I]$ iff it is valid in every frame.
2. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IP]$ iff it is valid in every commutative frame.
3. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IC]$ iff it is valid in every idempotent frame.
4. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IPC]$ iff it is valid in every commutative idempotent frame.
5. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IW]$ iff it is p-valid in every frame.
6. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IPW]$ iff it is p-valid in every commutative frame.
7. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[ICW]$ iff it is p-valid in every idempotent frame.
8. A sequent $\Gamma \vdash A$ is derivable in $\mathbf{ND}[IPCW]$ iff it is p-valid in every commutative idempotent frame.

Theorem B.1 (Completeness)**.** *1-8 all are true.*

Proof. In each case we construct a single model, satisfying the relevant conditions, that accepts a sequent iff it is provable.

- W is the set of equivalence classes, $[\Gamma]$ of sequences of sentences, Γ, where the equivalence relation is (i) identity (for cases 1,5), (ii) being sequences related by a permutation (for cases 2,6), (iii) being sequences related by contractions or duplications of sentence letters (3, 7), (iv) being sequences related by either permutations or contractions/duplications (4,8).
- $[\Gamma] \circ [\Delta] := [\Gamma, \Delta]$.

For the chosen system, $\mathbf{ND}[X]$, we define:

- $[\Gamma] \Vdash A$ if and only if $\Gamma \vdash A$ is derivable in $\mathbf{ND}[X]$

This is well-defined: one can show if $\Delta \in [\Gamma]$, then $\Delta \vdash A$ iff $\Gamma \vdash A$ in the chosen system. (Because the equivalence relation on contexts allows exactly the rearrangements you can make in the corresponding logic.)

Table B.2 Associativity and unit laws

$(\epsilon, \Gamma) = \Gamma$	**Right unit**
$(\Gamma, \epsilon) = \Gamma$	**Left unit**
$(\epsilon, \Gamma) = (\Gamma, \epsilon)$	**Unit permutation**
$(\epsilon, \epsilon) = \epsilon$	**Unit contraction**
$((\Gamma, \Delta), \Sigma) = (\Gamma, (\Delta, \Sigma))$	**Associativity**
$((\Gamma, \Delta), \Sigma) = (\Delta, (\Gamma, \Sigma))$	**Twisted associativity**

Next one shows that this definition of $\Vdash$ really defines a model. In cases of 1-4 we must show it defines an arbitrary model. I.e. we must show:

$[\Gamma] \Vdash A \to B$ iff, whenever $[\Delta] \Vdash A$, $[\Gamma] \circ [\Delta] \Vdash B$

Left to right: suppose $[\Gamma] \Vdash A \to B$ and $[\Delta] \Vdash A$. So $\Gamma \vdash A \to B$ and $\Delta \vdash A$, and thus $\Gamma, \Delta \vdash B$ by Generalized Modus Ponens. So $[\Gamma, \Delta] \Vdash B$ as required.

Right to left: suppose $[\Gamma] \circ [\Delta] \Vdash B$ whenever $[\Delta] \Vdash A$. We know, in particular that $A \vdash A$, by Identity, and thus $[A] \Vdash A$, so $[\Gamma, A] \Vdash B$. This means $\Gamma, A \vdash B$, so by Conditional Proof we can conclude that $\Gamma \vdash A \to B$, and thus $[\Gamma] \Vdash A \to B$, as required.

In the cases of 5-8, we must additionally show that it defines a persistent model. I.e., if $[\Gamma] \Vdash A$, then $[\Delta] \circ [\Gamma] \Vdash A$. If $[\Gamma] \Vdash A$, then $\Gamma \vdash A$, and so $\Delta, \Gamma \vdash A$, by Weakening, and finally $[\Delta, \Gamma] \Vdash A$ as required. □

B.5 Identity and associativity

Although I have not done so, it is possible to generalize our results to even weaker substructural type theories. Here I will offer only the briefest remarks about how this might be done. Just after our definition of a type assignment (Definition 10.2) we postulated two laws governing how they were combined: associativity $((\Gamma, \Delta), \Sigma) = (\Gamma, (\Delta, \Sigma))$ and Identity $(\epsilon, \Gamma) = \Gamma = (\Gamma, \epsilon)$. One could weaken these in various ways—a non-comprehensive list is provided in Section B.2 .

We could make corresponding changes to our definition a definability frame. For instance, we could replace the laws Unit and Associativity with weakenings corresponding to the above identifications. For instance, Unit Permutation and Unit Idempotence would correspond to the following frame conditions.

$0 \circ x = x \circ 0$ forall $x \in W$

$0 \circ 0 = 0$

It is a worthwhile exercise thinking through how our soundness and completeness results would generalize in this setting.

Notice that by identifying $((\Gamma, \Delta), \Sigma)$ with $(\Gamma, (\Delta, \Sigma))$ we license both directions of the following substructural inference:

$$\frac{((\Gamma, \Delta), \Sigma) \vdash A}{(\Gamma, (\Delta, \Sigma) \vdash A}$$

Another approach is to always treat $((\Gamma, \Delta), \Sigma)$ and $(\Gamma, (\Delta, \Sigma))$ as distinct premises, but put in primitive rules, of which the above are instances, allowing one to infer one sort of sequent from the other (even when embedded in larger contexts). This gives us more control, because one can accept one direction of these inferences but not the other. This is the approach taken in Restall (2000). The reader should consult this book for a more comprehensive treatment of substructural logics.

Let me finally (and briefly) talk about the eight systems without Identity that we have not talked about that are obtained from dropping Identity from the eight systems we considered earlier in this chapter. This includes the Curry-Howard analogue of the Structural Calculus, **ND**[], that we identified as being of particular interest for structured theories of propositions. The first thing to observe is that if we simply ran the above proof of completeness with one of these weaker systems it would break down. The point at which it breaks down is the crucial appeal to Identity in the part of the proof labeled 'right to left', showing that our definition $[\Gamma] \Vdash A \to B$ really satisfied the conditions for the conditional. I do not know of a natural semantics for propositional logic for which these systems are sound and complete.[4]

A fundamentally different way of preventing the Identity combinator, $\lambda x.x$, from being typed is to keep the Identity law, but modify the unit laws for the empty premise, ϵ. Note carefully that in the statement of Abstraction you need a type assignment Γ to the left of the variable you are abstracting. The following derivation typing the identity combinator makes this more explicit:

$$\dfrac{\dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\,\text{Identity}}{\epsilon, x : \sigma \vdash x : \sigma}\,\text{Left Unit}}{\epsilon \vdash \lambda x.x : \sigma \to \sigma}\,\text{Abstraction}$$

Thus even with Identity you cannot type the Identity combinator without the Left Unit principle. Indeed, it should be evident that it's not possible to type any combinators without either the Left and or Right Unit laws.

The system you get by weakening the unit laws for ϵ can be given sound and complete semantics within the definability semantics, unlike the systems that simply drop the Identity law.

But how do these two approaches to blocking the identity combinator relate? Consider **ND**[], the structural system without Identity, and the result of dropping the unit laws from **ND**[*I*] and replacing them with Unit Permutation and Unit Idempotence. Neither system has $\epsilon \vdash \lambda x.x : \sigma \to \sigma$ as a theorem, as we just observed. However, they clearly do not prove the same sequents—for instance, the result of dropping the unit laws, while keeping the Identity law has the sequent $x : \sigma \vdash x : \sigma$ as a theorem, whereas we clearly cannot derive this sequent in **ND**[].

