# FUNCTIONALITY, POLYMORPHISM, AND CONCURRENCY:

# A MATHEMATICAL INVESTIGATION OF PROGRAMMING PARADIGMS

Peter Selinger

A Dissertation in Mathematics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1997

Supervisor of Dissertation

Graduate Group Chairperson

*In Memory of Moez Alimohamed*

*1967–1994*

# Acknowledgments

This research was supported by graduate fellowships from the Institute for Research in Cognitive Science and from the School of Arts and Sciences at the University of Pennsylvania, and by an Alfred P. Sloan Doctoral Dissertation Fellowship.

Some of the results in Chapters 3 and 4 were announced in a paper that appeared in the Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science [57]. Parts of Chapter 6 are scheduled to appear in the Proceedings of CONCUR '97 [58].

Part of this research was done while I was visiting the Isaac Newton Institute at the University of Cambridge as an affiliated participant of the Semantics of Computation Program in the Fall of 1995.

I am grateful to Gordon Plotkin for introducing me to the problem of partial orders on term models, and for many stimulating and enjoyable discussions on the lambda calculus. He has kindly consented to the inclusion of his previously unpublished proof of Theorem 3.16, as well as the material in Section 3.4, some of which resulted from his discussions with Alex Simpson.

I would also like to thank Peter Freyd, Furio Honsell, Martin Hyland, Catuscia Palamidessi, Benjamin Pierce, Simona Ronchi, Davide Sangiorgi, Dana Scott, Phil Scott, and Glynn Winskel. They all have contributed valuable comments on parts of this work at various stages of its completion.

Thanks to Peter Freyd, David Harbater, Andre Scedrov, and Scott Weinstein for serving on my Ph.D. committee.

Thanks to the staff at the Math Department at the University of Pennsylvania for helping me time and again with the administrative aspects of being a graduate student.

Thanks to Jeny Carden, for her love and support.

Finally, I would like to express my special gratitude to my advisor Andre Scedrov for his guidance, support, candid criticism, and for his faith in my ability to complete this task.

Philadelphia, June 1997

ABSTRACT

FUNCTIONALITY, POLYMORPHISM, AND CONCURRENCY:

A MATHEMATICAL INVESTIGATION OF PROGRAMMING PARADIGMS

Peter Selinger

Andre Scedrov

The search for mathematical models of computational phenomena often leads to problems that are of independent mathematical interest. Selected problems of this kind are investigated in this thesis. First, we study models of the untyped lambda calculus. Although many familiar models are constructed by order-theoretic methods, it is also known that there are some models of the lambda calculus that cannot be non-trivially ordered. We show that the standard open and closed term algebras are unorderable. We characterize the absolutely unorderable $\mathbf{T}$-algebras in any algebraic variety $\mathbf{T}$. Here an algebra is called absolutely unorderable if it cannot be embedded in an orderable algebra. We then introduce a notion of finite models for the lambda calculus, contrasting the known fact that models of the lambda calculus, in the traditional sense, are always non-recursive. Our finite models are based on Plotkin's syntactical models of reduction. We give a method for constructing such models, and some examples that show how finite models can yield useful information about terms. Next, we study models of typed lambda calculi. Models of the polymorphic lambda calculus can be divided into environment-style models, such as Bruce and Meyer's non-strict set-theoretic models, and categorical models, such as Seely's interpretation in *PL*-categories. Reynolds has shown that there are no set-theoretic strict models. Following a different approach, we investigate a notion of non-strict categorical models. These provide a uniform framework in which one can describe various classes of non-strict models, including set-theoretic models with or without empty types, and Kripke-style models. We show that completeness theorems correspond to categorical representation theorems, and we reprove a completeness result by Meyer *et al.* on set-theoretic models of the simply-typed lambda calculus with possibly empty types. Finally, we study properties of asynchronous communication in networks of communicating processes. We formalize several notions of asynchrony independently of any particular concurrent process paradigm. A process is asynchronous if its input and/or output is filtered through a communication medium, such as a buffer or a queue, possibly with feedback. We prove that the behavior of asynchronous processes can be equivalently characterized by first-order axioms.

# Contents


**Introduction** **1**

**1 Preliminaries** **5**

1.1 Basic category theory . . . 5
1.1.1 Categories . . . 5
1.1.2 Functors . . . 6
1.1.3 Natural transformations . . . 7
1.1.4 Adjunctions . . . 7
1.1.5 Limits and colimits . . . 7
1.1.6 Cartesian-closed categories . . . 8
1.2 Basic domain theory . . . 10
1.2.1 Preorders and posets . . . 10
1.2.2 Complete partial orders . . . 11
1.2.3 Bounded complete partial orders . . . 11
1.2.4 Stability . . . 11
1.2.5 Domain equations . . . 12
1.2.6 The $D_\infty$-construction . . . 13
1.3 Basic universal algebra . . . 13
1.3.1 $\Sigma$-algebras . . . 13
1.3.2 Term algebras . . . 14
1.3.3 Algebraic varieties . . . 14
1.3.4 Indeterminates . . . 15
1.3.5 Ordered algebras . . . 16
1.3.6 Dcpo-algebras . . . 17

**2 The Lambda Calculus is Algebraic** **19**

2.1 The lambda calculus . . . 20
2.1.1 Lambda conversion . . . 20
2.1.2 Lambda reduction and consistency . . . 21
2.2 Combinatory models of the lambda calculus . . . 22
2.2.1 Combinatory algebras and combinatory logic . . . 22
2.2.2 The derived lambda abstractor . . . 23
2.2.3 The local interpretation of lambda terms . . . 23
2.2.4 Lambda algebras . . . 24
2.3 Lambda algebras and indeterminates . . . 25
2.3.1 A characterization of $\mathbf{A}[x]$ for lambda algebras . . . 25
2.3.2 The absolute interpretation . . . 26
2.3.3 Soundness and completeness for lambda algebras . . . 28
2.4 Lambda theories and lambda algebras form equivalent categories . . . 28
2.5 Lambda models . . . 29

2.6 Models of the lambda-$\beta\eta$-calculus . . . 30
2.6.1 Curry algebras . . . 30
2.6.2 Extensional models . . . 30
2.7 Lambda algebras and categorical models . . . 30
2.7.1 Reflexive ccc models . . . 30
2.7.2 Reflexive ccc models and lambda algebras . . . 31

**3 Unorderability** **35**
3.1 Lambda terms cannot be ordered . . . 36
3.1.1 Plotkin's unorderable algebra: Separability . . . 36
3.1.2 The standard term algebras are unorderable . . . 36
3.2 The Topological Completeness Problem . . . 37
3.3 A characterization of absolutely unorderable algebras . . . 39
3.3.1 Absolutely unorderable algebras and generalized Mal'cev operators . . . 39
3.3.2 An application to ordered algebras and dcpo-algebras . . . 40
3.4 Absolutely unorderable combinatory algebras . . . 41
3.5 Relating different notions of unorderability . . . 43
3.5.1 Local notions . . . 43
3.5.2 Absolute notions . . . 44

**4 Finite Lambda Models** **47**
4.1 Models of reduction . . . 47
4.1.1 Syntactical models of reduction . . . 47
4.1.2 Categorical models of reduction . . . 48
4.1.3 Models of $\beta\eta$-reduction: Order-extensionality . . . 49
4.2 Tree models . . . 49
4.2.1 Recapturing convertibility . . . 49
4.2.2 A method for constructing models . . . 50
4.3 Partial models . . . 51
4.4 Examples . . . 52
4.4.1 A class of finite models to distinguish the terms $\Omega_n$ . . . 52
4.4.2 A non-trivial 3-element model . . . 53
4.5 Completeness . . . 54
4.6 Relating models of reduction to $D_\infty$-models . . . 54

**5 Henkin Representations, Polymorphism, and Empty Types** **57**
5.1 Henkin representations of cartesian-closed categories . . . 58
5.1.1 Henkin representations . . . 58
5.1.2 Henkin representations and well-pointed ccc's . . . 59
5.1.3 Freely adjoining arrows to a ccc . . . 60
5.1.4 Henkin representation theorems . . . 62
5.2 The interpretation of the simply-typed lambda calculus . . . 64
5.2.1 The simply-typed lambda calculus . . . 64
5.2.2 Strict interpretation in a cartesian-closed category . . . 65
5.2.3 The cartesian-closed category associated to a theory . . . 67
5.2.4 Henkin representations of a free ccc . . . 67
5.2.5 The non-strict interpretation of the simply-typed lambda calculus . . . 67
5.3 From Henkin representation theorems to completeness theorems . . . 68
5.3.1 The problem with empty types . . . 68
5.3.2 A categorical analysis of the rule (*non-empty*) . . . 69
5.3.3 Set-theoretic models with non-empty types . . . 69
5.3.4 Set-theoretic models with empty types . . . 69

5.3.5 Kripke lambda models . . . 72
5.3.6 A remark on the principal model property . . . 72
5.4 Henkin representations of *PL*-categories . . . 72
5.4.1 *PL*-categories . . . 72
5.4.2 Henkin-*PL*-representations . . . 74
5.4.3 Standard structures . . . 75
5.4.4 Freely adjoining arrows to the base of a *PL*-category . . . 76
5.4.5 Henkin-*PL*-representation theorems . . . 77
5.5 The interpretation of the polymorphic lambda calculus . . . 78
5.5.1 The polymorphic lambda calculus . . . 78
5.5.2 Strict interpretation in a *PL*-category . . . 79
5.5.3 The *PL*-category associated to a theory . . . 81
5.5.4 The non-strict interpretation of the polymorphic lambda calculus . . . 81
5.6 From Henkin-*PL*-representation theorems to polymorphic completeness theorems . . . 82
5.6.1 Set-theoretic models with non-empty types . . . 82
5.6.2 Polymorphic Kripke models . . . 82

**6 First-Order Axioms for Asynchrony** **83**
6.1 An elementary definition of asynchrony . . . 83
6.1.1 Labeled transition systems and bisimulation . . . 84
6.1.2 Input, output and sequential composition . . . 85
6.1.3 Buffers and queues . . . 87
6.1.4 Notions of asynchrony . . . 87
6.1.5 Examples . . . 88
6.2 First-order axioms for asynchrony . . . 89
6.2.1 Out-buffered agents . . . 89
6.2.2 In-buffered agents . . . 92
6.2.3 Out-queued and in-queued agents . . . 93
6.3 More agent constructors and asynchrony with feedback . . . 93
6.3.1 Some operations on agents . . . 93
6.3.2 Asynchrony with feedback . . . 94
6.4 Example: Asynchronous CCS . . . 96
6.5 Example: The core join calculus . . . 99
6.6 Other characterizations of asynchrony . . . 100
6.6.1 Out-buffered agents . . . 100
6.6.2 In-buffered agents . . . 102
6.6.3 Out-queued and in-queued agents . . . 104

# List of Tables


1.1 Some posets . . . 10
1.2 Equational rules for $\Sigma$-algebras . . . 15
1.3 Inequational rules for $\Sigma$-algebras . . . 16

2.1 The axioms and rules of the lambda calculus . . . 20
2.2 Reduction rules of the lambda calculus . . . 21
2.3 The axioms and rules of combinatory logic . . . 22

4.1 Multiplication table for a partial model . . . 53
4.2 Values for $\psi(c, b, a)$ and $\mathrm{k} \cdot c \cdot b \cdot a$ . . . 54

5.1 Typing rules for the simply-typed lambda calculus . . . 64
5.2 Equational rules for the simply-typed lambda calculus . . . 65
5.3 Rules for the simply-typed lambda calculus with emptiness assertions . . . 70
5.4 Typing rules for the polymorphic lambda calculus . . . 78
5.5 Equational rules for the polymorphic lambda calculus . . . 78

6.1 First-order axioms for out-buffered agents . . . 89
6.2 First-order axioms for in-buffered agents . . . 90
6.3 First-order axioms for out-queued agents . . . 90
6.4 First-order axioms for in-queued agents . . . 90
6.5 First-order axioms for out-buffered agents with feedback . . . 95
6.6 Transitions rules for asynchronous CCS . . . 97
6.7 Transitions rules for the core join calculus . . . 99
6.8 Second-order axioms for out-buffered agents . . . 101
6.9 Second-order axioms for in-buffered agents . . . 103
6.10 Second-order axioms for out-queued agents . . . 103
6.11 Second-order axioms for in-queued agents . . . 103

# Introduction

The central aim in giving mathematical meaning to computer programs is to represent *computational objects*, such as procedures, data types, or communication channels, by *mathematical objects*, such as functions, sets, or more generally, points in suitable mathematical spaces. Often, one begins with an idealized programming language, such as the lambda calculus or Milner's calculus of communicating systems, and then seeks to find a mathematical model that reflects the relevant computational properties. The search for such models is guided by computational as well as mathematical intuitions, and it often leads to problems that are of independent mathematical interest. Some selected problems of this kind are investigated in this dissertation.

The first part of this thesis is devoted to the model theory of the untyped lambda calculus. D. Scott discovered in the late 1960's that models of the untyped lambda calculus can be constructed by a combination of order-theoretic and topological methods. Scott's methods have been widely studied and adapted to numerous situations, and today one can choose from a wide array of model constructions that are based on Scott's principles. On the other hand, there are results that indicate that Scott's methods may not in general be complete: Honsell and Ronchi Della Rocca [27] have shown that there exists a lambda theory that does not arise as the theory of a reflexive model in the cartesian-closed category of complete partial orders and Scott-continuous functions. Moreover, there are properties that one may desire in a model, but that are incompatible with the presence of a partial order: for instance, Plotkin [50] has recently shown that there exists an extensional lambda algebra which is *finitely separable*. By definition, a lambda algebra $X$ is finitely separable if for every finite subset $A \subseteq X$ and for every function $f : A \to X$, there exists an element $\hat{f} \in X$ such that $\hat{f} \cdot x = f(x)$ for all $x \in X$. It is not hard to see that a finitely separable algebra cannot be non-trivially partially ordered in a way such that the order is compatible with the algebra structure.

In general, we define a lambda algebra $X$ to be *unorderable* if there does not exist a non-trivial partial order on $X$ for which the application operation is monotone. Our first main result is the following: The standard open and closed term algebras of the $\lambda\beta$- and $\lambda\beta\eta$-calculi are unorderable. Recall that the standard term algebras are just made up from lambda terms, taken up to $\beta$- or $\beta\eta$-equivalence. The unorderability of the standard term algebras is a surprising fact, because the algebras that were previously known to be unorderable, such as Plotkin's finitely separable algebra, require a much more delicate syntactic construction. As a consequence of this result, it follows that if a partially ordered model of the untyped lambda calculus is complete for one of the theories $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$, then the denotations of closed terms in that model are pairwise incomparable, *i.e.* the term denotations form an anti-chain.

Closely related to the question of unorderability is the question of *order-incompleteness*: does there exist a lambda theory that does not arise as the theory of a non-trivially partially ordered model? Or, expressed in terms of algebras: does there exist a lambda algebra which cannot be embedded in an orderable one? We call such an algebra *absolutely unorderable*. The concept of absolute unorderability can be formulated in any algebraic variety $\mathbf{T}$, and our second main result is a theorem in universal algebra: In any algebraic variety $\mathbf{T}$, an algebra $\mathbf{A}$ is absolutely unorderable if and only if, for some $n \geqslant 1$, there exist polynomials $M_1, \ldots, M_n \in \mathbf{A}[x_1, x_2, x_3]$ such that the equations $t = M_1(t, u, u)$, $M_i(t, t, u) = M_{i+1}(t, u, u)$ for $1 \leqslant i < n$, and $M_n(t, t, u) = u$ hold in $\mathbf{A}[u, t]$. Operators $M_1, \ldots, M_n$ satisfying this condition are called *generalized Mal'cev operators*. Such operators were first used by Hagemann and Mitschke [25] to characterize varieties with $n$-permutable congruences. The connection to unorderability was first noticed by Taylor [63, 11], who proved that algebras in a variety with $n$-permutable congruences are unorderable; the converse is a new result.

As a consequence, the question of order-incompleteness for the untyped lambda calculus has been reduced to the question whether it is consistent, for some $n \geqslant 1$, to add generalized Mal'cev operators $M_1, \dots, M_n$ to the lambda calculus. It was proved by Plotkin and Simpson that a Mal'cev operator is inconsistent with the lambda calculus for $n = 1$. Later, Plotkin and myself showed that it is also inconsistent for $n = 2$. In the remaining cases, the answer is not known.

We continue our investigation of models of the untyped lambda calculus by introducing a notion of *finite lambda models*. These models provide a tool for predicting the evaluation behavior of a lambda term by finitary means. This yields a novel proof method for proving inequalities of untyped lambda terms. Finite models differ from traditional models of the untyped lambda calculus, which are always infinite and in fact never even recursive, in that they are models of *reduction*, rather than models of *conversion*. This means that they are equipped with a partial order and a soundness property of the form $M \longrightarrow N \Rightarrow [\![M]\!] \leqslant [\![N]\!]$, where $\longrightarrow$ denotes either $\beta$- or $\beta\eta$-reduction. Models of reduction were considered by several authors [23, 30, 49], and we use a formulation which was given by Plotkin [49] in the spirit of the familiar syntactical lambda models [5]. We focus on practical methods of constructing such models, and we show in two examples that, despite their simplicity, finite models can yield useful information about lambda terms.

The second part of this thesis is devoted to models of the simply-typed and the polymorphic lambda calculus. The models in the literature follow one of two basic designs: set-theoretic environment-style models, such as Henkin models for the simply-typed lambda calculus [21] or Bruce-Meyer models for polymorphism [10], and categorical models, such as the interpretation of the simply-typed calculus in a cartesian closed category [33] or of the polymorphic calculus in a *PL*-category [56]. Environment-style models are typically *non-strict*, in the sense that a function type $\sigma \to \tau$ is interpreted as a *subset* of the set of functions from $\sigma$ to $\tau$. On the other hand, categorical models are always strict.

Reynolds has shown that there are no strict set-theoretic models of the polymorphic lambda calculus [52]. Here, we take the opposite approach and consider non-strict categorical models. This generalizes both environment-style models and strict categorical models. The central concept is that of a *Henkin representation*: a functor $H$ between cartesian-closed categories that preserves finite products, such that for all objects $A, B$, the canonical morphism $H(B^A) \to H(B)^{H(A)}$ is monic. Henkin representations provide a uniform framework in which one can describe various classes of non-strict models, including set-theoretic models with possibly empty types [39], set-theoretic models with non-empty types [21], and Kripke-style models [42]. We show that completeness theorems for each of these classes of models correspond naturally to categorical Henkin representation theorems. One such Henkin representation theorem characterizes those cartesian-closed categories that can be Henkin-embedded in the category of sets: we show that this is the case if and only if every object $A$ is either partially initial, or the canonical morphism $A \to 1$ is epic. This allows a new proof of a result by Meyer *et al.* [39] on the semantic consequences that hold in set-theoretic models of the simply-typed lambda calculus with possibly empty types.

The last part of this dissertation is concerned with the study of properties of asynchronous communication in networks of communicating processes. Informally, communication in such a network is said to be *synchronous* if message transmission is instantaneous, such that sender and receiver must be available at the same time in order to communicate. It is *asynchronous* if messages are assumed to travel through a communication medium with possible delay, such that the sender cannot be certain when a message has been received. Asynchronous communication is often studied in the framework of a concurrent process calculus such as the asynchronous $\pi$-calculus [26, 9] or the join calculus [17, 18]. Here, we study asynchronous communication in general, independently of any particular process paradigm. We model processes by labeled transition systems with input and output. These transition systems are similar to the input/output automata by Lynch and Stark [35], but our presentation is more category-theoretic in a style that resembles Abramsky's interaction categories [1, 2]. In particular, we adopt Abramsky's notation $\mathbf{S};\mathbf{T}$ for the sequential composition of two processes, by which we mean the process obtained by connecting the output of $\mathbf{S}$ to the input of $\mathbf{T}$.

First, we formalize the intuitive notion of asynchrony in elementary terms: we define a process to be asynchronous if its input and/or output is filtered through an explicitly modeled communication medium, such as a buffer or a queue, possibly with feedback. For instance, we call a process *out-buffered* if it is, up to weak bisimulation, of the form $\mathbf{S};\mathcal{B}$, where $\mathcal{B}$ is a special buffer process. Our main result about asynchronous processes is a characterization of various different such notions of asynchrony in terms of first- and second-order axioms. These axioms refer directly to the behavior of a process, without mentioning buffers or queues explicitly. For instance, a process is out-buffered if and only if it is weakly bisimilar to a process satisfying three properties which we call output-commutativity, output-confluence, and output-determinacy. We illustrate these concepts by applying them to an asynchronous version of Milner's CCS and to the core join calculus.

This thesis is organized as follows: Chapter 1 is a summary of standard concepts of category theory, domain theory, and universal algebra, which are needed throughout the thesis. Chapter 2 is an introduction to the untyped lambda calculus and its combinatory models. In Chapter 3, we investigate unorderable and absolutely unorderable models of the untyped lambda calculus. Chapter 4 is devoted to finite lambda models. In Chapter 5, we study Henkin representation theorems and their applications to non-strict models of the simply-typed and polymorphic lambda calculi. In Chapter 6, we investigate properties of asynchronous communication.

# Chapter 1

# Preliminaries

We begin by gathering some basic concepts from category theory, domain theory, and universal algebra. This is mostly for the purpose of fixing terminology and notation for the later chapters of this thesis, and to provide a brief reference. We do not give any proofs in this chapter. For a more complete and detailed introduction to category theory, see *e.g.* [20] or [36]. For an introduction to domain theory, see *e.g.* [3] or [47]. For an introduction to universal algebra, see *e.g.* [24] or [13].

## 1.1 Basic category theory

### 1.1.1 Categories

A ***category*** $\mathbf{C} = \langle|\mathbf{C}|, (-,-), \mathsf{id}, \circ\rangle$ consists of a class $|\mathbf{C}|$ of ***objects***, together with a set $(A, B)$ of ***morphisms*** for each pair of objects $A, B \in |\mathbf{C}|$, and together with operations

$$\begin{aligned}&\mathsf{id}_A \in (A, A)\\ &\circ_{A,B,C} : (B, C) \times (A, B) \to (A, C)\end{aligned}$$

for all $A, B, C \in |\mathbf{C}|$, satisfying

$$\begin{array}{ll}\mathsf{id}_B \circ f = f = f \circ \mathsf{id}_A, & \text{for } f \in (A, B)\\ (h \circ g) \circ f = h \circ (g \circ f) & \text{for } f \in (A, B), g \in (B, C), h \in (C, D).\end{array}$$

We will often omit the subscripts on $\mathsf{id}$ and $\circ$. A morphism $\mathsf{id}_A$ is called an ***identity morphism***, and $g \circ f$ is called the ***composition*** of $f$ and $g$. The set $(A, B)$ is called the ***hom-set*** of $A$ and $B$. If we want to make the category unambiguous, we also write hom-sets as $\mathbf{C}(A, B)$. A morphism $f \in (A, B)$ is also written $f : A \to B$ or $A \xrightarrow{f} B$, and we call $A$ the ***source*** or the ***domain*** and $B$ the ***target*** or the ***codomain*** of $f$. If $f : A \to B$ and $g : B \to C$, then we sometimes write the composition $g \circ f$ in diagrammatic order as $A \xrightarrow{f} B \xrightarrow{g} C$ or as $f; g$.

*Example 1.1.* The category $\mathscr{S}$ of sets has sets as its objects, and functions as its morphisms. Notice that the collection of all sets is not itself a set; this is why, in the definition of a category, one allows the collection of objects to be a proper class. A category is said to be ***small*** if the collection of its objects is a set.

A category is ***discrete*** if its only morphisms are identity morphisms. If $\mathbf{C}$ is any category, then its ***dual category*** $\mathbf{C}^{op}$ is defined by $|\mathbf{C}^{op}| = |\mathbf{C}|$ and $\mathbf{C}^{op}(A, B) = \mathbf{C}(B, A)$, *i.e.* by reversing the direction of all morphisms. If $\mathbf{C}$ and $\mathbf{D}$ are categories, then their ***product*** $\mathbf{C} \times \mathbf{D}$ is defined by $|\mathbf{C} \times \mathbf{D}| = |\mathbf{C}| \times |\mathbf{D}|$ and $(\langle A, A'\rangle, \langle B, B'\rangle) = (A, B) \times (A', B')$, with the pointwise identities and composition.

A diagram

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\ C \arrow[r, "k"'] & D \end{tikzcd}$$

is used as a notation for the statement

$$f \in (A,B) \text{ and } g \in (B,D) \text{ and } h \in (A,C) \text{ and } k \in (C,D) \text{ and } g \circ f = k \circ h,$$

and similarly for other diagrams. Note that this notation is not meant to imply that $A, B, C, D$ or $f, g, h, k$ are different. In the diagrammatic notation, we may also omit the names of the objects. Of course, it is then still understood that the appropriate morphisms are composable. The symbol ⊹ in a diagram removes exactly one equation, such that

$$\begin{tikzcd} \cdot \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] \arrow[r, phantom, "\text{⊹}"] & \cdot \arrow[r, "f"] & \cdot \end{tikzcd}$$

means $f \circ g = f \circ h$. This diagram does not say whether $g = h$. Diagrams are just a notation for ordinary mathematical statements, and we may use them together with logical symbols, quantifiers etc.

*Example 1.2.* A morphism $f$ is said to be ***monic*** or a ***monomorphism*** if

$$\forall g, h \qquad ( \begin{tikzcd} \cdot \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] \arrow[r, phantom, "\text{⊹}"] & \cdot \arrow[r, "f"] & \cdot \end{tikzcd} \qquad \Rightarrow \qquad g = h).$$

Dually, $f$ is said to be ***epic*** or an ***epimorphism*** if

$$\forall g, h \qquad ( \begin{tikzcd} \cdot \arrow[r, "f"] & \cdot \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] \arrow[r, phantom, "\text{⊹}"] & \cdot \end{tikzcd} \qquad \Rightarrow \qquad g = h).$$

Also, $f$ is said to be ***iso*** or an ***isomorphism*** if

$$\exists g \qquad \begin{tikzcd} \cdot \arrow[rr, "\mathsf{id}"] \arrow[dr, "g"'] & & \cdot \arrow[dr, "g"] & \\ & \cdot \arrow[ur, "f"'] \arrow[rr, "\mathsf{id}"'] & & \cdot \end{tikzcd}$$

If $f$ is an isomorphism, then $g$ is uniquely determined. $g$ is called the ***inverse*** of $f$ and it is denoted by $f^{-1}$. If there is an isomorphism $f : A \to B$, then $A$ and $B$ are said to be isomorphic objects. We sometimes write $f : A \rightarrowtail B$ for a monomorphism, $f : A \twoheadrightarrow B$ for an epimorphism, and $f : A \xrightarrow{\sim} B$ for an isomorphism. Notice that if $f : A \to B$ has a left inverse $g \circ f = \mathsf{id}_A$, then $f$ is a monic, called a ***split monic***, and $g$ is an epic, called a ***split epic***. A collection of morphisms $(A \xrightarrow{f_i} B_i)_{i \in I}$ with the same source is called ***collectively monic*** or a ***monic cone*** if for all $g, h : C \to A$, whenever $f_i \circ g = f_i \circ h$ for all $i \in I$, then $g = h$.

### 1.1.2 Functors

If $\mathbf{C}$ and $\mathbf{D}$ are categories, then a ***(covariant) functor*** $F : \mathbf{C} \to \mathbf{D}$ is a map $F : |\mathbf{C}| \to |\mathbf{D}|$ of objects, together with a map $F : \mathbf{C}(A, B) \to \mathbf{D}(FA, FB)$ for each hom-set, such that

$$\begin{aligned} F\mathsf{id}_A &= \mathsf{id}_{FA} \\ F(g \circ f) &= Fg \circ Ff \end{aligned}$$

The category of small categories, together with functors between them, is denoted $\mathcal{C}at$. A functor $F : \mathbf{C}^{op} \to \mathbf{D}$ is also called a ***contravariant functor*** from $\mathbf{C}$ to $\mathbf{D}$.

*Example 1.3.* For any category $\mathbf{C}$, there is a functor $\mathrm{Hom} : \mathbf{C}^{op} \times \mathbf{C} \to \mathscr{S}$, which is defined by $\mathrm{Hom}(A, B) = \mathbf{C}(A, B)$ and $\mathrm{Hom}(f, g)(x) = g \circ x \circ f$. For any object $A \in |\mathbf{C}|$, the functor $(A, -) : \mathbf{C} \to \mathscr{S}$ is called the $A$-th ***(covariant) representable functor***. Dually, the functor $(-, A) : \mathbf{C}^{op} \to \mathscr{S}$ is called the $A$-th ***contravariant representable functor***.

A functor $F : \mathbf{C} \to \mathbf{D}$ is ***full*** if each $F : (A, B) \to (FA, FB)$ is onto. $F$ is an ***embedding*** if each $F : (A, B) \to (FA, FB)$ is one-to-one. We say $F$ is ***faithful*** if it is an embedding and it reflects isomorphisms, *i.e.*, whenever $Ff$ is an isomorphism, then so is $f$.

A category $\mathbf{C}$ is a ***subcategory*** of $\mathbf{D}$ if $|\mathbf{C}| \subseteq |\mathbf{D}|$, and for all $A, B \in |\mathbf{C}|$, $\mathbf{C}(A, B) \subseteq \mathbf{D}(A, B)$. The corresponding ***inclusion functor*** $I : \mathbf{C} \to \mathbf{D}$, with $IA = A$ and $If = f$, is always an embedding. $\mathbf{C}$ is said to be a ***full subcategory*** if $I$ is full, and a ***faithful subcategory*** if $I$ is faithful.

### 1.1.3 Natural transformations

A ***natural transformation*** $\eta : F \to G$ between functors $F, G : \mathbf{C} \to \mathbf{D}$ is a family $(\eta_A)_{A \in |\mathbf{C}|}$ of morphisms $\eta_A : FA \to GA$ such that for all $f : A \to B$,

$$\begin{tikzcd} FA \arrow[r, "\eta_A"] \arrow[d, "Ff"'] & GA \arrow[d, "Gf"] \\ FB \arrow[r, "\eta_B"'] & GB \end{tikzcd}$$

There is a category whose objects are functors $F : \mathbf{C} \to \mathbf{D}$, for fixed $\mathbf{C}$ and $\mathbf{D}$ (say, $\mathbf{C}$ is small). The morphisms are natural transformations. For any functor $F$, the identity natural transformation $\mathsf{id}_F : F \to F$ is defined by $(\mathsf{id}_F)_A = \mathsf{id}_{FA}$. Composition of natural transformations $\eta : F \to G$ and $\eta' : G \to H$ is defined by $(\eta' \circ \eta)_A = \eta'_A \circ \eta_A$. The resulting category is written $\mathbf{D}^{\mathbf{C}}$, and it is called a ***functor category***.

Two functors $F, G : \mathbf{C} \to \mathbf{D}$ are said to be ***naturally isomorphic***, in symbols $F \cong G$, if there are natural transformations $\eta : F \to G$ and $\eta^{-1} : G \to F$ such that $\eta \circ \eta' = \mathsf{id}_G$ and $\eta' \circ \eta = \mathsf{id}_F$.

We sometimes write $\eta : F(A) \to_A G(A)$, $\eta : F(A, B) \to_{A,B} G(A, B)$ etc. to express that $\eta$, as a transformation of functors, is natural in the indicated arguments. Similarly, we write $F(A) \cong_A G(A)$ etc. to express that $F$ and $G$ are naturally isomorphic. Notice that this is different from writing $(\forall A) F(A) \cong G(A)$; the latter statement expresses only a condition on objects, and not on morphisms.

An ***equivalence of categories*** $\mathbf{C}$ and $\mathbf{D}$ is a pair of functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ such that $G \circ F$ and $F \circ G$ are naturally isomorphic to the identity functors on $\mathbf{C}$ and $\mathbf{D}$, respectively.

### 1.1.4 Adjunctions

An ***adjunction*** between functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ is a natural isomorphism

$$\varphi : (FA, B) \xrightarrow{\sim}_{A,B} (A, GB).$$

In this case, the pair of functors $F$ and $G$ is called an ***adjoint pair***, and we write $\varphi : F \dashv G$, or simply $F \dashv G$. $F$ is a ***left adjoint*** of $G$ and $G$ is a ***right adjoint*** of $F$. The ***unit*** $u : \mathsf{id}_{\mathbf{C}} \to G \circ F$ of an adjunction $\varphi$ is the natural transformation given by $u_A = \varphi(\mathsf{id}_{FA}) \in (A, GFA)$, and the ***co-unit*** $c : F \circ G \to \mathsf{id}_{\mathbf{D}}$ is defined dually. Each of the entities $\varphi$, $u$ and $c$ determines the two others uniquely. Moreover, $F$ and $G$ determine each other up to natural isomorphism.