But the differences run deeper than that. For instance, you can type λ-terms that abstract in predicating position from the weakened unit laws.

$$\dfrac{\dfrac{\dfrac{X : e \to t \vdash X : e \to t \qquad \epsilon \vdash a : e}{X, \epsilon \vdash Xa : t}\,\text{Generalized MP}}{\epsilon, X \vdash Xa : t}\,\text{Unit Permutation}}{\epsilon \vdash \lambda X.Xa : (e \to t) \to t}\,\text{Abstraction}$$

this is one of the terms that the system **ND**[] was designed to avoid typing: it corresponds to a relational diagram that's surround by a hole.

B.6 Definability structures for general λ-languages

Let us end by describing an interpretation of definability semantics that provides a natural class of applicative structures for interpreting substructurally defined general λ-languages. All of the definitions to follow work with an arbitrary definability frame $(W, \circ, 0)$, however to develop the interpretation properly it will be convenient to think of their elements more concretely along the lines of the definability interpretation mentioned earlier, i.e.:

> Members of W may be thought of as sets, multisets or sequences of simple constituents depending on the language type of theory we are modeling.
>
> $x \circ y$ should be read so the set or multiset union, or as sequence concatenation accordingly.

Thus, if we are modeling a view in which constituent occurrence number is well-defined but order is not, we might identify W with multisets of constituents.

Entities of a given type can then be organized according to which constituents they are built from.

> For each bundle of constituents $x \in W$, we can write A^{σ}_{x} to denote the set of entities of type σ that can be constructed from the constituents in x, (taking into account the number of times or the order of the constituents in x if applicable).

As before, there are two different ways of organizing entities, according to whether we are modeling views that allow for vacuous λ-abstraction or not (the substructural principle of Weakening). On the former interpretation we understand A^{σ}_{x} as the collection of entities that can be made using exactly the constituents in x (the right number of times/in the right order, accordingly). Compare with the notion of exact definability, $\rhd^{!}$, from Section 13.1. On the latter we interpret A^{σ}_{x} as the collection of entities that can be made using the constituents in x (the right number of times/in the right order, accordingly) without necessarily exhausting them. Compare with the notion of metaphysical definability, $\rhd$, from Section 13.1.

We'll call these mathematical objects 'organized sets'. In the first case, an organized set on $(W, \circ, 0)$ is simply a family of sets A_x indexed by the bundles constituents they are made from. In the latter case, we should have some sort of persistence condition: whenever $x \leq y$ (i.e. $y \in \{x\}\uparrow$), anything that can be metaphysically defined from x can also be metaphysically defined from y (i.e. from x and some unnecessary extra things). Thus we might expect that whenever $x \leq y$, every element of A_x corresponds to an element of A_y. In order to be sufficiently general in our representations we won't *identify* these elements, rather we will simply assume that whenever $x \leq y$ there is a mapping $i_{xy} : A_x \to A_y$ mapping each element in A_x representing something that can be defined from x to its corresponding representative in A_y. We should assume that these counterpart mappings compose in the obvious way: i_{xx} is the identity, if $x \leq y \leq z$ then $i_{xz} = i_{yz} \circ i_{xy}$. The reader should note that we have essentially just defined a modalized set over the Kripke frame $(W, \leq)$.

Note that we can think of the first interpretation as a special case of this one, except instead of interpreting $x \leq y$ as $y \in \{x\}\uparrow$ we interpret it as identity $x = y$, and the counterpart functions i_{xx} are simply the identity functions. To capture this generality it is convenient to allow the $\leq$ relation to be any pre-order on W that respects the monoid operation $\circ$.

Definition B.11 (Ordered frame)**.** *$(W, \circ, 0, \leq)$ is an ordered definability frame iff $(W, \leq, 0)$ is a frame, $\leq$ is a pre-order on W and:*

For any $x, y, z \in W$, if $x \leq y$ then $x \circ z \leq y \circ z$ and $z \circ x \leq z \circ y$.

The identity relation trivially satisfies this condition, and we have verified earlier that the relation $y \in \{x\}\uparrow$ does as well.

Definition B.12 (Organized set)**.** *Let $(W, \circ, 0, \leq)$ be an ordered definability frame. An organized set on this frame is a modalized set over the pointed Kripke frame $(W, \leq, 0)$.*

The notion of an organized set is a modalized set with some extra structure that has yet to do any work. The extra structure comes into play in defining the function space, which we will denote $A \Rightarrow B$. We will *not* define it as we did in Chapter 17. Our guiding idea here is the clause for the conditional in the definability semantics:

$x \Vdash A \to B$ iff for every $y \in W$, if $x \Vdash A$ then $x \circ y \Vdash B$.

An operation f in $(A \Rightarrow B)_x$ is intuitively an operation that can be built from the constituents x, so that when we apply it to any argument that can be built from y, the result can be built from $x \circ y$. Given a modalized set A, and $x \in W$ we will write $A_{x\circ -}$ for the modalized set B defined by $B_y = A_{x\circ y}$, and $i^B_{yz} := i^A_{(x\circ y)(x\circ z)}$.

Definition B.13 (Function space of organized sets)**.** *Given two organized sets, A and B, over $(W, \circ, 0, \leq)$ we define the function space as follows:*

- $(A \Rightarrow B)_x := \{f : A \to B_{x\circ -} \mid f$ *a homomorphism of modalized sets*$\}$.
- $(i_{xy} f)(z, a) = f(z, a)$.

Appealing to the definition of a homomorphism, this means that if $f \in (A \Rightarrow B)_x$ then:

For any $y \in W, f(y, \cdot) : A_y \to B_{x\circ y}$.

I.e. if f is definable from x and a from y then $f(y, a)$ is definable form $x \circ y$. We also have the following:

$$i_{yz}(f(y, a)) = f(z, i_{yz}a)$$

Thus, if I apply f at y to an element of A_y representing an entity built from y I should get the same result of applying f to this same entity's representative at z.

Definition B.14. *A definability structure on $(W, \circ, 0, \leq)$ is a modalized applicative structure (A, App) on the Kripke frame $(W, \leq)$ such that:*

- $A^{\sigma\to\tau} \leq A^\sigma \Rightarrow A^\tau$ *for every type σ and τ*
- $\mathrm{App}^{\sigma\tau}(x, (f, a)) = f(x, a)$.

A definability structure is full iff

$A^{\sigma\to\tau} = A^\sigma \Rightarrow A^\tau$ *for every type σ and τ*

We can spell out the sense in which a definability structure has combinators as follows.

Definition B.15. (A, App) *has:*

- *The I combinator iff there is an element* $i \in A_0^{\sigma\to\sigma}$ *such that for all* x, $i(x,a) = (x,a)$
- *The B combinator iff there is an element* $b \in A_0^{(\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$ *such that for all* $x,y,z \in W$, $b(x,f)(y,g)(z,a) = f(y \circ z, g(z,a))$
- *The C combinator iff there is an element* $c \in A_0^{(\sigma\to\tau\to\rho)\to\tau\to\sigma\to\rho}$ *such that for all* $x,y,z \in W$, $c(x,f)(y,a)(z,b) = f(z,b)(y,a)$
- *The W combinator iff there is an element* $w \in A_0^{(\sigma\to\sigma\to\tau)\to\sigma\to\tau}$ *such that for all* $x,y \in W$, $w(x,f)(y,a) = f(y,a)(y,a)$
- *The K combinator iff there is an element* $k \in A_0^{\sigma\to\tau\to\sigma}$ *such that for all* $x,y \in W$, $k(x,a)(y,b) = i_{x(x\circ y)}a$

Recall that we showed, in Exercise 17.6, that full modalized structures contain all the operations corresponding to combinators (we there showed S and K). By contrast, a full definability structure can lack certain combinators. By imposing various conditions on the definability frame $(W, \circ, 0)$ however, such as commutativity or idempotence, we can guarantee the presence of various combinators. The correspondence between these conditions on frames, and the relevant combinators should be familiar by now and are explored in the following exercise.