### 1.1.5 Limits and colimits

Let $I$ be a small category, $\mathbf{C}$ a category. A ***diagram*** in $\mathbf{C}$ modeled on $I$ is a functor $\Delta : I \to \mathbf{C}$. A ***cone*** over a diagram $\Delta$ is a pair $\langle D, (d_i)_{i \in |I|} \rangle$, consisting of an object $D$ and a family of morphisms $d_i : D \to \Delta(i)$ for each $i \in |I|$, such that for each $f : i \to j$ in $I$,

$$\begin{tikzcd}
\Delta(i) \arrow[rr, "\Delta(f)"] & & \Delta(j). \\
& D \arrow[ul, "d_i"] \arrow[ur, "d_j"'] &
\end{tikzcd}$$

A morphism between cones $\langle E, (e_i)_{i \in |I|} \rangle$ and $\langle D, (d_i)_{i \in |I|} \rangle$ over a diagram $\Delta$ is an arrow $f : E \to D$ such that $e_i = d_i \circ f$ for all $i \in |I|$. A cone $\langle D, (d_i)_{i \in |I|} \rangle$ is called ***limiting*** or a ***limit*** if it is terminal among cones over $\Delta$, *i.e.* from any other cone $\langle E, (e_i)_{i \in |I|} \rangle$, there is a unique morphism of cones $f : E \to D$. Sometimes, we also call the object $D$ a limit. The morphisms $e_i$ of a limiting cone are called ***limiting morphisms***. Limiting cones, if they exist at all, are uniquely determined up to isomorphism. Limiting cones are collectively monic. Cocones, colimits and colimiting morphisms are defined dually.

Some special limits are of interest: A limit of a diagram that is modeled on a discrete category is called a ***product***. The limiting morphisms of a product are called ***projections***. A limit of the empty diagram is called a ***terminator*** or a ***terminal object***. A limit of a diagram that is modeled on the category

$$\begin{tikzcd}
\cdot \arrow[r, shift left] \arrow[r, shift right] & \cdot
\end{tikzcd}$$

is called an ***equalizer***. A limit of a diagram that is modeled on the category

$$\begin{tikzcd}
& \cdot \arrow[d] \\
\cdot \arrow[r] & \cdot
\end{tikzcd}$$

is called a ***pullback***. The dual concepts are coproduct, coterminator or initial object, co-equalizer, and pushout.

**Definition.** A category is ***complete*** if every small diagram has a limit, and ***cocomplete*** if every small diagram has a colimit.

**Proposition 1.4.** *A category is complete iff it has products and equalizers. It is cocomplete iff it has coproducts and co-equalizers.*

### 1.1.6 Cartesian-closed categories

Recall that an object $B$ is a ***terminator*** if for all $A$, $(A, B)$ is a singleton. A terminator is unique up to isomorphism. If we have chosen a terminator in a category, we denote it by $1$. The unique morphism in $(A, 1)$ is then denoted $\bigcirc_A$.

A diagram

$$\begin{tikzcd}
& P \arrow[dl, "f"'] \arrow[dr, "g"] & \\
B & & C
\end{tikzcd}$$

is called a ***(binary) product diagram*** if for every pair of morphisms $q : A \to B$ and $r : A \to C$, there exists a unique $s : A \to P$ such that $f \circ s = q$ and $g \circ s = r$. This is the case if and only if

$$(A, P) \cong_A (A, B) \times (A, C)$$

via a natural isomorphism that relates $\mathsf{id}_P$ to the pair $\langle f, g \rangle$. Product diagrams are determined (for fixed $B$ and $C$) uniquely up to isomorphism. If we have chosen, for any $B$ and $C$, a product diagram, then we denote it by

$$\begin{tikzcd} & B \times C \arrow[dl, "\pi"'] \arrow[dr, "\pi'"] & \\ B & & C. \end{tikzcd}$$

The unique morphism $s : A \to B \times C$ such that $\pi \circ s = q : A \to B$ and $\pi' \circ s = r : A \to C$ is then denoted $\langle q, r\rangle$. The operation that takes $q$ and $r$ to $\langle q, r\rangle$ is called ***pairing***. If $b : B \to B'$ and $c : C \to C'$, then we denote by $b \times c$ the morphism $\langle b \circ \pi, c \circ \pi'\rangle : B \times B' \to C \times C'$. This makes $F(B, C) = B \times C$ into a functor.

In a category with chosen products, a diagram

$$D \times B \xrightarrow{f} C$$

is called an ***exponential diagram*** if for every morphism $g : A \times B \to C$ there is a unique $h : A \to D$ such that

$$\begin{tikzcd} D \times B \arrow[r, "f"] & C. \\ A \times B \arrow[u, "h \times \mathsf{id}_B"] \arrow[ur, "g"'] & \end{tikzcd}$$

This is the case if and only if

$$(A, D) \cong_A (A \times B, C)$$

via a natural isomorphism that relates $\mathsf{id}_D$ to $f$. For given $B$ and $C$, exponential diagrams are determined uniquely up to isomorphism. If we have chosen, for any $B$ and $C$, an exponential diagram, then we denote it by

$$C^B \times B \xrightarrow{\varepsilon} C.$$

The unique morphism $h : A \to C^B$ such that $\varepsilon \circ (h \times \mathsf{id}_B) = g : A \times B \to C$ is then denoted $g^\star$. The operation that takes $g$ into $g^\star$ is called ***currying***. The inverse operation, which takes $h$ to $h_\star = \varepsilon \circ (h \times \mathsf{id}_B)$, is called ***uncurrying***. If $b : B' \to B$ and $c : C \to C'$, then $c^b$ denotes the morphism $(c \circ \varepsilon \circ (\mathsf{id}_{C^B} \times b))^\star : C^B \to C'^{B'}$. This makes $F(B, C) = C^B$ into a functor, contravariant in the first argument and covariant in the second.

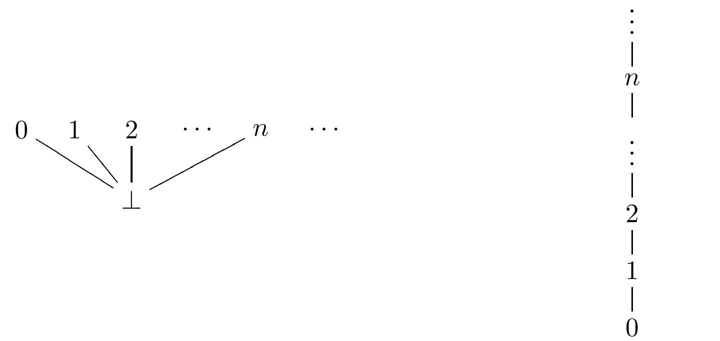
*Remark.* The following identities are often useful, where $a : A' \to A$, $h : A \to C^B$, $g : A \times B \to C$:

$$\begin{array}{rclcl} (g \circ (a \times \mathsf{id}_B))^\star & = & g^\star \circ a & : & A' \to C^B \\ \varepsilon^\star & = & \mathsf{id} & : & C^B \to C^B \\ (h \circ a)_\star & = & h_\star \circ (a \times \mathsf{id}_B) & : & A' \times B \to C \\ \mathsf{id}_\star & = & \varepsilon & : & C^B \times B \to C \end{array}$$

**Definition.** A ***cartesian-closed category (ccc)*** is a category with chosen terminator, chosen binary product diagrams and chosen exponential diagrams. A ***ccc-representation*** is a functor that preserves the chosen terminator, product and exponential diagrams. A functor that preserves ccc structure up to isomorphism is called a ***ccc-representation up to isomorphism***.

*Example 1.5.* For any small category $\mathbf{C}$, the functor category $\mathscr{S}^{\mathbf{C}^{op}}$ is cartesian-closed. The ***Yoneda embedding*** $Y : \mathbf{C} \to \mathscr{S}^{\mathbf{C}^{op}}$ maps an object $A$ to the functor $(-, A) : \mathbf{C}^{op} \to \mathscr{S}$. The Yoneda embedding is full and faithful, and if $\mathbf{C}$ is cartesian-closed, then $Y$ is a ccc-representation up to isomorphism. The functor category $\mathscr{S}^{\mathbf{C}^{op}}$ is called the ***category of presheaves over*** $\mathbf{C}$.

Table 1.1: Some posets



(1) The flat natural numbers (2) The ordinal $\omega$

## 1.2 Basic domain theory

We gather some basic domain-theoretic concepts. For a more detailed introduction, consult *e.g.* the texts by Abramsky and Jung [3] or Plotkin [47].

### 1.2.1 Preorders and posets

A binary relation $\leqslant$ on a set $D$ is called a ***preorder*** if

1. $\forall x \in D.\ x \leqslant x$ (Reflexivity).
2. $\forall x, y, z \in D.\ x \leqslant y$ and $y \leqslant z \Rightarrow x \leqslant z$ (Transitivity).

A preorder $\leqslant$ is a ***partial order*** if, in addition,

3. $\forall x, y \in D.\ x \leqslant y$ and $y \leqslant x \Rightarrow x = y$ (Antisymmetry).

A partially ordered set $\langle D, \leqslant\rangle$ is also called a ***poset***. A function $f : D \to E$ between posets is ***monotone*** if $x \leqslant y$ implies $fx \leqslant fy$, for all $x, y \in D$. We denote the category of posets and monotone functions by **POSET**. It is cartesian-closed. The exponential $E^D$ is given by the set of all monotone functions from $D$ to $E$, with the ***pointwise order***, $f \leqslant g$ if for all $x \in D$, $fx \leqslant gx$.

For $A \subseteq D$, let $\downarrow A$ be the set $\{y \in D \mid \exists x \in A. y \leqslant x\}$. A set $A$ is called ***downward closed*** or a ***downdeal*** if $A = \downarrow A$. If $A = \{x\}$ is a singleton, we also write $\downarrow x = \downarrow\{x\}$. The sets $\uparrow A$ and $\uparrow x$ are defined dually. An element $x \in A$ is said to be ***minimal*** in $A$ if $\downarrow x \cap A = \{x\}$. Also, $x \in A$ is said to be a ***minimum*** or a ***least element*** of $A$ if $A \subseteq \uparrow x$. Maximal elements and greatest elements are defined dually. An element $b \in D$ is said to be an ***upper bound*** of $A$ if $a \leqslant b$ for all $a \in A$. If among the upper bounds of $A$ there is a least one, it is called the ***least upper bound***, the ***join*** or the ***supremum*** of $A$, and it is denoted by $\bigvee A$ or $\bigvee_{x \in A} x$. We also write $x \vee y$ for the supremum of $\{x, y\}$, if it exists. Lower bounds are defined dually, and a greatest lower bound, denoted $\bigwedge A$, is also called a ***meet*** or an ***infimum***. A poset $D$ is called a ***lattice*** if it has finite suprema and infima, and a ***complete lattice*** if it has arbitrary suprema and infima.

A poset $\langle D, \leqslant\rangle$ is ***pointed*** if it has a least element $\perp$. $D$ is ***flat*** if it is pointed and if all elements $a \neq \perp$ are maximal. An example of a flat poset are the “flat natural numbers”, shown in Table 1.1(1). Two elements $x, y \in D$ are called ***compatible***, in symbols $x \frown y$, if there exists $z \in D$ with $x \leqslant z$ and $y \leqslant z$. Notice that two elements in a flat poset are compatible iff and of them is $\perp$.

*Remark.* Any poset $\langle D, \leqslant\rangle$ can itself be regarded as a category with hom-sets

$$D(x,y) = \begin{cases} \{*\} & \text{if } x \leqslant y \\ \emptyset & \text{else.} \end{cases}$$

Under this interpretation, a least element $\perp$ is just an initial object, suprema are colimits, functors are monotone maps, and an adjunction is a pair of monotone maps $f : D \to E$ and $g : E \to D$ such that

$$fx \leqslant y \iff x \leqslant gy.$$

### 1.2.2 Complete partial orders

A poset $I$ is ***directed*** if it is non-empty and if for all $x, y \in I$, there exists $z \in I$ with $x, y \leqslant z$. A poset $\langle D, \leqslant\rangle$ is ***directed complete*** if every directed subset has a supremum. A directed complete poset is also called a ***dcpo***. Directed suprema are also denoted by $\bigvee^{\!\nearrow} I$ or $\bigvee^{\!\nearrow}_{x\in I} x$. A function $f : D \to E$ between dcpo's is called ***Scott-continuous*** if it is monotone and it preserves directed suprema. We denote the category of dcpo's and Scott-continuous functions by **DCPO**. The full subcategory of pointed dcpo's is denoted by $\mathbf{DCPO}_\perp$. Both these categories are cartesian-closed. The exponential $E^D$ is given by the set of all Scott-continuous functions from $D$ to $E$ with the pointwise order. Directed suprema in $E^D$ can be computed pointwise, *i.e.*

$$(\bigvee^{\!\nearrow}_i f_i)(x) = \bigvee^{\!\nearrow}_i (f_i x)$$

A poset $I$ is ***linearly ordered*** or a ***chain*** if for all $x, y \in I$, either $x \leqslant y$ or $x \geqslant y$. An example of a linearly ordered set is the ordinal $\omega$, which is the set of natural numbers with their natural order, as shown in Table 1.1(2). A set $I$ which is isomorphic to $\omega$ is called an ***ω-chain***. A poset $\langle D, \leqslant\rangle$ is ***ω-complete*** if every $\omega$-chain $I \subseteq D$ has a supremum in $D$. An $\omega$-complete poset is also called a ***complete partial order*** or a ***cpo***. A function $f : D \to E$ between cpo's is called ***ω-continuous*** if it is monotone and it preserves suprema of $\omega$-chains. We denote the category of cpo's and $\omega$-continuous functions by **CPO**, and its full subcategory of pointed cpo's by $\mathbf{CPO}_\perp$. These categories are cartesian-closed, with the exponential $E^D$ given by the set of all $\omega$-continuous functions from $D$ to $E$ with the pointwise order, and pointwise suprema of $\omega$-chains.

*Remark.* The categories **CPO** and **DCPO** have similar properties. **DCPO** is a subcategory of **CPO**, but is neither full, nor is it a sub-ccc.

### 1.2.3 Bounded complete partial orders

A subset $A$ of a partially ordered set $D$ is called ***bounded*** if there is $d \in D$ with $A \subseteq \downarrow d$. A cpo $D$ is ***bounded complete*** if every bounded subset $A \subseteq D$ has a supremum. Bounded complete cpo's and $\omega$-continuous functions form a full sub-ccc $\mathbf{CPO}^{bc}$ of **CPO**. Notice that we do *not* require the morphisms to preserve all bounded suprema. The categories $\mathbf{CPO}^{bc}_\perp$, $\mathbf{DCPO}^{bc}$, and $\mathbf{DCPO}^{bc}_\perp$ are defined analogously.

### 1.2.4 Stability

A cpo $D$ is a ***meet cpo*** if it has bounded binary meets which act continuously. This means, that for every $x \in D$, the set $\downarrow x$ has binary meets, and the function $\langle a, b\rangle \mapsto a \wedge b$ is continuous on $\downarrow x \times \downarrow x$. A function $f : D \to E$ between meet cpo's is ***stable*** if it preserves bounded binary meets. We denote the category of meet cpo's and stable maps by $\mathbf{CPO}^\wedge$, and its full subcategory of pointed meet cpo's by $\mathbf{CPO}^\wedge_\perp$. These categories are cartesian-closed too, and the exponential $E^D$ is given by the set of all stable functions from $D$ to $E$, not with the pointwise order, but with the ***Berry order*** or ***stable order***:

$$f \leqslant_s g \iff (\forall x, y \in D.\, x \leqslant y \Rightarrow f(x) = f(y) \wedge g(x))$$

Directed suprema, as well as bounded infima, with respect to the Berry order are taken pointwise. The cartesian-closed categories $\mathbf{DCPO}^\wedge$ and $\mathbf{DCPO}^\wedge_\perp$ are defined analogously.

The theory of meet cpo's and stable functions is due to Berry [7], who used them to study the semantics of sequential computations.

### 1.2.5 Domain equations

Let $\mathcal{D}$ be any one of the pointed categories $\mathbf{DCPO}_\perp$, $\mathbf{CPO}_\perp$, $\mathbf{DCPO}^\wedge_\perp$, $\mathbf{CPO}^\wedge_\perp$, $\mathbf{DCPO}^{bc}_\perp$, or $\mathbf{CPO}^{bc}_\perp$. The objects of $\mathcal{D}$ are called ***domains***. One of the main features of these categories of domains is that they can be used to solve ***domain equations***, such as

$$D \cong D^D.$$

A solution to such an equation in a category $\mathcal{D}$ consists of an object $D$, together with an isomorphism $\varphi : D \to D^D$. The ability to solve domain equations is an essential tool in mathematical programming semantics to give meaning to a variety of programming language constructs, such as recursive data types. We are particularly interested in solutions to the "classic" domain equation $D \cong D^D$, whose solutions yield models of the untyped lambda-$\beta\eta$-calculus (see Section 2.7).

General methods for solving domain equations were pioneered by D. Scott [53], and further developed by Smyth and Plotkin [61]. In general, a domain equation takes the form $D \cong F(D)$. Notice that, since the right-hand-side may contain positive (covariant) as well as negative (contravariant) occurrences of $D$, $F$ will not in general be a functor. The problem of mixed variance can be solved by passing from the category $\mathcal{D}$ to a category $\mathcal{D}^e$ of embeddings. The objects of $\mathcal{D}^e$ are the same as the objects of $\mathcal{D}$. The morphisms of $\mathcal{D}^e$ are embeddings, where $e : D \to E$ in $\mathcal{D}$ is called an ***embedding*** if there exists a ***projection*** $p : E \to D$ in $\mathcal{D}$ such that

$$p \circ e = \mathsf{id}_D \qquad e \circ p \leqslant \mathsf{id}_E,$$

where the inequality is understood to be with respect to the relevant order on functions, *i.e.* the pointwise order in the case of $\mathbf{DCPO}_\perp$, $\mathbf{CPO}_\perp$, etc., and the stable order in the case of $\mathbf{DCPO}^\wedge_\perp$ or $\mathbf{CPO}^\wedge_\perp$. An embedding $e$ is uniquely determined by its associated projection $p$ and *vice versa*. We write $p = e^*$ and $e = p_*$. One also speaks of $\mathcal{D}^e$ as a category of ***embedding-projection pairs***.

An ***expanding sequence*** in $\mathcal{D}^e$ is a diagram modeled on the ordinal $\omega$, *i.e.* a functor $\Delta : \omega \to \mathcal{D}^e$. In more concrete terms, an expanding sequence is a sequence $(D_n)_{n\in\mathbb{N}}$ of objects, together with embeddings $e_{nm} : D_n \to D_m$ for all $n \leqslant m$, such that $e_{mm} = \mathsf{id}_{D_m}$ and $e_{mn} \circ e_{pm} = e_{pn}$, for all $p \leqslant m \leqslant n$.

**Proposition 1.6. Limit-colimit coincidence.** *Every expanding sequence $\langle (D_n)_n, (e_{nm})_{n\leqslant m}\rangle$ in $\mathcal{D}^e$ has a colimit $D$ in $\mathcal{D}^e$, with colimiting morphisms $e_n : D_n \to D$. Moreover, $\langle D, (e_n)_n\rangle$ is also a colimit in $\mathcal{D}$, and $\langle D, (e^*_n)_n\rangle$ is a limit of $\langle (D_n)_n, (e^*_{nm})_{n\leqslant m}\rangle$ in $\mathcal{D}$. This is called the **limit-colimit coincidence**, and $D$ is also called a **bilimit**.* □

**Proposition 1.7. Characterization of bilimits.** *Let $\langle D, (e_n)_n\rangle$ be a cocone over the expanding sequence $\langle (D_n)_n, (e_{nm})_{n\leqslant m}\rangle$ in $\mathcal{D}^e$. Then $\langle D, (e_n)_n\rangle$ is a bilimit if and only if*

$$\mathsf{id}_D = \bigvee^{\uparrow}_{n} e_n \circ e^*_n.$$

□

**Definition.** A functor $F : \mathcal{D}^e \to \mathcal{D}^e$ is ***continuous*** if for every expanding sequence $\langle (D_n)_n, (e_{nm})_{n\leqslant m}\rangle$ with colimit $\langle D, (e_n)_n\rangle$, the sequence $\langle (FD_n)_n, (Fe_{nm})_{n\leqslant m}\rangle$ has colimit $\langle FD, (Fe_n)_n\rangle$.

**Proposition 1.8. Solution of domain equations.** *Consider a domain equation $D \cong F(D)$, where $F : \mathcal{D}^e \to \mathcal{D}^e$ is a continuous functor. Starting with a domain $D_0$ and an embedding $e_{01} : D_0 \to F(D_0)$, let $D_{n+1} = F(D_n)$ and $e_{n+1,n+2} = F(e_{n,n+1})$ for all $n \in \mathbb{N}$. Let $D$ be the colimit of the expanding sequence $\langle (D_n)_n, (e_{nm})_{n\leqslant m}\rangle$, where $e_{nm} = e_{m-1,m} \circ \cdots \circ e_{n,n+1}$, for $n \leqslant m$. Then $D \cong F(D)$.*

*Proof.* Since $F$ is continuous, the sequence $\langle (FD_n)_n, (Fe_{nm})_{n\leqslant m}\rangle = \langle (D_{n+1})_n, (e_{n+1,m+1})_{n\leqslant m}\rangle$ has colimit $F(D)$. On the other hand, $D$ is a colimit of the same sequence, hence one gets $D \cong F(D)$. □

The question remains how to identify a given functor as continuous. A useful criterion was given by Smyth and Plotkin [61], who observed that a continuous functor $F$ on $\mathcal{D}^e$ can be obtained from a *locally continuous functor* $\hat{F}$ on $\mathcal{D}$. This works even if $\hat{F}$ is of mixed variance.

**Definition.** A functor $\hat{F} : \mathcal{D}^{op} \times \mathcal{D} \to \mathcal{D}$ is ***locally continuous*** if for all objects $D, D', E, E' \in \mathcal{D}$,

$$\hat{F} : \mathcal{D}(D, D') \times \mathcal{D}(E, E') \to \mathcal{D}(\hat{F}(D', E), \hat{F}(D, E'))$$

if continuous as a map between hom-sets (with the pointwise order in the case of $\mathbf{DCPO}_\perp$ and $\mathbf{CPO}_\perp$, and the stable order in the case of $\mathbf{DCPO}^\wedge_\perp$ and $\mathbf{CPO}^\wedge_\perp$).

**Proposition 1.9.** *Every locally continuous functor $\hat{F} : \mathcal{D}^{op} \times \mathcal{D} \to \mathcal{D}$ gives rise to a continuous functor $F : \mathcal{D}^e \to \mathcal{D}^e$, defined by*

$$F(D) = \hat{F}(D, D) \qquad F(e) = \hat{F}(e^*, e)$$

□

### 1.2.6 The $D_\infty$-construction

The method of Proposition 1.8, applied to the locally continuous functor $F(D, E) = E^D$, serves to solve the "classic" domain equation $D \cong D^D$. This construction is due to D. Scott, and it is called the $D_\infty$-construction.

*Remark.* Notice that the construction of a $D_\infty$-model is dependent on some parameters, namely a category $\mathcal{D}$, an object $D_0$ and an embedding $e_{01} : D_0 \to D_0^{D_0}$. Hence, there is a whole class of such models. Among these, we distinguish the ***standard $D_\infty$-model*** to be the one model constructed in $\mathbf{CPO}_\perp$ from the cpo $D_0$ with two elements $\perp \leqslant \top$, and the embedding $e_{01} : D_0 \to D_0^{D_0}$ which maps $\perp$ to the constant $\perp$ function and $\top$ to the constant $\top$ function.

## 1.3 Basic universal algebra

### 1.3.1 $\Sigma$-algebras

An ***algebraic signature*** $\Sigma$ is a pair $\langle \Omega, \alpha \rangle$ consisting of a set $\Omega$ of ***function symbols*** and a map $\alpha : \Omega \to \mathbb{N}$, assigning an ***arity*** $k \geqslant 0$ to each $f \in \Omega$. We let $\Omega_k = \{f \in \Omega \mid \alpha(f) = k\}$ be the set of $k$-ary function symbols. A **$\Sigma$*-algebra*** $\mathbf{A} = \langle A, I \rangle$ is a set $A$ together with an interpretation $I(f)$ of every function symbol as a map from $A^{\alpha(f)} \to A$. We often write $|\mathbf{A}|$, or even $\mathbf{A}$ for the underlying set $A$ of an algebra, and $f_\mathbf{A}$ or even $f$ for the interpretation $I(f)$. A ***homomorphism of $\Sigma$-algebras*** $\varphi : \mathbf{A} \to \mathbf{B}$ is a function $\varphi : |\mathbf{A}| \to |\mathbf{B}|$ such that for all $f \in \Omega_k$ and all $a_1, \ldots, a_k \in \mathbf{A}$

$$\varphi(f_\mathbf{A}(a_1, \ldots, a_k)) = f_\mathbf{B}(\varphi a_1, \ldots, \varphi a_k)$$

We denote the category of $\Sigma$-algebras and homomorphisms by $\Sigma$-**Alg**. The category $\Sigma$-**Alg** has all limits, and the forgetful functor $U : \Sigma\text{-}\mathbf{Alg} \to \mathscr{S}$ preserves and reflects them. For instance, binary products are given by $|\mathbf{A} \times \mathbf{B}| = |\mathbf{A}| \times |\mathbf{B}|$ and $f_{\mathbf{A}\times\mathbf{B}}(\langle a_1, b_1 \rangle, \ldots, \langle a_k, b_k \rangle) = \langle f_\mathbf{A}(a_1, \ldots, a_k), f_\mathbf{B}(b_1, \ldots, b_k) \rangle$. A $\Sigma$-algebra $\mathbf{A}$ is a ***subalgebra*** of another $\Sigma$-algebra $\mathbf{B}$ if $|\mathbf{A}| \subseteq |\mathbf{B}|$ and for all $f \in \Omega_k$ and $a_1, \ldots, a_k \in \mathbf{A}$, $f_\mathbf{A}(a_1, \ldots, a_k) = f_\mathbf{B}(a_1, \ldots, a_k)$. The inclusion map $\mathbf{A} \to \mathbf{B}$ of a subalgebra is a homomorphism.

**Definition 1.10.** A binary relation $R$ on a $\Sigma$-algebra $\mathbf{A}$ is ***compatible*** if it is a subalgebra of $\mathbf{A} \times \mathbf{A}$. This is the case if and only if whenever $\langle a_i, b_i \rangle \in R$ for $i = 1 \ldots k$, then $\langle f a_1 \ldots a_k, f b_1 \ldots b_k \rangle \in R$, for each $k$-ary function symbol $f \in \Omega_k$. A ***congruence*** on a $\Sigma$-algebra $\mathbf{A}$ is a compatible equivalence relation. If $\sim$ is a congruence, then the ***quotient algebra*** $\mathbf{A}/\!\sim$ is a well-defined $\Sigma$-algebra via $f_{\mathbf{A}/\sim}([x_1]_\sim, \ldots, [x_k]_\sim) = [f_\mathbf{A}(x_1, \ldots, x_k)]_\sim$. The natural map $\mathbf{A} \to \mathbf{A}/\!\sim$ is a homomorphism of $\Sigma$-algebras. The ***kernel*** $\ker\varphi$ of a homomorphism $\varphi : \mathbf{A} \to \mathbf{B}$ is the congruence relation $\sim$ on $\mathbf{A}$ defined by $a \sim a'$ iff $\varphi(a) = \varphi(a')$.

### 1.3.2 Term algebras

Let $X$ be a set. For each $x \in X$, pick a distinct symbol $c_x$, which is not in $\Omega$. Let $W(X,\Omega)$ be the set of words (*i.e.*finite sequences) from the alphabet $\{c_x \mid x \in X\} \cup \Omega$.

**Definition.** The set of ***Σ-terms over X*** is defined to be the smallest subset $T \subseteq W(X,\Omega)$ such that

$$\frac{x \in X}{c_x \in T} \qquad \frac{f \in \Omega_k \qquad t_1 \in T \qquad \dots \qquad t_k \in T}{ft_1 \dots t_k \in T}.$$

Let $\Sigma_X$ be the set of $\Sigma$-terms thus defined. It has a natural $\Sigma$-algebra structure via $f_{\Sigma_X}(t_1,\dots,t_k) = ft_1 \dots t_k$. The algebra $\Sigma_X$ is called the ***Σ-term algebra over X***.

*Remark.* We have represented terms as words from some alphabet. There are other possible choices; for instance, one could represent a term $ft_1 \dots t_k$ as a labeled rooted tree with label $f$ at the root and with immediate subtrees $t_1,\dots,t_k$. In general, we will not be too concerned here with the details of how to represent syntax; rather, we will treat syntax as a primitive notion. Independently of which concrete representation for terms one chooses, $\Sigma_X$, together with its natural map $\jmath : X \to \Sigma_X : x \mapsto c_x$, is completely determined by the following universal property:

**Proposition 1.11.** *For any $\Sigma$-algebra $\mathbf{B}$ and any map $\rho : X \to \mathbf{B}$, there is a unique homomorphism $\hat{\rho} : \Sigma_X \to \mathbf{B}$ such that*

$$\begin{tikzcd} X \arrow[r, "\jmath"] \arrow[dr, "\rho"'] & \Sigma_X \arrow[d, "\hat{\rho}"] \\ & \mathbf{B}. \end{tikzcd}$$

*Equivalently, the forgetful functor $U : \Sigma\text{-}\mathbf{Alg} \to \mathscr{S}$ has a left adjoint $\mathcal{F} : \mathscr{S} \to \Sigma\text{-}\mathbf{Alg}$ with $\mathcal{F}(X) = \Sigma_X$, and with $X \to \Sigma_X$ as the unit of the adjunction.* □

A map $\rho : X \to \mathbf{B}$ is also called a ***valuation*** in $\mathbf{B}$. If $\hat{\rho}$ is the unique extension of $\rho$ to terms, then we often write $[\![t]\!]_\rho$ instead of $\hat{\rho}(t)$ for the interpretation of a term $t \in \Sigma_X$. The defining equations for $[\![\ ]\!]$ are

$$\begin{array}{rcll} [\![x]\!]_\rho & = & \rho(x), & \text{for } x \text{ a variable,} \\ [\![ft_1 \dots t_k]\!]_\rho & = & f_{\mathbf{A}}([\![t_1]\!]_\rho,\dots,[\![t_k]\!]_\rho), & \text{for } f \in \Omega_k. \end{array}$$

If $X = \{x_1,\dots,x_n\}$ is a finite set of variables, then a term $t \in \Sigma_X$ is also called a ***n-ary operation*** in $\Sigma$. We write $t = t(x_1,\dots,x_n)$. If $b_1,\dots,b_n$ are elements of a $\Sigma$-algebra $\mathbf{B}$, then we sometimes write $t(b_1,\dots,b_n)$ for $[\![t]\!]_\rho$ where $\rho : X \to \mathbf{B} : x_i \mapsto b_i$.

### 1.3.3 Algebraic varieties

Fix a countable set $\mathcal{V}$ of ***variables***. A ***Σ-equation*** is a pair of terms $\langle t,s\rangle \in \Sigma_{\mathcal{V}} \times \Sigma_{\mathcal{V}}$. Equations are often written in the form $t = s$. A $\Sigma$-algebra $\mathbf{A}$ ***satisfies*** an equation $t = s$, in symbols $\mathbf{A} \models t = s$, if for all homomorphisms $\varphi : \Sigma_{\mathcal{V}} \to \mathbf{A}$, $\varphi(t) = \varphi(s)$. Equivalently, $\mathbf{A} \models t = s$ if for all valuations $\rho : \mathcal{V} \to \mathbf{A}$, $[\![t]\!]_\rho = [\![s]\!]_\rho$.

**Definition.** Let $\Sigma$ be a signature, and let $\mathcal{E}$ be a set of $\Sigma$-equations. A $\Sigma$-algebra $\mathbf{A}$ that satisfies all equations in $\mathcal{E}$ is called a ***$(\Sigma,\mathcal{E})$-algebra***. The $(\Sigma,\mathcal{E})$-algebras form a full subcategory of $\Sigma$-**Alg**, which we denote by $(\Sigma,\mathcal{E})$-**Alg**. Any full subcategory $\mathbf{T}$ of $\Sigma$-**Alg** that arises in this way is called an ***algebraic variety***. The algebras of an algebraic variety $\mathbf{T}$ are also called ***T-algebras***.