Exercise B.14. *Let* $(W, \circ, 0, \leq)$ *be an ordered frame and let* (A, App) *bet a full definability structure over this frame.*

a. Show that $i \in A_0^{\sigma\to\sigma}$ *and* $b \in A_0^{(\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho}$.

b. Show that if $(W, \circ, 0)$ *is commutative then then* $c \in A_0^{(\sigma\to\tau\to\rho)\to\tau\to\sigma\to\rho}$.

c. Show that if $(W, \circ, 0)$ *is idempotent then then* $w \in A_0^{(\sigma\to\sigma\to\tau)\to\sigma\to\tau}$

d. Show that if $x \leq y$ *iff* $y \in \{x\}\uparrow$, *then* $x \leq x \circ y$ *and that* $k \in A_0^{\sigma\to\tau\to\sigma}$ *is well-defined.*

Indeed, one can show that for any general λ-language defined by a substructural system $\mathbf{ND}[X]$, it is possible to interpret that language in definability structures whose frames validate the theorems of the propositional logic defined by $\mathbf{ND}[X]$ by extending an interpretation mapping each constant $c : \sigma$ to $[\![c]\!] \in \bigcup_{x\in W} A_x^\sigma$ to every term $M : \sigma$.

- $[\![c]\!] \in A_x^\sigma$ for some x, when $c : \sigma$
- $[\![x]\!]^g = g(x)$
- If $[\![M]\!]^g \in A_x^{\sigma\to\tau}$, $[\![N]\!]^g \in A_y^\sigma$, $[\![MN]\!]^g = [\![M]\!]^g(y, [\![N]\!]^g) \in A_{x\circ y}^\tau$.
- $[\![\lambda v.M]\!]^g = (x,a) \mapsto [\![M]\!]^{g[a/v]} \in A_x^{\sigma\to\tau}$ for some $x \in W$.

Finally, it's worth noting that there are many places in this book where we have used Kripke frames in the modeling of the full λ-language. This is not so surprising given the relationship between the full λ-language, intuitionistic logic and Kripke semantics. Often analogous techniques are possible for substructurally defined λ-languages by using definability frames instead. For example, there is a notion of a Kripke logical relation.

Definition B.16 (Definability logical relations)**.** *Let* $(W, \leq, 0)$ *be a definability frame, and* $\mathbf{A} = (A^{\cdot}, \mathrm{App}^{\cdot}_{A})$ *and* $\mathbf{B} = (B^{\cdot}, \mathrm{App}^{\cdot}_{M})$ *applicative structures. A definability logical relation between* $\mathbf{A}$ *and* $\mathbf{B}$ *is a family of relations* $R^{\sigma}_{x} \subseteq A^{\sigma} \times B^{\sigma}$ *for each type* σ *and* $x \in W$ *such that:*

$R^{\sigma\to\tau}_{x} fg$ *if and only if, for every* $y \in W$ *and* $a \in A^{\sigma}, b \in B^{\sigma}$*, if* $R^{\sigma}_{y} ab$ *then* $R^{\tau}_{x\circ y} \mathrm{App}^{\sigma\tau}_{A}(f, a)\, \mathrm{App}^{\sigma\tau}_{B}(g, b)$.

A logical relation is persistent iff

If $R^{\sigma}_{x} ab$ *and* $x \leq y$ *then* $R^{\sigma}_{y} ab$*, where* σ *is a base type.*

As with Kripke logical relations, definability logical relations can be used to characterize the λ-terms definable in various substructurally define λ-languages.[5]

Endnotes

1. See Urquhart (1972) and Fine (1974) for the origins of this work in the context of relevant logic.
2. See, for instance, Humberstone (1987), Mares (2004), and Fine (2017) respectively.
3. The reader should remember that $\leq$ as we have defined it is a pre-order and that one can have distinct elements x and y such that $x \leq y$ and $y \leq x$ so that Commutativity^{-} is genuinely weaker than Commutativity.
4. One could obtain completeness by having a wider conception of model. For instance, by imposing the constraint on the $\Vdash$ relation that $x \Vdash A \to B$ only if (not necessarily if) $x \circ y \Vdash B$ whenever $y \Vdash A$, as suggested earlier. However, this approach is unsatisfactory because we can no longer obtain the soundness of rules like Permutation by imposing conditions on frames—the constraint is consistent with all conditionals being false in a model—rather we have to simply restrict attention to models that satisfy Permutation to obtain a soundness result.
5. It is possible to prove very general results stating that a term is definable by a term of $\mathbf{ND}[X]$ iff it is invariant under every definability logical relation over frames for the propositional logic $\mathbf{ND}[X]$, although one needs a slightly more general notion of logical relation along the lines of Jung and Tiuryn (1993).

Hints for exercises

[i] **Hint:** Think about models where $[\![p]\!] = \{x\}$ $[\![q]\!] = \{x \circ x\}$ and consider the inference $p, p \vdash q$.

Bibliography

Peter Aczel. Frege structures and the notions of truth and proposition. In Jon Barwise, Howard Jerome Keisler, and Kenneth Kunen, editors, *The Kleene Symposium: Proceedings of the Symposium*, June 18–24, 1978, Madison, Wisconsin, 1980.

Alan R. Anderson and Nuel D. Belnap. *Entailment: The Logic of Relevance and Neccessity*, Vol. I. Princeton, NJ: Princeton University Press, 1975.

Peter B. Andrews. Resolution in type theory. *The Journal of Symbolic Logic*, 36(3):414–432, 1971.

Andrew Bacon. Radical anti-disquotationalism. *Philosophical Perspectives*, 32(1):41–107, 2018a. doi: 10.1111/phpe.12109.

Andrew Bacon. Tense and relativity. *Noûs*, 52(3):667–696, 2018b. doi: 10.1111/nous.12187.

Andrew Bacon. The broadest necessity. *Journal of Philosophical Logic*, 47(5):733–783, 2018c. doi: 10.1007/s10992-017-9447-9.

Andrew Bacon. *Vagueness and Thought*. Oxford, England: Oxford University Press, 2018d.

Andrew Bacon. Is reality fundamentally qualitative? *Philosophical Studies*, 176(1):259–295, 2019a. doi: 10.1007/s11098-017-1015-1.

Andrew Bacon. Substitution structures. *Journal of Philosophical Logic*, 48(6):1017–1075, 2019b. doi: 10.1007/s10992-019-09505-z.

Andrew Bacon. Logical combinatorialism. *Philosophical Review*, 129(4):537–589, 2020. doi: 10.1215/00318108-8540944.

Andrew Bacon. Opacity and paradox. In Carlo Nicolai and Johannes Stern, editors, *Modes of Truth*, pp. 231–265. London: Routledge, 2021.

Andrew Bacon. A case for higher-order metaphysics. In Peter Fritz and Nicholas K. Jones, editors, *Higher-Order Metaphysics*. Oxford: Oxford University Press, forthcoming.

Andrew Bacon. A theory of structured propositions. *The Philosophical Review*, 132 (2): 173–238, 2023.

Andrew Bacon and Cian Dorr. Classicism. In Peter Fritz and Nicholas K. Jones, editors, *Higher-Order Metaphysics*. Oxford: Oxford University Press, forthcoming.

Andrew Bacon and Kit Fine. The logic of logical necessity. In Yale Weiss and Romina Padró, editors, *Saul Kripke on Modal Logic*. Cham: Springer, 2023.