Let $\mathbf{T}$ be an algebraic variety, defined by a signature $\Sigma$ and equations $\mathcal{E}$. We construct $\mathbf{T}_X$, the ***free T-algebra*** over a set $X$, as follows: On the term algebra $\Sigma_X$, consider the smallest congruence relation $\sim$ such that

$$\frac{\langle s,t\rangle \in \mathcal{E} \qquad \rho : \Sigma_{\mathcal{V}} \to \Sigma_X}{\rho(s) \sim \rho(t)}.$$

Table 1.2: Equational rules for $\Sigma$-algebras

$$(\textit{refl})\ \frac{}{s=s} \qquad (\textit{symm})\ \frac{s=t}{t=s} \qquad (\textit{trans})\ \frac{s=t \quad t=u}{s=u}$$

$$(\textit{cong})\ \frac{f\in\Omega_k \quad s_i=t_i \quad (i=1\ldots k)}{fs_1\ldots s_k = ft_1\ldots t_k} \qquad (\textit{subst})\ \frac{s=t \quad \varphi:\Sigma_{\mathcal{V}}\to\Sigma_{\mathcal{V}}}{\varphi(s)=\varphi(t)}$$

Let $\mathbf{T}_X$ be the algebra $\Sigma_X/\!\sim$. Then $\mathbf{T}_X$ is a $\mathbf{T}$-algebra. Together with the natural map $\jmath : X \to \mathbf{T}_X$, it has the universal property:

**Proposition 1.12.** *For any* $\mathbf{T}$*-algebra* $\mathbf{B}$ *and any map* $\rho : X \to \mathbf{B}$*, there is a unique homomorphism* $\hat\rho : \mathbf{T}_X \to \mathbf{B}$ *such that*

$$\begin{tikzcd} X \arrow[r, "\jmath"] \arrow[dr, "\rho"'] & \mathbf{T}_X \arrow[d, "\hat\rho"] \\ & \mathbf{B}. \end{tikzcd}$$

□

We say that a set of equations $\mathcal{E}$ ***entails*** an equation $s=t$, in symbols $\mathcal{E}\vdash_{eq} s=t$, if $s=t$ can be derived from the hypotheses $\mathcal{E}$ by the rules in Table 1.2. We write $\mathcal{E}\models_{\Sigma\text{-}\mathbf{Alg}} s=t$ if for all $\Sigma$-algebras $\mathbf{A}$, if $\mathbf{A}\models\mathcal{E}$, then $\mathbf{A}\models s=t$.

**Proposition 1.13. Soundness and Completeness for $\Sigma$-algebras.**

$$\mathcal{E}\vdash_{eq} s=t \quad \textit{if and only if} \quad \mathcal{E}\models_{\Sigma\text{-}\mathbf{Alg}} s=t$$

□

### 1.3.4 Indeterminates

Let $\mathbf{T}$ be a variety with signature $\Sigma$ and equations $\mathcal{E}$. Let $\mathbf{A}$ be a $\mathbf{T}$-algebra, and let $X$ be a set. Assume without loss of generality that $X$ and $|\mathbf{A}|$ are disjoint. Relative to the variety $\mathbf{T}$, the ***polynomial algebra*** $\mathbf{A}[X]$ is defined as follows: On the term algebra $\Sigma_{|\mathbf{A}|+X}$, consider the smallest congruence relation $\sim$ such that

$$\frac{a = f_{\mathbf{A}}a_1\ldots a_k}{c_a \sim fc_{a_1}\ldots c_{a_k}} \qquad \frac{\langle s,t\rangle\in\mathcal{E} \quad \rho:\Sigma_{\mathcal{V}}\to\Sigma_{|\mathbf{A}|+X}}{\rho(s)\sim\rho(t)}.$$

Let $\mathbf{A}[X]$ be the algebra $\Sigma_{|\mathbf{A}|+X}/\!\sim$. Together with $\mathbf{A}[X]$, consider the natural maps $\iota:\mathbf{A}\to\mathbf{A}[X]$ defined by $\iota(a)=[c_a]_\sim$, and $\jmath : X\to\mathbf{A}[X]$ defined by $\jmath(x)=[c_x]_\sim$.

**Proposition 1.14.** $\mathbf{A}[X]$ *is a* $\mathbf{T}$*-algebra with the following universal property: For any* $\mathbf{T}$*-algebra* $\mathbf{B}$*, any homomorphism* $f:\mathbf{A}\to\mathbf{B}$ *of* $\mathbf{T}$*-algebras, and any map* $g:X\to\mathbf{B}$*, there is a unique homomorphism* $h:\mathbf{A}[X]\to\mathbf{B}$ *such that*

$$\begin{tikzcd} X \arrow[r, "\jmath"] \arrow[dr, "g"'] & \mathbf{A}[X] \arrow[d, "h"] & \mathbf{A} \arrow[l, "\iota"'] \arrow[dl, "f"] \\ & \mathbf{B}. & \end{tikzcd}$$

□

*Remark.* The map $\iota:\mathbf{A}\to\mathbf{A}[X]$ is always an injection; we will often regard it as an inclusion. Notice that $\mathbf{A}[X][Y]\cong\mathbf{A}[X+Y]$. In the case where $X=\{x_1,\ldots,x_n\}$ is finite, we write $\mathbf{A}[X]=\mathbf{A}[x_1,\ldots,x_n]$. The elements of $\mathbf{A}[X]$ are called ***polynomials***, and $X$ is called a set of ***indeterminates***.

Table 1.3: Inequational rules for Σ-algebras

$$(\mathit{refl})\quad \frac{}{s \leqslant s} \qquad\qquad (\mathit{cong})\quad \frac{f \in \Omega_k \qquad s_i \leqslant t_i \qquad (i = 1 \dots k)}{f s_1 \dots s_k \leqslant f t_1 \dots t_k}$$

$$(\mathit{trans})\quad \frac{s \leqslant t \qquad t \leqslant u}{s \leqslant u} \qquad\qquad (\mathit{subst})\quad \frac{s \leqslant t \qquad \varphi : \Sigma_{\mathcal{V}} \to \Sigma_{\mathcal{V}}}{\varphi(s) \leqslant \varphi(t)}$$

### 1.3.5 Ordered algebras

Let $\mathbf{A}$ be a $\Sigma$-algebra, and let $\leqslant$ be a partial order on the carrier set $|\mathbf{A}|$. The pair $\langle \mathbf{A}, \leqslant \rangle$ is called an ***ordered* $\boldsymbol{\Sigma}$*-algebra*** if the order $\leqslant$ is a compatible relation on $\mathbf{A}$ in the sense of Definition 1.10. Concretely, this is the case iff for each $f \in \Omega_k$, $f_{\mathbf{A}} : \mathbf{A}^k \to \mathbf{A}$ is a monotone map with respect to $\leqslant$. A ***homomorphism of ordered* $\boldsymbol{\Sigma}$*-algebras*** is a homomorphism of $\Sigma$-algebras that is monotone. We denote the resulting category of ordered $\Sigma$-algebras by $\Sigma$-**Ord**.

Just as we considered sets of equations for $\Sigma$-algebras, we may consider sets of inequations for ordered $\Sigma$-algebras. Recall that $\mathcal{V}$ is a countable set of variables, and that $\Sigma_{\mathcal{V}}$ is the $\Sigma$-term algebra. A $\boldsymbol{\Sigma}$***-inequation*** is a pair of terms $\langle t, s \rangle \in \Sigma_{\mathcal{V}} \times \Sigma_{\mathcal{V}}$, often written $t \leqslant s$. An ordered $\Sigma$-algebra $\langle \mathbf{A}, \leqslant \rangle$ ***satisfies*** an inequation $t \leqslant s$, in symbols $\mathbf{A} \models t \leqslant s$, if for all homomorphisms $\varphi : \Sigma_{\mathcal{V}} \to \mathbf{A}$ of $\Sigma$-algebras, $\varphi(t) \leqslant \varphi(s)$. Equivalently, $\mathbf{A} \models t \leqslant s$ if for all valuations $\rho : \mathcal{V} \to \mathbf{A}$, $[\![t]\!]_\rho \leqslant [\![s]\!]_\rho$.

**Definition.** An ordered $\Sigma$-algebra that satisfies a given set $\mathcal{I}$ of inequations is called an ***ordered* $(\boldsymbol{\Sigma}, \boldsymbol{\mathcal{I}})$*-algebra***. The ordered $(\Sigma, \mathcal{I})$-algebras form a full subcategory of $\Sigma$-**Ord**, which we denote by $(\Sigma, \mathcal{I})$-**Ord**. A full subcategory $\mathbf{O}$ of $\Sigma$-**Ord** that arises in this way is called an ***ordered variety***.

Let $\mathbf{O}$ be an ordered variety, defined by a pair $(\Sigma, \mathcal{I})$ of a signature and a set of inequations. The ***free ordered* $(\boldsymbol{\Sigma}, \boldsymbol{\mathcal{I}})$*-algebra*** over a poset $P$, denoted $\mathbf{O}_P$, is constructed as follows: On the term algebra $\Sigma_P$, consider the smallest compatible preorder $\preccurlyeq$ satisfying

$$\frac{\langle s, t \rangle \in \mathcal{I} \qquad \rho : \Sigma_{\mathcal{V}} \to \Sigma_P}{\rho(s) \preccurlyeq \rho(t)} \qquad\qquad \frac{x \leqslant y \in P}{c_x \preccurlyeq c_y}.$$

Let $\sim$ be the congruence $\preccurlyeq \cap \succcurlyeq$ on $\Sigma_P$, and let $\mathbf{O}_P$ be the algebra $\Sigma_P/{\sim}$, together with the partial order $\leqslant$ induced by $\preccurlyeq$ via $[x]_\sim \leqslant [y]_\sim$ iff $x \preccurlyeq y$. Then $\mathbf{O}_P$ is an ordered $(\Sigma, \mathcal{I})$-algebra, and the natural map $\jmath : P \to \mathbf{O}_P$ is monotone. The following universal property holds:

**Proposition 1.15.** *For any ordered* $(\Sigma, \mathcal{I})$*-algebra* $\mathbf{B}$ *and any monotone map* $\rho : P \to \mathbf{B}$*, there is a unique homomorphism of ordered* $(\Sigma, \mathcal{I})$*-algebras* $\hat{\rho} : \mathbf{O}_P \to \mathbf{B}$ *such that*

$$\begin{tikzcd} P \arrow[r, "\jmath"] \arrow[dr, "\rho"'] & \mathbf{O}_P \arrow[d, "\hat{\rho}"] \\ & \mathbf{B}. \end{tikzcd}$$

□

Rules for deriving inequations are given in Table 1.3. We say that a set of inequations $\mathcal{I}$ ***entails*** an inequation $s \leqslant t$, in symbols $\mathcal{I} \vdash_{\mathit{ineq}} s \leqslant t$, if $s \leqslant t$ can be derived from the hypotheses $\mathcal{I}$ by these rules. We write $\mathcal{E} \models_{\Sigma\text{-}\mathbf{Ord}} s \leqslant t$ if for all ordered $\Sigma$-algebras $\mathbf{A}$, if $\mathbf{A} \models \mathcal{E}$, then $\mathbf{A} \models s \leqslant t$.

**Proposition 1.16. Soundness and Completeness for ordered Σ-algebras.**

$$\mathcal{E} \vdash_{\mathit{eq}} s \leqslant t \quad \textit{if and only if} \quad \mathcal{E} \models_{\Sigma\text{-}\mathbf{Ord}} s \leqslant t$$

□

We also sometimes write $\mathcal{I} \vdash_{\mathit{ineq}} s = t$ as an abbreviation for $\mathcal{I} \vdash_{\mathit{ineq}} s \leqslant t$ and $\mathcal{I} \vdash_{\mathit{ineq}} t \leqslant s$.

### 1.3.6 Dcpo-algebras

Let $\Sigma$ be a signature. An ordered $\Sigma$-algebra $\langle \mathbf{A}, \leqslant \rangle$ is called a ***$\boldsymbol{\Sigma}$-dcpo-algebra*** if the partial order $\leqslant$ is directed complete, and if each interpreted operation $f_{\mathbf{A}} : \mathbf{A}^k \to \mathbf{A}$ is Scott-continuous. A ***homomorphism of $\boldsymbol{\Sigma}$-dcpo-algebras*** is a Scott-continuous homomorphism of ordered $\Sigma$-algebras. We denote the resulting category of $\Sigma$-dcpo-algebras by $\Sigma$-**DCPO**. Each set $\mathcal{I}$ of inequations determines a full subcategory of $\Sigma$-**DCPO**, which we denote by $(\Sigma, \mathcal{I})$-**DCPO**. We call such a subcategory a ***dcpo-variety***.

Let $\mathbf{D}$ be a dcpo-variety, defined by $(\Sigma, \mathcal{I})$. For every dcpo $D$, there exists a free $(\Sigma, \mathcal{I})$-dcpo-algebra $\mathbf{D}_D$ over $D$, with an associated continuous map $\jmath : D \to \mathbf{D}_D$, satisfying the usual universal property. The construction of the free dcpo-algebra is less trivial than in the case of ordered algebras, and it relies an Freyd's Adjoint Functor Theorem. A proof of the existence of $\mathbf{D}_D$ can be found in Abramsky and Jung [3].

# Chapter 2

# The Lambda Calculus is Algebraic

The correspondence between Church's untyped lambda calculus and Curry's and Schönfinkel's combinatory algebras is among the oldest known, and most esthetically pleasing, facts about the lambda calculus. However, the combinatory interpretation is also known to be somewhat imperfect, as Curry's combinatory abstraction operator does not in general satisfy the rule

$$(\xi)\ \frac{M = N}{\lambda x.M = \lambda x.N}.$$

One usually resolves this problem by moving from the class of lambda algebras to the smaller class of *lambda models*, which are, by definition, those lambda algebras in which ($\xi$) holds. However, unlike the class of lambda algebras, the class of lambda models is not equationally definable. Therefore, it fails to enjoy some useful closure properties such as being closed under subalgebras.

In this chapter, we point out that the failure of the $\xi$-rule, and the subsequent need for a non-equational class of models, is not due to the lambda calculus itself, but to the way free variables are usually interpreted in these models. The usual interpretation of a lambda term is defined relative to a valuation of its free variables. Essentially, this amounts to interpreting a term $M$ with $n$ free variables as a function $\mathbf{A}^n \to \mathbf{A}$. We argue that it is more natural to model free variables as algebraic indeterminates and to interpret $M$ as an element of a polynomial algebra $\mathbf{A}[x_1, \ldots, x_n]$. Based on this interpretation, we show that the class of lambda algebras is sound and complete for arbitrary lambda theories. In particular, the notorious rule ($\xi$) is sound with respect to this interpretation.

This chapter is intended to serve as a self-contained, brief introduction to the lambda calculus and its combinatory models. We do not claim originality for the results in this chapter, which follow from known results in Barendregt's book [5] and in the work of Koymans [31]. We do however hope to present these issues from a fresh point of view, particularly where the interpretation of free variables is concerned. Maybe this exposition will help to clarify the precise relationship between the lambda calculus, lambda algebras, and lambda models, which are sometimes confused in the literature.

Lambda conversion and reduction are introduced in Section 2.1. Combinatory algebras and lambda algebras are defined in Section 2.2. Section 2.3 contains a detailed analysis of the behavior of indeterminates in the theory of lambda algebras, which leads to a streamlined interpretation of the lambda calculus. In Section 2.4, we show that the categories of lambda theories and of lambda algebras are equivalent. This, to some extent, justifies the slogan "the lambda calculus is algebraic". Lambda models are the subject of Section 2.5, and Section 2.6 is devoted to models of the lambda-$\beta\eta$-calculus. Finally, in Section 2.7, we relate the different kinds of algebraic models to reflexive ccc models.

Table 2.1: The axioms and rules of the lambda calculus

| | | | |
|---|---|---|---|
| (*refl*) | $\dfrac{}{M = M}$ | (*cong*) | $\dfrac{M = M' \qquad N = N'}{MN = M'N'}$ |
| (*symm*) | $\dfrac{M = N}{N = M}$ | ($\xi$) | $\dfrac{M = N}{\lambda x.M = \lambda x.N}$ |
| (*trans*) | $\dfrac{M = N \qquad N = P}{M = P}$ | ($\beta$) | $\dfrac{}{(\lambda x.M)N = M[N/x]}$ |

## 2.1 The lambda calculus

The lambda calculus is a theory of *functions as rules*. Its two basic constructions are *functional application*, where $(fx)$ denotes the application of a function $f$ to an argument $x$, and *functional abstraction*, where $\lambda x.t$ denotes the function that maps $x$ to $t$.