Andrew Bacon and Jeffrey Sanford Russell. The logic of opacity. *Philosophy and Phenomenological Research*, 99(1):81–114, 2019. doi: 10.1111/phpr.12454.

Andrew Bacon and Gabriel Uzquiano. Some results on the limits of thought. *Journal of Philosophical Logic*, 47(6):991–999, 2018. doi: 10.1007/s10992-018-9458-1.

Andrew Bacon and Jin Zeng. A theory of necessities. *Journal of Philosophical Logic*, 51(1): 151–199, 2022. doi: 10.1007/s10992-021-09617-5.

Andrew Bacon, John Hawthorne, and Gabriel Uzquiano. Higher-order free logic and the prior-kaplan paradox. *Canadian Journal of Philosophy*, 46(4-5):493–541, 2016. doi: 10.1080/00455091.2016.1201387.

Ralf M. Bader. Fundamentality and non-symmetric relations. In David Glick, George Darby, and Anna Marmodoro, editors, *The Foundation of Reality: Fundamentality, Space, and Time*, p. 15. Oxford: Oxford University Press, 2020.

Ruth C. Barcan. The identity of individuals in a strict functional calculus of second order. *Journal of Symbolic Logic*, 12(1):12–15, 1947. doi: 10.2307/2267171.

Henk Barendregt, Martin Bunder, and Wil Dekkers. Systems of illative combinatory logic complete for first-order propositional and predicate calculus. *The Journal of Symbolic Logic*, 58(3):769–788, 1993.

Chris Barker and Chung-Chieh Shan. *Continuations and Natural Language*. Oxford: Oxford University Press, 2014.

Elizabeth Barnes and J. Robert G. Williams. A theory of metaphysical indeterminacy. In Karen Bennett and Dean W. Zimmerman, editors, *Oxford Studies in Metaphysics*, vol. 6, pp. 103–148. Oxford: Oxford University Press, 2011.

John L. Bell. *Higher-Order Logic and Type Theory: Elements in Philosophy and Logic*. Cambridge: Cambridge University Press, 2022.

Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher-order semantics and extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004. doi: 10.2178/jsl/1102022211.

Paul Bernays and Moses Schönfinkel. Zum entscheidungsproblem der mathematischen logik. *Mathematische Annalen*, 99(1):342–372, 1928.

Max Black. The identity of indiscernibles. *Mind*, 61(242):153–164, 1952. doi: 10.1093/mind/LXI.242.153.

Bernard Bolzano. *Bernard Bolzanos Wissenschaftslehre in Vier Bänden*. Germany: F. Meiner, 1929.

George Boole. *The Mathematical Analysis of Logic*. New York: Philosophical Library, 1847.

George Boolos. To be is to be a value of a variable (or to be some values of some variables). *Journal of Philosophy*, 81(8):430–449, 1984. doi: jphil198481840.

George S. Boolos, John P. Burgess, and Richard C. Jeffrey. *Computability and Logic*. Cambridge, England: Cambridge University Press, 1974.

Francis Herbert Bradley. *Appearance and Reality: A Metaphysical Essay*. London: Routledge, 2016.

Chad E. Brown. *Automated Reasoning in Higher-Order Logic: Set Comprehension and Extensionality in Church's Type Theory*. Norcross, GA: College Publications, 2007.

Martin W. Bunder. Scott's models and illative combinatory logic. *Notre Dame Journal of Formal Logic*, 20(3):609–612, 1979.

Tyler Burge and Herbert Enderton. *The Collected Works of Alonzo Church*. Cambridge, MA: MIT Press, 2019.

John P. Burgess. *Fixing Frege*. Princeton: Princeton University Press, 2005.

John P. Burgess. On a derivation of the necessity of identity. *Synthese*, 191(7):1–19, 2014. doi: 10.1007/s11229-013-0351-8.

Michael Caie. Bunder's paradox. *Review of Symbolic Logic*, 13(4):829–844, 2020. doi: 10.1017/s1755020319000054.

Michael Caie, Jeremy Goodman, and Harvey Lederman. Classical opacity. *Philosophy and Phenomenological Research*, 101(3):524–566, 2020. doi: 10.1111/phpr.12587.

Felice Cardone and J. Roger Hindley. History of lambda-calculus and combinatory logic. *Handbook of the History of Logic*, 5:723–817, 2006.

Rudolf Carnap. Modalities and quantification. *Journal of Symbolic Logic*, 11(2):33–64, 1946. doi: 10.2307/2268610.

Rudolf Carnap. *Meaning and Necessity*. Chicago, IL: University of Chicago Press, 1947.

Bob Carpenter. *Type-Logical Semantics*. Cambridge, MA: MIT Press, 1997.

Roberto Casati and Achille C. Varzi. *Holes and Other Superficialities*. Cambridge, MA: MIT Press, 1994.

Hugh S. Chandler. Plantinga and the contingently possible. *Analysis*, 36(2):106–109, 1976. doi: 10.1093/analys/36.2.106.

Charles S. Chihara. *The Worlds of Possibility: Modal Realism and the Semantics of Modal Logic*. Oxford, England: Oxford University Press, 1998.

Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, vol. 33, pp. 346–366, 1932.

Alonzo Church. A set of postulates for the foundation of logic (part 2). *Annals of Mathematics*, 34(4): pp. 839–864, 1933.

Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5 (3):114–115, 1940. doi: 10.2307/2266866.

Alonzo Church. A formulation of the logic of sense and denotation. In Tyler Burge and Herbert Enderton, editors, *The Collected Works of Alonzo Church (2019)*, pp. 292–306. Cambridge, MA: MIT Press, 1951.

Alonzo Church. Comparison of russell's resolution of the semantical antinomies with that of tarski. *The Journal of Symbolic Logic*, 41(4):747–760, 1976.

John Corcoran and Alfred Tarski. What are logical notions? *History and Philosophy of Logic*, 7(2):143–154, 1986. doi: 10.1080/01445348608837096.

Max J. Cresswell. Propositional identity. *Logique Et Analyse*, 40:283–291, 1967.

Max J. Cresswell. Carnap and Mckinsey: Topics in the pre-history of possible-worlds semantics. In *Proceedings of the 12th Asian Logic Conference*, pp. 53–75. Singapore: World Scientific, 2013.

Max J. Cresswell and George Edward Hughes. *A New Introduction to Modal Logic*. London: Routledge, 1996.

John N. Crossley and I. Lloyd Humberstone. The logic of actually. *Journal of Symbolic Logic*, 46(2):424–426, 1981. doi: 10.2307/2273639.

Haskell Brooks Curry and Robert Feys. *Combinatory Logic*, vol. 1. Amsterdam: North-Holland, 1958.

Łukasz Czajka. Higher-order illative combinatory logic. *The Journal of Symbolic Logic*, 78 (3):837–872, 2013.

Scott Dixon. Directionalism and relations of arbitrary symmetry. *Dialectica*, forthcoming.

T. Scott Dixon. Plural slot theory. In Karen Bennett and Dean Zimmerman, editors, *Oxford Studies in Metaphysics*, vol. 11, pp. 193–223. Oxford: Oxford University Press, 2018.

Maureen Donnelly. Positionalism revisited. In Anna Marmodoro and David Yates, editors, *The Metaphysics of Relations*, pp. 80–99. Oxford: Oxford University Press, 2016.

Cian Dorr. Non-symmetric relations. In Dean Zimmerman, editor, *Oxford Studies in Metaphysics*, vol. 1, pp. 155–92. Oxford: Oxford University Press, 2004.