**Definition.** Let $\mathcal{V}$ be a countable set of ***variables***, fixed throughout the rest of this chapter. Let $C$ be a set of ***constants***. The set of ***raw lambda terms*** $\Lambda_C^{\mathsf{raw}}$ is defined to be the least set of terms such that

$$\frac{x \in \mathcal{V}}{x \in \Lambda_C^{\mathsf{raw}}} \qquad \frac{c \in C}{c \in \Lambda_C^{\mathsf{raw}}} \qquad \frac{M, N \in \Lambda_C^{\mathsf{raw}}}{(MN) \in \Lambda_C^{\mathsf{raw}}} \qquad \frac{x \in \mathcal{V} \qquad M \in \Lambda_C^{\mathsf{raw}}}{(\lambda x.M) \in \Lambda_C^{\mathsf{raw}}}.$$

Notation: We often use upper case letters $M, N, \ldots$, as well as lower case letters $s, t, u, \ldots$ to denote lambda terms. We use $x, y, \ldots$ to denote variables. To save parentheses, we write $MNP$ instead of $((MN)P)$, $\lambda x.MN$ instead of $(\lambda x.(MN))$, and $\lambda x_1 \ldots x_n.M$ instead of $(\lambda x_1.(\ldots(\lambda x_n.M)\ldots))$. The set $\mathrm{FV}(M) \subseteq \mathcal{V}$ of ***free variables*** of a raw lambda term $M$ is defined recursively:

$$\mathrm{FV}(x) = \{x\} \qquad \mathrm{FV}(c) = \emptyset \qquad \mathrm{FV}(MN) = \mathrm{FV}(M) \cup \mathrm{FV}(N) \qquad \mathrm{FV}(\lambda x.M) = \mathrm{FV}(M) \setminus \{x\}.$$

Variables that are not free are ***bound***. We write $M =_\alpha N$ if $M$ and $N$ are equal up to renaming of bound variables. The set $\Lambda_C$ of ***lambda terms*** is then defined to be the set $\Lambda_C^{\mathsf{raw}}/{=_\alpha}$ of $\alpha$-equivalence classes of raw terms. From now on, we will consider terms up to $\alpha$-equivalence without further mentioning it. A term with no free variables is ***closed***. The set of closed terms is denoted $\Lambda_C^0$. We write $M[N/x]$ for the result of substituting $N$ for $x$ in $M$, taking appropriate care to ensure that neither $x$ nor any of the free variables of $N$ are bound in $M$. For a rigorous treatment of $\alpha$-equivalence and substitution, see *e.g.* [5].

### 2.1.1 Lambda conversion

The axioms and rules for deriving equations between lambda terms are shown in Table 2.1. If $\mathcal{E}$ is a set of equations, we write $\mathcal{E} \vdash_\beta M = N$ if $M = N$ is derivable from $\mathcal{E}$ by using these rules. A ***lambda theory*** is a set $\mathcal{T}$ of closed equations that is closed under derivability, *i.e.* $\mathcal{T} \vdash_\beta M = N$ implies $M = N \in \mathcal{T}$, for closed $M$ and $N$. For a given set of constants, there is a unique smallest theory $\underline{\lambda\beta}$, called the ***pure theory*** or the theory of ***$\beta$-conversion***. We also write $\vdash_\beta M = N$ as $M =_\beta N$ and we say that $M$ and $N$ are ***$\beta$-convertible***.

The ***lambda-$\beta\eta$-calculus*** is the lambda calculus with the additional axiom

$$(\eta)\,\frac{x \notin \mathrm{FV}(M)}{\lambda x.Mx = M}.$$

We write $\mathcal{E} \vdash_{\beta\eta} M = N$ if $M = N$ is derivable from a set of equations $\mathcal{E}$ and the axiom ($\eta$). A lambda theory $\mathcal{T}$ which is closed under $\vdash_{\beta\eta}$ is called a ***lambda-$\beta\eta$-theory***. The unique smallest such theory is called the theory of ***$\beta\eta$-conversion***, and it is denoted $\underline{\lambda\beta\eta}$. If $\vdash_{\beta\eta} M = N$, then we write $M =_{\beta\eta} N$ and we say that $M$ and $N$ are ***$\beta\eta$-convertible***.

Table 2.2: Reduction rules of the lambda calculus

$$(\mathit{refl})\ \frac{}{M \longrightarrow M} \qquad (\mathit{cong})\ \frac{M \longrightarrow M' \quad N \longrightarrow N'}{MN \longrightarrow M'N'}$$

$$(\mathit{trans})\ \frac{M \longrightarrow N \quad N \longrightarrow P}{M \longrightarrow P} \qquad (\xi)\ \frac{M \longrightarrow N}{\lambda x.M \longrightarrow \lambda x.N}$$

$$(\beta)\ \frac{}{(\lambda x.M)N \longrightarrow M[N/x]}$$

*Remark.* The notion of theory given here is a slightly more liberal than the one given in [5], where the equations of a theory are not allowed to contain any constants.

Notice that the lambda calculus is not given by a signature and equations in the sense of universal algebra. However, we will show in Section 2.3 that the lambda calculus is equivalent, in a suitable sense, to an algebraic theory.

### 2.1.2 Lambda reduction and consistency

When considering functions as rules, it is natural to think of the evaluation of a function applied to an argument as a dynamic process. This process is made explicit in the notions of $\beta$-reduction and $\beta\eta$-reduction.

A term of the form $(\lambda x.M)N$ is called a ***β-redex***, and it ***β-reduces*** to $M[N/x]$. The relation $\xrightarrow{\beta}$ is the reflexive, transitive and contextual closure of this one-step $\beta$-reduction. More precisely, $\xrightarrow{\beta}$ is the smallest relation on lambda terms satisfying the axioms and rules in Table 2.2. A term of the form $\lambda x.Mx$, where $x \notin \mathrm{FV}(M)$, is called an ***η-redex***, and it ***η-reduces*** to $M$. The relation $\xrightarrow{\beta\eta}$ is the reflexive, transitive and contextual closure of the one-step $\beta$- and $\eta$-reductions, *i.e.*, it is the smallest relation satisfying the axioms and rules in Table 2.2 and also the axiom

$$(\eta)\ \frac{x \notin \mathrm{FV}(M)}{\lambda x.Mx \longrightarrow M}.$$

A term $M$ is said to be in ***β-normal form*** no subterm is a $\beta$-redex, *i.e.* if $M \xrightarrow{\beta} M' \Rightarrow M = M'$. Similarly, $M$ is in ***βη-normal form*** if it contains no $\beta$- or $\eta$-redex, *i.e.* if $M \xrightarrow{\beta\eta} M' \ \Rightarrow\ M = M'$. Examples of terms in $\beta$-normal form include the ***booleans*** $T = \lambda xy.x$ and $F = \lambda xy.y$, as well as the ***Church numerals*** $\bar{0} = \lambda xy.x$, $\bar{1} = \lambda xy.xy$, $\bar{2} = \lambda xy.x(xy)$ etc.; all of these except for $\bar{1}$ are also in $\beta\eta$-normal form.

**Definition.** A binary relation $\longrightarrow$ is said to have the ***diamond property*** if whenever $a \longrightarrow b$ and $a \longrightarrow c$, then there exists $d$ such that $b \longrightarrow d$ and $c \longrightarrow d$. In diagrams:

$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ b & & c \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ b \arrow[dr] & & c \arrow[dl] \\ & d & \end{tikzcd}$$

Also, a relation $\longrightarrow$ is said to be ***Church-Rosser*** if the transitive closure $\longrightarrow^*$ has the diamond property.

**Theorem 2.1 (Church, Rosser [12]).** *The relations $\xrightarrow{\beta}$ and $\xrightarrow{\beta\eta}$ are Church-Rosser.* □

This theorem was first proved by Church and Rosser in 1936 [12]. Since then, the proof has been adapted and streamlined in various ways by Tait, Martin-Löf, Girard and others. One can find a proof in Barendregt's book [5].

The Church-Rosser Theorem has several important consequences. As a first consequence, one proves that for each pair of $\beta$-convertible lambda terms $M =_\beta N$, there is a term $P$ with $M \xrightarrow{\beta} P$ and $N \xrightarrow{\beta} P$. This

Table 2.3: The axioms and rules of combinatory logic

| | | | |
|---|---|---|---|
| (*refl*) | $\dfrac{}{A = A}$ | (*cong*) | $\dfrac{A = A' \qquad B = B'}{AB = A'B'}$ |
| (*symm*) | $\dfrac{A = B}{B = A}$ | (*k*) | $\dfrac{}{\mathbf{K}AB = A}$ |
| (*trans*) | $\dfrac{A = B \qquad B = C}{A = C}$ | (*s*) | $\dfrac{}{\mathbf{S}ABC = AC(BC)}$ |

is easily shown by induction on the derivation of $M =_\beta N$, using the rules in Table 2.1. This immediately implies consistency of the lambda calculus:

**Corollary 2.2. Consistency.** *If $M$ and $N$ are two different terms in $\beta$-normal form, then $M \neq_\beta N$. If $M$ and $N$ are two different terms in $\beta\eta$-normal form, then $M \neq_{\beta\eta} N$.* □

## 2.2 Combinatory models of the lambda calculus

### 2.2.1 Combinatory algebras and combinatory logic

**Definition.** An ***applicative structure*** $(\mathbf{A}, \cdot)$ is a set $\mathbf{A}$ together with a binary operation. A ***combinatory algebra*** $(\mathbf{A}, \cdot, k, s)$ is an applicative structure with distinguished elements $k$ and $s$ such that for all $x, y, z \in \mathbf{A}$,

$$kxy = x \qquad sxyz = xz(yz)$$

Here we write $kxy$ for $(k \cdot x) \cdot y$, etc. A ***homomorphism of combinatory algebras*** is $f\colon \mathbf{A} \to \mathbf{B}$ such that $fk = k$, $fs = s$ and $f(x \cdot y) = fx \cdot fy$, for all $x, y \in \mathbf{A}$.

*Example.* The ***closed term algebra*** associated with a lambda theory $\mathcal{T}$ is $(\Lambda^0_C/\mathcal{T}, \cdot, K, S)$, where $\Lambda^0_C/\mathcal{T}$ is the set of $\mathcal{T}$-equivalence classes of closed terms, $M \cdot N = (MN)$, $K = \lambda xy.x$ and $S = \lambda xyz.xz(yz)$. Similarly, the ***open term algebra*** is $(\Lambda_C/\mathcal{T}, \cdot, K, S)$.

Combinatory algebras form an algebraic variety. The corresponding algebraic language is combinatory logic: let $\mathcal{V}$ be a set of variables and $C$ a set of constants as before. The set $\mathfrak{C}_C$ of ***combinatory terms*** or ***terms of combinatory logic*** is defined to be the smallest set of terms such that

$$\frac{x \in \mathcal{V}}{x \in \mathfrak{C}_C} \qquad \frac{c \in C}{c \in \mathfrak{C}_C} \qquad \frac{A, B \in \mathfrak{C}_C}{(AB) \in \mathfrak{C}_C} \qquad \frac{}{\mathbf{K} \in \mathfrak{C}_C} \qquad \frac{}{\mathbf{S} \in \mathfrak{C}_C}.$$

Again, we economize the use of parentheses by writing $ABC$ instead of $((AB)C)$. A combinatory term is ***closed*** if it contains no variables. The set of closed terms is denoted by $\mathfrak{C}^0_C$. A closed and constant-free term, *i.e.* a term that is made up only from $\mathbf{K}$ and $\mathbf{S}$, is also called a ***combinator***. The axioms and rules for deriving equations of combinatory logic are shown in Table 2.3. We write $\mathcal{E} \vdash_{CL} A = B$ if $A = B$ can be derived from a set of equations $\mathcal{E}$ by these rules. A ***theory of combinatory logic*** is a set of closed equations that is closed under derivability. The minimal theory is denoted *CL*, and we also write $A =_{CL} B$ instead of $\vdash_{CL} A = B$.

Terms of combinatory logic can be interpreted in a combinatory algebra $\mathbf{A}$, relative to a valuation $\rho$ of variables an an interpretation $I$ of constants. We call this the *local* interpretation to distinguish it from the absolute interpretation that we will consider in Section 2.3.2.

**Definition. Local interpretation of combinatory logic.** Let $\mathbf{A}$ be a combinatory algebra, and let $I : C \to \mathbf{A}$ be an interpretation of constants in $\mathbf{A}$. A ***valuation*** of variables in $\mathbf{A}$ is a map $\rho\colon \mathcal{V} \to \mathbf{A}$. The ***local***

***interpretation*** $[\![A]\!]^I_\rho$ of a term $A \in \mathfrak{C}_C$ is defined inductively:

$$[\![x]\!]^I_\rho = \rho(x) \qquad [\![c]\!]^I_\rho = I(c) \qquad [\![\mathbf{K}]\!]^I_\rho = k \qquad [\![\mathbf{S}]\!]^I_\rho = s \qquad [\![AB]\!]^I_\rho = [\![A]\!]^I_\rho \cdot [\![B]\!]^I_\rho.$$

For terms $A, B \in \mathfrak{C}_C$, we say that the interpretation $I$ ***locally satisfies*** the equation $A = B$, notation $I \models A = B$, if for all valuations $\rho$ in $\mathbf{A}$, $[\![A]\!]^I_\rho = [\![B]\!]^I_\rho$. We write $\mathcal{E} \models_{\mathbf{CA}} A = B$ if for all combinatory algebras $\mathbf{A}$ and all interpretations $I : C \to \mathbf{A}$, if $I \models \mathcal{E}$ then $I \models A = B$.

**Proposition 2.3. Soundness and Completeness for combinatory logic.** *Let $\mathcal{E}$ be a set of closed equations of combinatory logic. For combinatory terms $A$ and $B$,*

$$\mathcal{E} \vdash_{CL} A = B \quad \textit{if and only if} \quad \mathcal{E} \models_{\mathbf{CA}} A = B.$$

□

### 2.2.2 The derived lambda abstractor

The significance of the two combinators $\mathbf{K}$ and $\mathbf{S}$ of combinatory logic lies in the fact that they can be used to simulate lambda abstraction. Define $\mathbf{I} = \mathbf{SKK}$. Notice that $\mathbf{I}x =_{CL} x$, for all $x$. For a combinatory term $A \in \mathfrak{C}_C$ and a variable $x \in \mathcal{V}$, define the term $\lambda^* x.A \in \mathfrak{C}_C$ inductively:

$$\begin{array}{rcll} \lambda^* x.x & = & \mathbf{I} & \\ \lambda^* x.B & = & \mathbf{K}B, & \text{if } x \notin \mathrm{FV}(B) \\ \lambda^* x.BC & = & \mathbf{S}(\lambda^* x.B)(\lambda^* x.C), & \text{otherwise.} \end{array}$$

Notice that $(\lambda^* x.A)x =_{CL} A$ can be shown by induction for any term $A$. Also, $\mathrm{FV}(\lambda^* x.A) = \mathrm{FV}(A) \setminus \{x\}$. We call $\lambda^*$ the ***derived lambda abstractor*** of combinatory logic. It is important to remark here that, in general, the operator $\lambda^*$ is well-defined only on *terms*, and not on equivalence classes of terms. For this reason, the $\lambda^*$ operator does not, in general, yield an operator $\lambda^* \colon \mathbf{A}[x] \to \mathbf{A}$, for a combinatory algebra $\mathbf{A}$. We will see in Section 2.3.2 that we do get such an operator when $\mathbf{A}$ is a lambda algebra.

**Proposition 2.4. Combinatory completeness.** *For every term $B \in \mathfrak{C}_C$ with variables in $x_1, \dots, x_n$, there exists a closed term $A$ such that $B =_{CL} Ax_1 \cdots x_n$.*

*Proof.* Let $B = \lambda^* x_1 \dots x_n.A$. □

As a consequence, in the variety of combinatory algebras, all elements of $\mathbf{A}[x_1, \dots, x_n]$ can be written in the form $Ax_1 \dots x_n$, where $A \in \mathbf{A}$. However, such $A$ is not necessarily unique.

### 2.2.3 The local interpretation of lambda terms

Using the derived lambda abstractor $\lambda^*$ of combinatory logic, we can define translations $cl \colon \Lambda_C \to \mathfrak{C}_C$ and $\lambda \colon \mathfrak{C}_C \to \Lambda_C$ from lambda terms to combinatory terms and *vice versa*:

$$\begin{array}{rcl} x_{cl} & = & x \\ c_{cl} & = & c \\ (MN)_{cl} & = & M_{cl}N_{cl} \\ (\lambda x.M)_{cl} & = & \lambda^* x.M_{cl} \end{array} \qquad \begin{array}{rcl} x_\lambda & = & x \\ c_\lambda & = & c \\ (AB)_\lambda & = & A_\lambda B_\lambda \\ \mathbf{K}_\lambda & = & \lambda xy.x \\ \mathbf{S}_\lambda & = & \lambda xyz.xz(yz) \end{array}$$

Notice: Again, these translations are defined on terms, rather than equivalence classes of terms. For example, $(\lambda z.(\lambda x.x)z)_{cl} = \mathbf{S}(\mathbf{KI})\mathbf{I}$ and $(\lambda z.z)_{cl} = \mathbf{I}$ are not equivalent in combinatory logic. The following hold:

**Lemma 2.5.** *For any lambda term $M$, we have $M_{cl,\lambda} =_\beta M$. For combinatory terms $A, B$, if $A =_{CL} B$ then $A_\lambda =_\beta B_\lambda$. For lambda terms $M, N$, if $M_{cl} =_{CL} N_{cl}$, then $M =_\beta N$. For a combinatory term $A$, $(\lambda^* x.A)_\lambda =_\beta \lambda x.A_\lambda$.* □

We can now interpret lambda terms in any combinatory algebra, by first translating them into combinatory logic via *cl*:

**Definition. Local interpretation of lambda terms.** Let $\mathbf{A}$ be a combinatory algebra and $I : C \to \mathbf{A}$ an interpretation of constants. For lambda terms $M, N \in \Lambda_C$ and a valuation $\rho\colon \mathcal{V} \to \mathbf{A}$, define

$$\begin{array}{rcl} [\![M]\!]^I_\rho & = & [\![M_{cl}]\!]^I_\rho \\ I \models M = N & \text{iff} & I \models M_{cl} = N_{cl} \\ \mathrm{Th}(I) & = & \{M = N \mid M, N \in \Lambda^0_C,\ I \models M = N\} \end{array}$$

This interpretation is not sound for the lambda calculus, since there are derivable equations, such as for instance $\lambda z.(\lambda x.x)z = \lambda z.z$, that do not hold in all combinatory algebras. In particular, Th$(I)$ need not be a lambda theory!