Cian Dorr. To be f is to be g. *Philosophical Perspectives*, 30(1):39–134, 2016. doi: 10.1111/phpe.12079.

Cian Dorr. Fundamentality : A compendium. handout.

Cian Dorr and Jeremy Goodman. Diamonds are forever. *Noûs*, 54(3):632–665, 2020. doi: 10.1111/nous.12271.

Cian Dorr and John Hawthorne. Naturalness. In Karen Bennett and Dean Zimmerman, editors, *Oxford Studies in Metaphysics*, vol. 8, p. 1. Oxford: Oxford University Press, 2013.

Cian Dorr, John Hawthorne, and Juhani Yli-Vakkuri. *The Bounds of Possibility: Puzzles of Modal Variation*. Oxford: Oxford University Press, 2021.

Maegan Fairchild. A paradox of matter and form. *Thought: A Journal of Philosophy*, 6(1): 33–42, 2017. doi: 10.1002/tht3.230.

Robert Feys. Peano et burali-forti précurseurs de la logique combinatoire. *Proceedings of the XIth International Congress of Philosophy*, 5:70–72, 1953.

Hartry Field. *Saving Truth From Paradox*. Oxford, England: Oxford University Press, 2008.

Kit Fine. Propositional quantifiers in modal logic. *Theoria*, 36(3):336–346, 1970. doi: 10.1111/j.1755-2567.1970.tb00432.x.

Kit Fine. Models for entailment. *Journal of Philosophical Logic*, 3(4):347–372, 1974. doi: 10.1007/bf00257480.

Kit Fine. Properties, propositions and sets. *Journal of Philosophical Logic*, 6(1):135–191, 1977. doi: 10.1007/bf00262054.

Kit Fine. Neutral relations. *Philosophical Review*, 109(1):1–33, 2000. doi: 10.1215/00318108-109-1-1.

Kit Fine. *The Limits of Abstraction*. Oxford, England: Oxford University Press, 2002a.

Kit Fine. Varieties of necessity. In Tamar Szabo Gendler and John Hawthorne, editors, *Conceivability and Possibility*, pp. 253–281. Oxford, England: Oxford University Press, 2002b.

Kit Fine. Some puzzles of ground. *Notre Dame Journal of Formal Logic*, 51(1):97–118, 2010. doi: 10.1215/00294527-2010-007.

Kit Fine. A difficulty for the possible worlds analysis of counterfactuals. *Synthese*, 189(1): 29–57, 2012a. doi: 10.1007/s11229-012-0094-y.

Kit Fine. Counterfactuals without possible worlds. *Journal of Philosophy*, 109(3):221–246, 2012b. doi: 10.5840/jphil201210938.

Kit Fine. Guide to ground. In Fabrice Correia and Benjamin Schnieder, editors, *Metaphysical Grounding*, pp. 37–80. Cambridge: Cambridge University Press, 2012c.

Kit Fine. A theory of truthmaker content i: Conjunction, disjunction and negation. *Journal of Philosophical Logic*, 46(6):625–674, 2017. doi: 10.1007/s10992-016-9413-y.

Kit Fine. Modal logic as metalogic. unpublished.

Frederic Brenton Fitch. A system of formal logic without an analogue to the curry w operator1. *The Journal of Symbolic Logic*, 1(3):92–100, 1936.

Frederic Brenton Fitch. *Elements of Combinatory Logic*. New Haven: Yale University Press, 1974.

Graeme Forbes. *The Metaphysics of Modality*. Oxford: Clarendon Press, 1985.

Gottleb Frege and Montgomery Furth. The basic laws of arithmetic. *British Journal for the Philosophy of Science*, 17(3):249–253, 1966.

Gottlob Frege. *Begriffsschrift: Eine der Arithmetischen Nachgebildete Formelsprache des Reinen Denkens*. Halle a.d.S.: Louis Nebert, 1879.

Gottlob Frege. *Grundgesetze der Arithmetik: begriffsschriftlich abgeleitet*, vol. 1. Pohle, Jena: Verlag Hermann Pohle, 1893.

Harvey M. Friedman. One hundred and two problems in mathematical logic. *Journal of Symbolic Logic*, 40(2):113–129, 1975a. doi: 10.2307/2271891.

Harvey M. Friedman. Equality between functionals. In *Logic Colloquium: Symposium on Logic*, Boston, MA, pp. 22–37. Springer, 1975b.

Harvey M. Friedman. Some systems of second order arithmetic and their use. In *Proceedings of the International Congress of Mathematicians (Vancouver, BC, 1974)*, vol. 1, pp. 235–242. Princeton, NJ: Citeseer, 1975c.

Harvey M. Friedman. A complete theory of everything: Satisfiability in the universal domain. MS.

Peter Fritz. Propositional contingentism. *Review of Symbolic Logic*, 9(1):123–142, 2016. doi: 10.1017/s1755020315000325.

Peter Fritz. Logics for propositional contingentism. *Review of Symbolic Logic*, 10(2):203–236, 2017a. doi: 10.1017/S1755020317000028.

Peter Fritz. A purely recombinatorial puzzle. *Noûs*, 51(3):547–564, 2017b. doi: 10.1111/nous.12172.

Peter Fritz. Higher-order contingentism, part 2: Patterns of indistinguishability. *Journal of Philosophical Logic*, 47(3):407–418, 2018a. doi: 10.1007/s10992-017-9432-3.

Peter Fritz. Higher-order contingentism, part 3: Expressive limitations. *Journal of Philosophical Logic*, 47(4):649–671, 2018b. doi: 10.1007/s10992-017-9443-0.

Peter Fritz. On higher-order logical grounds. *Analysis*, 80(4):656–666, 2020. doi: 10.1093/analys/anz085.

Peter Fritz. Ground and grain. *Philosophy and Phenomenological Research*, forthcoming a. doi: 10.1111/phpr.12822.

Peter Fritz. Operands and instances. *Review of Symbolic Logic*, pp. 1–22, forthcoming b. doi: 10.1017/s175502032100040x.

Peter Fritz and Jeremy Goodman. Higher-order contingentism, part 1: Closure and generation. *Journal of Philosophical Logic*, 45(6):645–695, 2016. doi: 10.1007/s10992-015-9388-0.

Peter Fritz and Jeremy Goodman. Counterfactuals and propositional contingentism. *Review of Symbolic Logic*, 10(3):509–529, 2017. doi: 10.1017/s1755020317000144.

Peter Fritz and Nicholas K. Jones. *Higher-Order Metaphysics*. Oxford: Oxford University Press, forthcoming.

Peter Fritz, Harvey Lederman, and Gabriel Uzquiano. Closed structure. *Journal of Philosophical Logic*, 50(6):1249–1291, 2021. doi: 10.1007/s10992-021-09598-5.

Daniel Gallin. *Intensional and Higher-Order Modal Logic: With Applications to Montague Semantics*. Amsterdam, Netherlands: American Elsevier Pub. Co., 1975.

Robin O. Gandy. On the axiom of extensionality, part I. *Journal of Symbolic Logic*, 21(1): 36–48, 1956. doi: 10.2307/2268484.

Cody Gilmore. Slots in universals. *Oxford Studies in Metaphysics*, 8:187–233, 2013.

Steven Givant and Paul Halmos. Introduction to boolean algebras (undergraduate texts in mathematics). *Bulletin of Symbolic Logic*, 16(2):281–282, 2010.

Kurt Gödel. On the completeness of the calculus of logic. In Kurt Godel, Solomon Feferman, John W. Dawson, Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean van Heijenoort, editors, *Kurt Gödel, K Collected Works*, vol. 1. Oxford: Oxford University Press, 1929.

Kurt Gödel. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. New York: Basic Books, 1962.

Robert Goldblatt. Topoi: The categorial analysis of logic. *British Journal for the Philosophy of Science*, 33(1): 95–97, (982).

Jeremy Goodman. An argument for necessitism. *Philosophical Perspectives*, 30(1):160–182, 2016. doi: 10.1111/phpe.12086.

Jeremy Goodman. Reality is not structured. *Analysis*, 77(1):43–53, 2017. doi: 10.1093/analys/anw002.

Jeremy Goodman. Agglomerative algebras. *Journal of Philosophical Logic*, 48(4):631–648, 2018a. doi: 10.1007/s10992-018-9488-8.

Jeremy Goodman. Counterpart models. unpublished note, 2018b.

Jeremy Goodman. Grounding generalizations, 2022.

Zachary Goodsell. Arithmetic is determinate. *Journal of Philosophical Logic*, 51(1):127–150, 2022. doi: 10.1007/s10992-021-09613-9.

Zachary Goodsell and Juhani Yli-Vakkuri. *Logical Foundations*. Unpublished.

Michael J. C. Gordon and Tom F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge: Cambridge University Press, 1993.

Dorothy Grover. *A Prosentential Theory of Truth*. Princeton, NJ: Princeton University Press, 1992.

Bob Hale. Absolute necessities. *Philosophical Perspectives*, 10:93–117, 1996.

John H. Harris. What's so logical about the 'logical' axioms? *Studia Logica*, 41(2–3):159–171, 1982. doi: 10.1007/BF00370342.

William S. Hatcher. *The Logical Foundations of Mathematics*. Oxford: Pergamon Press, 1982.

Allen Hazen. Expressive completeness in modal language. *Journal of Philosophical Logic*, 5(1):25–46, 1976. doi: 10.1007/BF00263656.

Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar*. Hoboken, NJ: Blackwell, 1998.

Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950. doi: 10.2307/2266967.

David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Berlin: Springer, 1928. Page numbers refer to the translation by Hammond Lewis M., Leckie George G., and Steinhardt F., 1999, American Mathematical Soc., vol. 69.

James Roger Hindley. *Basic Simple Type Theory*. Number 42. Cambridge: Cambridge University Press, 1997.

James Roger Hindley and Jonathan P Seldin. *Lambda-Calculus and Combinators, an Introduction*, vol. 2. Cambridge: Cambridge University Press, 2008.

Harold T. Hodes. Why ramify? *Notre Dame Journal of Formal Logic*, 56(2):379–415, 2015. doi: 10.1215/00294527-2864352.

Thomas Hofweber. *Ontology and the Ambitions of Metaphysics*. Oxford, England: Oxford University Press, 2016.

Ian Lloyd Humberstone. Operational semantics for positive "R". *Notre Dame Journal of Formal Logic*, 29(n/a):61–80, 1987. doi: 10.1305/ndjfl/1093637771.

Ian Lloyd Humberstone. *Philosophical Applications of Modal Logic*. Kings College London: College Publications, 2016.

Pauline Jacobson. Towards a variable-free semantics. *Linguistics and Philosophy*, 22(2): 117–185, 1999. doi: 10.1023/A:1005464228727.

Nicholas K. Jones. Nominalist realism. *Noûs*: 0–1, 2018. doi: 10.1111/nous.12193.

Achim Jung and Jerzy Tiuryn. A new characterization of lambda definability. In *International Conference on Typed Lambda Calculi and Applications*, pp. 245–257. Berlin/Heidelberg, Germany: Springer, 1993.

David Kaplan. On the logic of demonstratives. *Journal of Philosophical Logic*, 8(1):81–98, 1979. doi: 10.1007/BF00258420.

David Kaplan. A problem in possible worlds semantics. In Walter Sinnott-Armstrong, Diana Raffman, and Nicholas Asher, editors, *Modality, Morality and Belief: Essays in Honor of Ruth Barcan Marcus*, pp. 41–52. Cambridge: Cambridge University Press, 1995.

Hubert Kennedy. *Peano: Life and Works of Giuseppe Peano*, vol. 4. Berlin/Heidelberg, Germany: Springer Science & Business Media, 2012.

Jeffrey C. King. *The Nature and Structure of Content*. New York: Oxford University Press, 2007.

Jeffrey C. King. Structured propositions. In Edward N. Zalta, editor, *Stanford Encyclopedia of Philosophy*, 2008.

Kevin C. Klement. Russell-myhill paradox. In James Fieser, editor, *Internet Encyclopedia of Philosophy*, 2003.

Stephan Krämer. A simpler puzzle of ground. *Thought: A Journal of Philosophy*, 2(2):85–89, 2013. doi: 10.1002/tht3.77.

Georg Kreisel. Informal rigour and completeness proofs. In Imre Lakatos, editor, *Problems in the Philosophy of Mathematics*, pp. 138–157. Amsterdam: North-Holland, 1967.

Saul A. Kripke. Outline of a theory of truth. *Journal of Philosophy*, 72(19):690–716, 1975. doi: 10.2307/2024634.

Saul A. Kripke. *Naming and Necessity: Lectures Given to the Princeton University Philosophy Colloquium*. Cambridge, MA: Harvard University Press, 1980.

Saul A. Kripke. A puzzle about time and thought. In Saul A. Kripke, editor, *Philosophical Troubles*, Collected Papers vol. I. Oxford: Oxford University Press, 2011.

Saul A. Kripke. Fregean quantification theory. *Journal of Philosophical Logic*, 43(5):879–881, 2014. doi: 10.1007/s10992-013-9299-x.

Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958.

Joachim Lambek. Cartesian closed categories and typed λ-calculi. In *LITP Spring School on Theoretical Computer Science*, pp. 136–175. Berlin/Heidelberg, Germany: Springer, 1986.

Joachim Lambek and Philip J. Scott. *Introduction to Higher-Order Categorical Logic*, vol. 7. Cambridge: Cambridge University Press, 1988.

Edward John Lemmon. *An Introduction to Modal Logic: The Lemmon Notes*. Oxford, England: Blackwell, 1977.

Joop Leo. Modeling relations. *Journal of Philosophical Logic*, 37(4):353–385, 2008. doi: 10.1007/s10992-007-9076-9.

Joop Leo. Modeling occurrences of objects in relations. *Review of Symbolic Logic*, 3(1): 145–174, 2010. doi: 10.1017/s1755020309990347.

Azriel Levy. *Basic Set Theory*. New York: Dover Publications, 2012. ISBN 9780486150734. URL `https://books.google.com/books?id=zbGjAQAAQBAJ`.

Clarence Irving Lewis and Cooper Harold Langford. *Symbolic Logic*, vol. 170. New York: Dover Publications, 1959.

David K. Lewis. New work for a theory of universals. *Australasian Journal of Philosophy*, 61(4): 343-377, 1983. doi: 10.1080/00048408312341131.

David K. Lewis. *On the Plurality of Worlds*. Hoboken, NJ: Wiley-Blackwell, 1986.

David K. Lewis and Stephanie Lewis. Holes. *Australasian Journal of Philosophy*, 48(2): 206–212, 1970. doi: 10.1080/00048407012341181.

Øystein Linnebo. The potential hierarchy of sets. *Review of Symbolic Logic*, 6(2):205–228, 2013. doi: 10.1017/s1755020313000014.

Øystein Linnebo. *Thin Objects: An Abstractionist Account*. Oxford: Oxford University Press, 2018.