This leads us to consider the class of lambda algebras, which are precisely those combinatory algebras that satisfy all the equations of the lambda calculus.

### 2.2.4 Lambda algebras

Let $\mathbf{A}$ be a combinatory algebra. Then $\mathfrak{C}_{\mathbf{A}}$ is the set of combinatory terms with one constant symbol for each element of $\mathbf{A}$. Let $I_0 : \mathbf{A} \to \mathbf{A}$ be the canonical interpretation of each constant symbol as itself, *i.e.* the identity function. For $A, B \in \mathfrak{C}_{\mathbf{A}}$, we write $\mathbf{A} \models A = B$ instead of $I_0 \models A = B$.

**Definition.** A combinatory algebra $\mathbf{A}$ is called a ***lambda algebra*** if for all combinatory terms $A, B \in \mathfrak{C}_{\mathbf{A}}$,

$$A_\lambda =_\beta B_\lambda \qquad \Rightarrow \qquad \mathbf{A} \models A = B.$$

A ***homomorphism of lambda algebras*** is a homomorphism of combinatory algebras.

*Example.* For any lambda theory $\mathcal{T}$, the open term algebra $\Lambda_C/\mathcal{T}$ and the closed term algebra $\Lambda^0_C/\mathcal{T}$ are lambda algebras. In the open terms algebra, $\Lambda_C/\mathcal{T} \models A = B$ iff $\mathcal{T} \vdash_\beta A_\lambda = B_\lambda$.

**Proposition 2.6 (Curry).** *Lambda algebras form an algebraic variety. In fact, the class of lambda algebras can be axiomatized over the class of combinatory algebras by the following five closed equations, known as the* ***Curry axioms****:*

1. $k = s(s(ks)(s(kk)k))(k(skk))$
2. $s = s(s(ks)(s(k(s(ks)))(s(k(s(kk)))s)))(k(k(skk)))$
3. $s(kk) = s(s(ks)(s(kk)(s(ks)k)))(kk)$
4. $s(ks)(s(kk)) = s(kk)(s(s(ks)(s(kk)(skk)))(k(skk)))$
5. $s(k(s(ks)))(s(ks)(s(ks))) = s(s(ks)(s(kk)(s(ks)(s(k(s(ks)))s))))(ks)$

*Proof.* See [5]. We will give a different axiomatization of lambda algebras in Remark 2.20. $\square$

We denote the variety of lambda algebras by **LA**. We write $\vdash_{LA}$ for provability from the axioms and rules of combinatory logic plus the five Curry axioms. We also write $A =_{LA} B$ instead of $\vdash_{LA} A = B$. The following complements Lemma 2.5:

**Lemma 2.7.** *For any combinatory term $A$, we have $A_{\lambda,cl} =_{LA} A$. For lambda terms $M, N$, if $M =_\beta N$, then $M_{cl} =_{LA} N_{cl}$. For combinatory terms $A, B$, if $A_\lambda =_\beta B_\lambda$, then $A =_{LA} B$.* $\square$

If $\mathcal{E}$ is a set of equations, we write $\mathcal{E} \models_{\mathbf{LA}} A = B$ if for all lambda algebras $\mathbf{A}$ and all interpretations $I : C \to \mathbf{A}$, if $I \models \mathcal{E}$ then $I \models A = B$. The following is a soundness and completeness theorem for the *pure* lambda calculus, *i.e.* for equations $M = N$ that are provable in the pure theory $\underline{\lambda\beta}$. In Section 2.3.3, we prove a more general theorem for arbitrary theories.

**Theorem 2.8. Soundness and completeness for the pure lambda calculus.** *For lambda terms $M, N \in \Lambda_C$,*

$$\vdash_\beta M = N \quad \textit{if and only if} \quad \models_{\mathbf{LA}} M = N.$$

*Proof.* Soundness follows directly from the definition of lambda algebras. For completeness, notice that the open term algebra $\Lambda/\underline{\lambda\beta}$ of the lambda beta calculus is a lambda algebra in which $M = N$ iff $M =_\beta N$. $\square$

**Remark 2.9. Failure of ($\xi$) for the local interpretation.** The reason that we state the soundness and completeness only for the pure lambda calculus at this point is that in general, the local interpretation in a lambda algebra does not satisfy the rule ($\xi$), *i.e.* it is not in general true that $I \models A = B$ implies $I \models \lambda^* x.A = \lambda^* x.B$. A counterexample is the closed term algebra $\mathcal{M}^0$ of the lambda-$\beta$-calculus. Plotkin [46] shows that there exist closed terms $M, N$ such that for all closed terms $Z$, $MZ =_\beta NZ$, but $Mx \neq_\beta Nx$ for a variable $x$. Hence $\mathcal{M}^0 \models Mx = Nx$, but $\mathcal{M}^0 \not\models \lambda x.Mx = \lambda x.Nx$. The absolute interpretation, to be defined in Section 2.3.2, takes care of this problem.

## 2.3 Lambda algebras and indeterminates

### 2.3.1 A characterization of $\mathrm{A}[x]$ for lambda algebras

Recall that for a combinatory algebra $\mathbf{A}$, we denote by $\mathbf{A}[x]$ the algebra obtained by freely adjoining an indeterminate $x$ to $\mathbf{A}$ in the variety of combinatory algebras. If $\mathbf{A}$ is a lambda algebra then so is $\mathbf{A}[x]$. More generally, if $\mathbf{A}$ is a lambda algebra and $f\colon \mathbf{A} \to \mathbf{B}$ is a homomorphism of combinatory algebras, then $\mathbf{B}$ is a lambda algebra. This is because lambda algebras are definable by closed equations (Proposition 2.6).

If $\mathbf{A}$ is a lambda algebra, then $\mathbf{A}[x]$ has an interesting explicit description. The following construction is similar to constructions given by Krivine [32] and, in the case of Curry algebras, by Freyd [19]. Let $\mathbf{A} = (\mathbf{A}, \cdot, k, s)$, and define $\mathbf{B} = (\mathbf{B}, \bullet, K, S)$, where

$$\begin{array}{rcll} \mathbf{B} & = & \{a \in \mathbf{A} \mid a = \mathbf{1}a\}, & \text{where } \mathbf{1} = s(ki),\ i = skk \\ a \bullet b & = & sab & \\ K & = & kk & \\ S & = & ks & \end{array}$$

Note: $ab$ denotes application in $\mathbf{A}$, and $a \bullet b$ denotes application in $\mathbf{B}$.

**Proposition 2.10.** *1. $\mathbf{B}$ is a well-defined combinatory algebra.*

2. *The map $\iota\colon \mathbf{A} \to \mathbf{B}$ with $\iota(a) = ka$ is a well-defined homomorphism.*

3. *For every homomorphism $f\colon \mathbf{A} \to \mathbf{C}$ and every $x \in \mathbf{C}$, there is a unique homomorphism $g\colon \mathbf{B} \to \mathbf{C}$ such that $f = g \circ \iota$ and $g(i) = x$. Consequently, $\mathbf{B} \cong \mathbf{A}[x]$.*

Notice that $\mathbf{1}ab =_{CL} ab$, and $\mathbf{1}_\lambda =_\beta \lambda xy.xy$. The proof of Proposition 2.10 relies on the following seven properties of lambda algebras. We will see later that lambda algebras are already characterized by these properties (see Remark 2.20).

**Lemma 2.11.** *The following hold in any lambda algebra:*

$$\begin{array}{lrcl} (a) & \mathbf{1}(sa) & = & sa, \\ (b) & \mathbf{1}(sab) & = & sab, \\ (c) & \mathbf{1}(ka) & = & ka, \\ (d) & s(s(kk)a)b & = & \mathbf{1}a, \\ (e) & s(s(s(ks)a)b)c & = & s(sac)(sbc), \\ (f) & k(ab) & = & s(ka)(kb), \\ (g) & s(ka)i & = & \mathbf{1}a, \end{array}$$

*Proof.* One easily checks that $(\mathbf{1}(sa))_\lambda =_\beta (sa)_\lambda$, and similarly for the other equations. $\square$

*Proof of Proposition 2.10:*

1.: It follows by Lemma 2.11$(a)$–$(c)$ that all of $K$, $S$, $a \bullet b$, $i$ and $\mathbf{1}$ are elements of $\mathbf{B}$, for any $a, b \in \mathbf{B}$. In particular, the operations on $\mathbf{B}$ are well-defined. Moreover, for all $a, b, c \in B$,

$$K \bullet a \bullet b = s(s(kk)a)b \overset{2.11(d)}{=} \mathbf{1}a = a, \qquad \text{and}$$

$$S \bullet a \bullet b \bullet c = s(s(s(ks)a)b)c \overset{2.11(e)}{=} s(sac)(sbc) = a \bullet c \bullet (b \bullet c).$$

2.: Using Lemma 2.11$(f)$,

$$\iota(ab) = k(ab) = s(ka)(kb) = \iota(a) \bullet \iota(b).$$

3.: Define $g(a) = f(a) \cdot x$, and check that this has the desired properties. For uniqueness, take any homomorphism $h \colon \mathbf{B} \to \mathbf{C}$ such that $f = h \circ \iota$ and $h(i) = x$. Then for all $a \in \mathbf{B}$,

$$h(a) = h(\mathbf{1}a) \overset{2.11(g)}{=} h(s(ka)i) = h((ka) \bullet i) = h(ka) \cdot h(i) = h(\iota a) \cdot h(i) = f(a) \cdot x = g(a).$$

$\square$

**Corollary 2.12.** *Let $\mathbf{A}$ be a lambda algebra, and $a, b \in \mathbf{A}$. Then $ax = bx$ holds in $\mathbf{A}[x]$ if and only if $\mathbf{1}a = \mathbf{1}b$ holds in $\mathbf{A}$.*

*Proof.* $\Rightarrow$: Suppose $a, b \in \mathbf{A}$ and $ax = bx$ in $\mathbf{A}[x]$. By Proposition 2.10, items 1. and 2., there is a unique map $h \colon \mathbf{A}[x] \to \mathbf{B}$ extending $\iota$ and sending $x$ to $i$. Then

$$\mathbf{1}a \overset{2.11(g)}{=} s(ka)i = (ka) \bullet i = h(ax) = h(bx) = (kb) \bullet i = s(kb)i \overset{2.11(g)}{=} \mathbf{1}b.$$

$\Leftarrow$: $\mathbf{1}a = \mathbf{1}b$ in $\mathbf{A} \Rightarrow \mathbf{1}a = \mathbf{1}b$ in $\mathbf{A}[x] \Rightarrow ax = \mathbf{1}ax = \mathbf{1}bx = bx$ in $\mathbf{A}[x]$. $\square$

### 2.3.2 The absolute interpretation

Let $M \in \mathfrak{C}_C$ be a lambda term whose free variables are among $x_1, \ldots, x_n = \bar{x}$. Let $\mathbf{A}$ be a combinatory algebra, and let $I : C \to \mathbf{A}$ be an interpretation of constants. The local interpretation $[\![M]\!]^I_\rho$, defined in Section 2.2.1, depends on a valuation of variables $\rho \colon \mathcal{V} \to \mathbf{A}$. Since, in fact, it depends only on the values of $\rho$ at $x_1, \ldots, x_n$, the local interpretation can be viewed as a function $[\![M]\!]^I_{\bar{x}} \colon \mathbf{A}^n \to \mathbf{A}$, sending an $n$-tuple $\bar{a} \in \mathbf{A}^n$ to $[\![M]\!]^I_{(\bar{x}:=\bar{a})}$. In these terms, an equation $M = N$ holds *locally* in $\mathbf{A}$ if $M$ and $N$ define the same function $\mathbf{A}^n \to \mathbf{A}$.

We will now consider a different interpretation of terms, interpreting $M$ as an element in $\mathbf{A}[\bar{x}]$, *i.e.* as a polynomial. We call this the *absolute* interpretation of $M$. The absolute interpretation distinguishes more terms than the local one, since, in general, two different polynomials may define the same function. For closed terms, however, the absolute and the local interpretations coincide.

**Definition. Absolute interpretation.** Let $\mathbf{A}$ be a combinatory algebra, and let $I : C \to \mathbf{A}$ be an interpretation of constants in $\mathbf{A}$. For each combinatory term $A \in \mathfrak{C}_C$ whose variables are among $\bar{x} = x_1, \ldots, x_n$, we define its ***absolute interpretation*** $[\![A]\!]^{\text{abs}}_{\bar{x}}$ as an element of $\mathbf{A}[\bar{x}]$ by the following inductive clauses. Notice that although the absolute interpretation depends on $I$, we omit the extra superscript.

$$[\![x_i]\!]^{\text{abs}}_{\bar{x}} = x_i \qquad [\![c]\!]^{\text{abs}}_{\bar{x}} = I(c) \qquad [\![\mathbf{K}]\!]^{\text{abs}}_{\bar{x}} = k \qquad [\![\mathbf{S}]\!]^{\text{abs}}_{\bar{x}} = s \qquad [\![AB]\!]^{\text{abs}}_{\bar{x}} = [\![A]\!]^{\text{abs}}_{\bar{x}} \cdot [\![B]\!]^{\text{abs}}_{\bar{x}}$$

We say that the interpretation $I$ ***absolutely satisfies*** the equation $A = B$, in symbols $I \models^{\text{abs}} A = B$, if $[\![A]\!]^{\text{abs}}_{\bar{x}} = [\![B]\!]^{\text{abs}}_{\bar{x}}$, where $FV(A, B) \subseteq \bar{x}$. Notice that, since the canonical homomorphism $\mathbf{A}[\bar{x}] \to \mathbf{A}[\bar{y}]$ is one-to-one for $\bar{x} \subseteq \bar{y}$, this notion is independent of the choice of variables $\bar{x}$. The absolute interpretation of lambda terms $M \in \Lambda_C$ is defined via the translation *cl*:

$$[\![M]\!]^{\text{abs}}_{\bar{x}} \quad = \quad [\![M_{cl}]\!]^{\text{abs}}_{\bar{x}}$$

$$I \models^{\text{abs}} M = N \quad \text{iff} \quad I \models^{\text{abs}} M_{cl} = N_{cl}$$

*Remark.* In the language of universal algebra, $[\![\cdot]\!]^{\text{abs}}_{\bar{x}}$ is just the unique map making the following diagram commute. Hence the local and the absolute interpretation can be defined in any algebraic variety.

$$\begin{tikzcd}
 & \Sigma_C \arrow[r, "\jmath"] \arrow[dd, "\hat{I}"'] & \Sigma_{C+\bar{x}\cdot} \arrow[dd, "{[\![\cdot]\!]^{\text{abs}}_{\bar{x}}}"'] & \\
C \arrow[ur, "\jmath"] \arrow[dr, "I"'] & & & \{\bar{x}\} \arrow[ul, "\jmath"'] \arrow[dl, "\jmath"] \\
 & \mathbf{A} \arrow[r, "\iota"'] & \mathbf{A}[\bar{x}] &
\end{tikzcd}$$

The terminology "an equation holds absolutely" is justified by the following lemma:

**Lemma 2.13.** *Let $A, B \in \mathfrak{C}_C$ be combinatory terms with variables in $\bar{x}$. Let $\mathbf{A}$ be a combinatory algebra and $I : C \to \mathbf{A}$ an interpretation of constants. The following are equivalent:*

1. $I \models^{\text{abs}} A = B$,
2. $\iota \circ I \models A = B$, where $\iota : \mathbf{A} \to \mathbf{A}[\bar{x}]$ is the canonical map,
3. $f \circ I \models A = B$ for all homomorphisms $f \colon \mathbf{A} \to \mathbf{B}$.