Øystein Linnebo and Agustin Rayo. Hierarchies ontological and ideological. *Mind*, 121 (482):269–308, 2012.

Bernard Linsky and Edward N. Zalta. In defense of the simplest quantified modal logic. *Philosophical Perspectives*, 8:431–458, 1994. doi: 10.2307/2214181.

Jon Erling Litland. Prospects for a theory of decycling. *Notre Dame Journal of Formal Logic*, 61(3):467–499, 2020. doi: 10.1215/00294527-2020-0016.

Jon Erling Litland. Collective abstraction. *Philosophical Review*, forthcoming.

Ralph Loader. The undecidability of λ-definability. In Charles Anthony Anderson and Michael Zelëny, editors, *Logic, Meaning and Computation*, pp. 331–342. Berlin/Heidelberg, Germany: Springer, 2001.

Fraser MacBride. Neutral relations revisited. *Dialectica*, 61(1):25–56, 2007. doi: 10.1111/j.1746-8361.2007.01092.x.

Saunders MacLane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Berlin/Heidelberg, Germany: Springer Science & Business Media, 2012.

Ofra Magidor. The last dogma of type confusions. *Proceedings of the Aristotelian Society*, 109(1pt1):1–29, 2009. doi: 10.1111/j.1467-9264.2009.00256.x.

Edwin D. Mares. *Relevant Logic: A Philosophical Interpretation*. Cambridge: Cambridge University Press, 2004.

Per Martin-Löf. An intuitionistic theory of types, 1972.

Ian McFetridge. Essay VIII. In John Haldane and Roger Scruton, editors, *Logical Necessity and Other Essays*, pp. viii–240. London: Aristotelian Society, 1990.

Vann McGee. How we learn mathematical language. *Philosophical Review*, 106(1):35–68, 1997. doi: 10.2307/2998341.

John C. C. McKinsey. On the syntactical construction of systems of modal logic. *Journal of Symbolic Logic*, 10(3):83–94, 1945. doi: 10.2307/2267027.

Michaela McSweeney. Following logical realism where it leads. *Philosophical Studies*, 176 (1):117–139, 2019. doi: 10.1007/s11098-017-1008-0.

Albert R. Meyer. What is a model of the lambda calculus? *Information and Control*, 52(1): 87–122, 1982.

Robert K. Meyer. On coherence in modal logics. *Logique Et Analyse*, 14:658–668, 1971.

John C. Mitchell. *Foundations for Programming Languages*, vol. 1. Cambridge, MA: MIT Press, 1996.

John C. Mitchell and Eugenio Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51(1-2):99–124, 1991.

Richard Montague. Set theory and higher-order logic. In J.N. Crossley, M.A.E. Dummett, editors, *Studies in Logic and the Foundations of Mathematics*, vol. 40, pp. 131–148. Amsterdam, Netherlands: Elsevier, 1965.

Richard Montague. The proper treatment of quantification in ordinary english. In Patrick Suppes, Julius Moravcsik, and Jaakko Hintikka, editors, *Approaches to Natural Language*, pp. 221–242. Netherlands: Springer, 1973.

Richard Montague. Reductions of higher-order logic. In J.W. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*, pp. 251–264. Amsterdam, Netherlands: Elsevier, 2014.

Gregory H. Moore. The emergence of first-order logic. *History and Philosophy of Modern Mathematics*, 11:95–135, 1988.

Michael Moortgat. *Categorial Investigations*. Berlin, Germany: De Gruyter, 2020.

Reinhard Muskens. Intensional models for the theory of types. *Journal of Symbolic Logic*, 72(1):98–118, 2007. doi: 10.2178/jsl/1174668386.

John Myhill. Problems arising in the formalization of intensional logic. *Logique Et Analyse*, 1(1):78–83, 1958. doi: 10.2307/2272577.

Daniel Nolan. Impossible worlds: A modest approach. *Notre Dame Journal of Formal Logic*, 38(4):535–572, 1997. doi: 10.1305/ndjfl/1039540769.

Steven Orey. Model theory for the higher order predicate calculus. *Transactions of the American Mathematical Society*, 92(1):72–84, 1959.

Francesco Orilia and Chris Swoyer. Properties. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2020 edition, 2020.

William Tuthill Parry. Modalities in the survey system of strict implication. *Journal of Symbolic Logic*, 4(4):137–154, 1939. doi: 10.2307/2268714.

Giuseppe Peano. *Arithmetices principia: Nova methodo exposita*. Rome: Fratres Bocca, 1889.

Alvin Plantinga. *The Nature of Necessity*. Oxford, England: Clarendon Press, 1974.

Gordon Plotkin. An illative theory of relations. *Situation Theory and Its Applications*, 22: 133–146, 1990.

Gordon D. Plotkin. Lambda-definability and logical relations. *Memorandum SAI-RM*, 4, 1973.

Gordon D. Plotkin. Lcf considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

Gordon D. Plotkin. Lambda-definability in the full type hierarchy. In John P. Seldin and John Roger Hindley, editors, *To HB Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 363–373. Cambridge, MA: Academic Press, 1980.

Emil L. Post. Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, 43(3):163–185, 1921.

Graham Priest. Intensional paradoxes. *Notre Dame Journal of Formal Logic*, 32(2):193–211, 1991. doi: 10.1305/ndjfl/1093635745.

Arthur Prior. Past, present, and future. *Revue Philosophique de la France Et de l'Etranger*, 157:476–476, 1967.

Arthur N. Prior. Modality and quantification in s5. *Journal of Symbolic Logic*, 21(1):60–62, 1956. doi: 10.2307/2268488.

Arthur N. Prior. On a family of paradoxes. *Notre Dame Journal of Formal Logic*, 2(1): 16–32, 1961. doi: 10.1305/ndjfl/1093956750.

Arthur N. Prior. Formal logic. *Studia Logica*, 15:298–301, 1964.

Arthur N. Prior. *Objects of Thought*. Oxford, England: Clarendon Press, 1971.

Arthur N. Prior and Kit Fine. Times, worlds and selves. *Synthese*, 40(2):389–408, 1979.

Tadeusz Prucnal. On two problems of harvey friedman. *Studia Logica*, 38(3):247–262, 1979. doi: 10.1007/BF00405383.

Willard Van Orman Quine. Variables explained away. *Journal of Symbolic Logic*, 32(1): 112–112, 1967. doi: 10.2307/2271258.

Willard Van Orman Quine. *Philosophy of Logic*. Cambridge, MA: Harvard University Press, 1970.

Frank P. Ramsey. Facts and propositions. *Aristotelian Society Supplementary Volume*, 7(1): 153–170, 1927.

William J. Rapaport, Nicholas M. Asher, and Johan A. W. Kamp. The knower's paradox and representational theories of attitudes. *Journal of Symbolic Logic*, 53(2):666, 1988. doi: 10.2307/2274549.

Agustín Rayo and Gabriel Uzquiano. *Absolute Generality*. Oxford and New York: Oxford University Press, 2006.

Agustin Rayo and Timothy Williamson. A completeness theorem for unrestricted first-order languages. In Jc Beall, editor, *Liars and Heaps*, pp. 331–356. Oxford: Oxford University Press, 2003.

Augustín Rayo and Gabriel Uzquiano. Toward a theory of second-order consequence. *Notre Dame Journal of Formal Logic*, 40(3):315–325, 1999. doi: 10.1305/ndjfl/1022615612.

Greg Restall. *An Introduction to Substructural Logics*. London: Routledge, 2000.

Alexander Roberts. Necessity in the highest degree. MS.

Ian Rumfitt. Logical necessity. In Bob Hale and Aviv Hoffmann, editors, *Modality: Metaphysics, Logic, and Epistemology*, pp. 35–64. Oxford: Oxford University Press, 2010.

Bertrand Russell. The principles of mathematics. *Revue de Métaphysique et de Morale*, 11 (4):11–12, 1903.

Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30(3):222–262, 1908. doi: 10.2307/2272708.

Bertrand Russell. *The Philosophy of Logical Atomism*. London: Routledge, 1918. Paginations refer to the 2010 reprint.

Bertrand Arthur William Russell. *Introduction to Mathematical Philosophy*. London, England: Dover Publications, 1919.

Berit Brogaard and Joe Salerno. Why Counterpossibles Are Non-Trivial. In Vincent Hendricks, editor, *Synthese volume*, forthcoming.

Nathan Salmon. The logic of what might have been. *Philosophical Review*, 98(1):3–34, 1989. doi: 10.2307/2185369.

Chris Scambler. Can all things be counted? *Journal of Philosophical Logic*, 50(5):1079–1106, 2021. doi: 10.1007/s10992-021-09593-w.

Barry Schein. *Plurals and Events*. Cambridge, MA: MIT Press, 1993.

Georg Schiemer and Erich H. Reck. Logic in the 1930s: Type theory and model theory. *Bulletin of Symbolic Logic*, 19(4):433–472, 2013. doi: 10.2178/bsl.1904010.

Moses Schönfinkel. Über die bausteine der mathematischen logik. *Mathematische Annalen*, 92(3):305–316, 1924.

Dana Scott. Combinators and classes. In *International Symposium on Lambda-Calculus and Computer Science Theory*, pp. 1–26. Berlin/Heidelberg, Germany: Springer, 1975.

Jonathan P. Seldin. Lambda-calculus and functional programming, 2000.

Stewart Shapiro. Principles of reflection and second-order logic. *Journal of Philosophical Logic*, 16(3):309–333, 1987. doi: 10.1007/BF00152934.

Stewart Shapiro. *Foundations Without Foundationalism: A Case for Second-Order Logic*. Oxford, England: Oxford University Press, 1991.

Theodore Sider. *Logic for Philosophy*. Oxford, England: Oxford University Press, 2010.

Theodore Sider. *Writing the Book of the World*. Oxford, England: Oxford University Press, 2011.

Stephen G. Simpson. *Subsystems of Second Order Arithmetic*, vol. 1. Cambridge: Cambridge University Press, 2009.

Scott Soames. Direct reference, propositional attitudes, and semantic content. *Philosophical Topics*, 15(1):47–87, 1987. doi: 10.5840/philtopics198715112.

Robert Stalnaker. Possible worlds. *Noûs*, 10(1):65–75, 1976. doi: 10.2307/2214477.

Robert Stalnaker. Merely possible propositions. In Bob Hale and Aviv Hoffmann, editors, *Modality: Metaphysics, Logic, and Epistemology*, pp. 21–32. Oxford University Press, 2010.

Robert Stalnaker. *Mere Possibilities: Metaphysical Foundations of Modal Semantics*. Princeton, NJ: Princeton University Press, 2012.

James P. Studd. The iterative conception of set: A (bi-)modal axiomatisation. *Journal of Philosophical Logic*, 42(5):1–29, 2013. doi: 10.1007/s10992-012-9245-3.

Roman Suszko. Identity connective and modality. *Studia Logica*, 27(1):7–39, 1971. doi: 10.1007/BF02282541.

Alfred Tarski. Sur les ensembles définissables de nombres réels. *Fundamenta Mathematicae*, 17(1):210–239, 1931. URL `http://eudml.org/doc/212515`.

Alfred Tarski. The concept of truth in formalized languages. In Alfred Tarski, editor, *Logic, Semantics, Metamathematics*, pp. 152–278. Oxford: Oxford University Press, 1936a.

Alfred Tarski. On the concept of logical consequence. *Logic, Semantics, Metamathematics*, 52:409–420, 1936b.

Pavel Tichý. *The Foundations of Frege's Logic*. Berlin and New York: De Gruyter, 1988.

Robert Trueman. *Properties and Propositions: The Metaphysics of Higher-Order Logic*. Cambridge: Cambridge University Press, 2020.

Dustin Tucker and Richmond H. Thomason. Paradoxes of intensionality. *Review of Symbolic Logic*, 4(3):394–411, 2011. doi: 10.1017/s1755020311000128.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

Alasdair Urquhart. Semantics for relevant logics. *Journal of Symbolic Logic*, 37(1):159–169, 1972. doi: 10.2307/2272559.

Gabriel Uzquiano. Models of second-order zermelo set theory. *Bulletin of Symbolic Logic*, 5(3):289–302, 1999. doi: 10.2307/421182.

Gabriel Uzquiano. A neglected resolution of russell's paradox of propositions. *Review of Symbolic Logic*, 8(2):328–344, 2015. doi: 10.1017/s1755020315000106.

Gabriel Uzquiano. Elusive propositions. *Journal of Philosophical Logic*, 50(4):705–725, 2021. doi: 10.1007/s10992-020-09582-5.

Gabriel Uzquiano. Ramified structure. *Philosophical Studies*, pp. 1–24, forthcoming. doi: 10.1007/s11098-022-01780-y.

Jouko Väänänen. Second order logic, set theory and foundations of mathematics. In P. Dybjer, Sten Lindström, Erik Palmgren, G. Sundholm, editors, *Epistemology versus Ontology*, pp. 371–380. Berlin/Heidelberg, Germany: Springer, 2012.

Alfred North Whitehead and Bertrand Russell. *Principia Mathematica by Alfred North Whitehead and Bertrand Russell*. Cambridge: Cambridge University Press, 1910-1913.

Isaac Wilhelm. Talk about types. 2022.

Timothy Williamson. Converse relations. *Philosophical Review*, 94(2):249–262, 1985. doi: 10.2307/2185430.

Timothy Williamson. Everything. *Philosophical Perspectives*, 17(1):415–465, 2003. doi: 10.1111/j.1520-8583.2003.00017.x.

Timothy Williamson. *Modal Logic as Metaphysics*. Oxford, England: Oxford University Press, 2013.

Timothy Williamson. Modal science. *Canadian Journal of Philosophy*, 46(4-5):453–492, 2016. doi: 10.1080/00455091.2016.1205851.

Timothy Williamson. Metametaphysics and semantics. *Metaphilosophy*, forthcoming. doi: 10.1111/meta.12528.

Kwasi Wiredu. On the necessity of s4. *Notre Dame Journal of Formal Logic*, 20(n/a):689, 1979. doi: 10.1305/ndjfl/1093882679.

Ludwig Wittgenstein. *Tractatus Logico-Philosophicus (Trans. Pears and McGuinness)*. London: Kegan Paul, Trench, Trubner and co. ltd, 1922.

Stephen Wolfram. Where did combinators come from? hunting the story of moses sch\" onfinkel. *arXiv preprint arXiv:2108.08707*, 2021.

Crispin Wright and Bob Hale. *The Reason's Proper Study: Essays Towards a Neo-Fregean Philosophy of Mathematics*. Oxford: Clarendon Press, 2001.

Juhani Yli-Vakkuri and John Hawthorne. Intensionalism and propositional. In: Uriah Kriegel, editor, *Oxford Studies in Philosophy of Mind*, vol. 2, p. 114. Oxford: Oxford University Press, 2021.

Jin Zeng. Grounding and granularity. MS.

Ernst Zermelo. Untersuchungen über die grundlagen der mengenlehre. i. *Mathematische Annalen*, 65(2):261–281, 1908.

Index