*Proof.* 1.⇒3.: Consider $f \colon \mathbf{A} \to \mathbf{B}$ and $\rho \colon \mathcal{V} \to \mathbf{B}$. Let $g \colon \mathbf{A}[\bar{x}] \to \mathbf{B}$ be the unique map extending $f$ such that $g(x_i) = \rho(x_i)$ for all $i$. Then $[\![A]\!]^{f\circ I}_{\rho} = g[\![A]\!]^{\text{abs}}_{\bar{x}}$ and $[\![B]\!]^{f\circ I}_{\rho} = g[\![B]\!]^{\text{abs}}_{\bar{x}}$, hence $[\![A]\!]^{f\circ I}_{\rho} = [\![B]\!]^{f\circ I}_{\rho}$, which proves $f \circ I \models A = B$. 3.⇒2.: Trivial. 2.⇒1.: $\iota \circ I \models A = B$ iff for all $\rho : \mathcal{V} \to \mathbf{A}[\bar{x}]$, $[\![A]\!]^{\iota\circ I}_{\rho} = [\![B]\!]^{\iota\circ I}_{\rho}$. Take $\rho(x_i) = x_i$ to get $[\![A]\!]^{\text{abs}}_{\bar{x}} = [\![B]\!]^{\text{abs}}_{\bar{x}}$. □

**Lemma 2.14.** *In any lambda algebra, $\mathbf{1}(\lambda^* x.A) = \lambda^* x.A$.*

*Proof.* By definition of $\lambda^*$ and Lemma 2.11($b$) and ($c$). □

The next lemma, which is crucial for the soundness of the interpretation of the lambda calculus, holds for absolute, but not for local interpretations.

**Lemma 2.15. The rule ($\xi$) is sound for the absolute interpretation.** *Let $\mathbf{A}$ be a lambda algebra, $I : C \to \mathbf{A}$ an interpretation and $A, B \in \mathfrak{C}_C$ combinatory terms. Then*

$$I \models^{\text{abs}} A = B \qquad \Longleftrightarrow \qquad I \models^{\text{abs}} \lambda^* x.A = \lambda^* x.B$$

*Proof.* Assume the variables of $A$ and $B$ are contained in $x, y_1, \ldots, y_n$.

⇒: Suppose $\mathbf{A}[x, \bar{y}] \models A = B$. Then $\mathbf{A}[x, \bar{y}] \models (\lambda^* x.A)x = A = B = (\lambda^* x.B)x$, hence by Corollary 2.12, $\mathbf{A}[\bar{y}] \models \mathbf{1}(\lambda^* x.A) = \mathbf{1}(\lambda^* x.B)$. The claim follows by Lemma 2.14.

⇐: Suppose $\mathbf{A}[\bar{y}] \models \lambda^* x.A = \lambda^* x.B$. Then $\mathbf{A}[x, \bar{y}] \models A = (\lambda^* x.A)x = (\lambda^* x.B)x = B$. □

It follows from this lemma that the derived lambda abstractor $\lambda^* x$ is a well-defined operator $\lambda^* x : \mathbf{A}[x] \to \mathbf{A}$ if $\mathbf{A}$ is a lambda algebra. When $\mathbf{A}[x]$ is explicitly constructed as $(B, \bullet, K, S)$ like in Section 2.3.1, then $\lambda^* x \colon \mathbf{B} \to \mathbf{A}$ turns out to be the map that sends every element $a$ to itself. Using this $\lambda^*$ operator, the absolute interpretation of a lambda term can be defined directly, *i.e.* without relying on the translation *cl* into combinatory logic:

$$[\![c]\!]^{\text{abs}}_{\bar{x}} = c \qquad [\![x_i]\!]^{\text{abs}}_{\bar{x}} = x_i \qquad [\![MN]\!]^{\text{abs}}_{\bar{x}} = [\![M]\!]^{\text{abs}}_{\bar{x}} \cdot [\![N]\!]^{\text{abs}}_{\bar{x}} \qquad [\![\lambda x.M]\!]^{\text{abs}}_{\bar{x}} = \lambda^* x.[\![M]\!]^{\text{abs}}_{x,\bar{x}}$$

**Proposition 2.16.** *In the category of lambda algebras, the derived lambda abstractor $\lambda^* x : \mathbf{A}[x] \to \mathbf{A}$ is natural in $\mathbf{A}$, i.e. for all $\varphi : \mathbf{A} \to \mathbf{B}$,*

$$\begin{tikzcd}
\mathbf{A}[x] \arrow[r, "{\varphi[x]}"] \arrow[d, "\lambda^* x"'] & \mathbf{B}[x]. \arrow[d, "\lambda^* x"] \\
\mathbf{A} \arrow[r, "\varphi"'] & \mathbf{B}
\end{tikzcd}$$

*Proof.* Any element of $\mathbf{A}[x]$ can be written (not uniquely) as $ax$, where $a \in \mathbf{A}$. Then $\varphi(\lambda^* x.ax) = \varphi(\mathbf{1}a) = \mathbf{1}(\varphi a) = \lambda^* x.(\varphi a)x = \lambda^* x.\varphi[x](ax)$. □

### 2.3.3 Soundness and completeness for lambda algebras

**Proposition 2.17. Soundness.** *The set of equations that hold absolutely in a lambda algebra* $\mathbf{A}$ *is closed under the axioms and rules of the lambda calculus. As a consequence,* $\text{Th}(\mathbf{A})$ *is a lambda theory for any lambda algebra* $\mathbf{A}$.

*Proof.* Consider each axiom and rule of the lambda calculus. $(\alpha)$ and $(\beta)$ are satisfied because $\mathbf{A}$ is a lambda algebra. The rules (*refl*), (*symm*), (*trans*) or (*cong*) are trivially satisfied. Finally, the rule $(\xi)$ is satisfied by Lemma 2.15. For the second claim, notice that a closed equation holds absolutely iff it holds locally. □

**Theorem 2.18. Soundness and Completeness for lambda algebras.** *Let* $\mathcal{E}$ *be a set of closed equations of the lambda calculus. Then for lambda terms* $M, N$,

$$\mathcal{E} \vdash_\beta M = N \quad \textit{if and only if} \quad \mathcal{E} \models_{\mathbf{LA}} M = N.$$

*Proof.* Soundness follows by Proposition 2.17. For completeness, observe that the open term algebra $\Lambda/\mathcal{T}$ associated with the theory $\mathcal{T}$ is a lambda algebra satisfying $M = N$ iff $\mathcal{T} \vdash_\beta M = N$. □

**Corollary 2.19.** *For a set* $\mathcal{E}$ *of closed equations of the lambda calculus, let* $\mathcal{E}_{cl}$ *be its translation into combinatory logic. Then for lambda terms* $M$ *and* $N$,

$$\mathcal{E} \vdash_\beta M = N \quad \textit{if and only if} \quad \mathcal{E}_{cl} \vdash_{LA} M_{cl} = N_{cl}.$$

□

*Remark 2.20.* It is worth noting that Corollary 2.12, Lemma 2.15, and Proposition 2.17 were all proved using only the seven properties of Lemma 2.11. Hence, if a combinatory algebra $\mathbf{A}$ satisfies 2.11$(a)$–$(g)$, then $\text{Th}(\mathbf{A})$ is a lambda theory, which implies that $\mathbf{A}$ is a lambda algebra. Thus, the class of lambda algebras is axiomatized over the class of combinatory algebras by the properties in Lemma 2.11. Of course, these axioms can be closed by using the derived lambda abstractor. However, after spelling everything out in terms of $s$ and $k$, the axioms given in Proposition 2.6 are considerably shorter.

## 2.4 Lambda theories and lambda algebras form equivalent categories

In this section, we define the category of lambda theories, and we show that it is equivalent to the category of lambda algebras.

**Definition.** The category **LT** of lambda theories is defined as follows: An object is a pair $\langle C, \mathcal{T} \rangle$, where $C$ is a set of constants and $\mathcal{T}$ a lambda theory in the language $\Lambda^0_C$. The pair $\langle C, \mathcal{T} \rangle$, like $\mathcal{T}$ itself, is called a ***lambda theory***. A ***translation*** from $C$ to $C'$ is a function $\varphi \colon C \to \Lambda^0_{C'}$. Any such $\varphi$ extends uniquely to a function $\tilde{\varphi} \colon \Lambda^0_C \to \Lambda^0_{C'}$, defined by $\tilde{\varphi} M(c_1, \ldots, c_n) = M(\varphi c_1, \ldots, \varphi c_n)$, where $c_1, \ldots, c_n$ are the constants that appear in $M$. A morphism from $\langle C, \mathcal{T} \rangle$ to $\langle C', \mathcal{T}' \rangle$ is named by a translation from $C$ to $C'$ such that $\mathcal{T} \vdash_\beta M = N$ implies $\mathcal{T}' \vdash_\beta \tilde{\varphi} M = \tilde{\varphi} N$ for all $M, N \in \Lambda^0_C$. $\varphi$ and $\psi$ name the same morphism if $\mathcal{T}' \vdash_\beta \tilde{\varphi} M = \tilde{\psi} M$ for all $M \in \Lambda^0_C$. Composition is defined by $\varphi \circ \psi := \tilde{\varphi} \circ \psi$.

**Theorem 2.21.** *The category* **LT** *of lambda theories is equivalent to the category* **LA** *of lambda algebras.*

*Proof.* We define a pair of functors $F \colon \mathbf{LT} \to \mathbf{LA}$ and $G \colon \mathbf{LA} \to \mathbf{LT}$. $F$ sends a lambda theory $\langle C, \mathcal{T} \rangle$ to its closed term algebra $\Lambda^0_C/\mathcal{T}$, which is always a lambda algebra. $F$ sends a morphism $\varphi \colon \langle C, \mathcal{T} \rangle \to \langle C', \mathcal{T}' \rangle$ to the homomorphism $f \colon \Lambda^0_C/\mathcal{T} \to \Lambda^0_{C'}/\mathcal{T}'$ induced by $\tilde{\varphi} \colon \Lambda^0_C \to \Lambda^0_{C'}$. $G$ sends a lambda algebra $\mathbf{A}$ to $\langle \mathbf{A}, \text{Th}(\mathbf{A}) \rangle$, which is a lambda theory by Proposition 2.17. $G$ sends a homomorphism $f \colon \mathbf{A} \to \mathbf{B}$ to the translation $\varphi \colon \mathbf{A} \to \Lambda^0_{\mathbf{B}}$ with $\varphi a = f a$.

Next, we describe a natural isomorphism $\eta \colon \mathsf{id}_{\mathbf{LA}} \to F \circ G$. For every lambda algebra $\mathbf{A}$, define $\eta_{\mathbf{A}} : \mathbf{A} \to F \circ G(\mathbf{A}) = \Lambda^0_{\mathbf{A}}/\text{Th}(\mathbf{A})$ by $\eta_{\mathbf{A}}(a) = a$. This is clearly a homomorphism, and it is natural in $\mathbf{A}$. To see that this is an isomorphism, notice that for every $M \in \Lambda^0_{\mathbf{A}}$ there is a unique $a \in \mathbf{A}$ with $\text{Th}(\mathbf{A}) \vdash_\beta M = a$, namely, $a = [\![M]\!]$.

In order to show the existence of a natural isomorphism $G \circ F \cong \mathsf{id}_{\mathbf{LT}}$, it now suffices to show that $F$ is full and faithful. $F$ is one-to-one on hom-sets by definition of morphisms in **LT**. $F$ is also full: if $f\colon \Lambda^0_C/\mathcal{T} \to \Lambda^0_{C'}/\mathcal{T}'$ is any homomorphism, then $f$ maps a closed lambda term $M(c_1,\dots,c_n)$ to $M(fc_1,\dots,fc_n)$, where $c_1,\dots,c_n$ are the constants that appear in $M$. This is because $M$ is equivalent to an applicative term made up from $c_1,\dots,c_n$ and the combinators $k$ and $s$, which are preserved by $f$. It follows that $f = F\varphi$, where $\varphi\colon C \to \Lambda^0_{C'}$ is defined by choosing a representative $\varphi(c)$ of $f(c)$, for every $c \in C$. □

## 2.5 Lambda models

The notion of ***lambda model*** arises, as in [5], if one attempts to prove Proposition 2.17 with respect to the equations that hold locally. To do this, one needs the "local" equivalent of Lemma 2.15:

$$\mathbf{A} \models A = B \qquad \Rightarrow \qquad \mathbf{A} \models \lambda^* x.A = \lambda^* x.B.$$

This property, which is called ***weak extensionality***, does not hold in general. Hence one defines a ***lambda model*** to be a weakly extensional lambda algebra.

From our point of view, lambda models can be characterized as those lambda algebras which are intrinsically local: in a lambda model, an equation holds absolutely if and only if it holds locally. Or in other words: in a lambda model, every polynomial is determined by its behavior as a function. In the language of category theory, such a property is called well-pointedness, and indeed lambda models correspond to well-pointed lambda algebras in a sense that will be made precise in Proposition 2.27.

**Proposition 2.22.** *The following are equivalent for a lambda algebra* $\mathbf{A}$*:*

1. $\mathbf{A}$ is weakly extensional.
2. $\mathbf{A}$ satisfies the following ***Meyer-Scott axiom***: for all $a, b \in \mathbf{A}$,

$$\frac{\forall x \in \mathbf{A}. ax = bx}{\mathbf{1}a = \mathbf{1}b}\,(\mathit{MS}), \qquad \text{where } \mathbf{1} = \mathbf{S}(\mathbf{KI}),$$

3. every equation that holds locally in $\mathbf{A}$ already holds absolutely.

*Proof.* 1. $\Rightarrow$ 3.: Let $\mathbf{A}$ be weakly extensional and $\mathbf{A} \models A = B$. Assume $FV(A,B) \subseteq \bar{x}$. By weak extensionality, $\mathbf{A} \models \lambda^*\bar{x}.A = \lambda^*\bar{x}.B$. This is a closed equation, hence $\mathbf{A} \models^{\text{abs}} \lambda^*\bar{x}.A = \lambda^*\bar{x}.B$, and finally $\mathbf{A} \models^{\text{abs}} A = B$ by Lemma 2.15.

3. $\Rightarrow$ 2.: We show (*MS*): Suppose for all $x \in \mathbf{A}$, $ax = bx$. Then $\mathbf{A} \models^{\text{abs}} ax = bx$ by 3., *i.e.* $ax = bx \in \mathbf{A}[x]$. Hence $\mathbf{1}a = \mathbf{1}b$ by Corollary 2.12.

2. $\Rightarrow$ 1.: To show weak extensionality, suppose $\mathbf{A} \models A = B$. Then $\mathbf{A} \models (\lambda^*x.A)x = A = B = (\lambda^*x.B)x$, hence by 2., $\mathbf{A} \models \mathbf{1}(\lambda^*x.A) = \mathbf{1}(\lambda^*x.B)$, hence by Lemma 2.14, $\mathbf{A} \models \lambda^*x.A = \lambda^*x.B$. □

Lambda models are less natural than lambda algebras, because they do not form an algebraic variety. Historically, lambda models were a vehicle for proving soundness and completeness theorems such as Theorem 2.18, see *e.g.* [5, Thm. 5.2.18]. We conclude this section by remarking that every lambda algebra can be embedded in a weakly extensional one:

**Proposition 2.23.** *If* $\mathbf{A}$ *is a lambda algebra and* $X$ *an infinite set, then* $\mathbf{A}[X]$ *is a lambda model.*

*Proof.* We show the Meyer-Scott axiom: assume $a, b \in \mathbf{A}[X]$ and $ax = bx$ for all $x \in \mathbf{A}[X]$. Then there is some finite $Y \subseteq X$ with $a, b \in \mathbf{A}[Y]$. Let $x \in X \setminus Y$, then $ax = bx$ in $\mathbf{A}[Y][x]$, hence $\mathbf{1}a = \mathbf{1}b$ in $\mathbf{A}[Y]$ by Corollary 2.12. □

## 2.6 Models of the lambda-$\beta\eta$-calculus

### 2.6.1 Curry algebras

A ***Curry algebra*** [33] is a lambda algebra with $\mathbf{1} = \mathbf{I}$. Note that Curry algebras form an algebraic variety.

**Proposition 2.24.** *A lambda algebra* $\mathbf{A}$ *is a Curry algebra if and only if* $\mathrm{Th}(\mathbf{A})$ *is a lambda-*$\beta\eta$*-theory.*

*Proof.* If $x \notin \mathrm{FV}(M)$, then $\lambda x.Mx =_\beta (\lambda xy.xy)M = \mathbf{1}_\lambda M$. Hence in any Curry algebra, $\lambda x.Mx = \mathbf{1}M = M$. Conversely, if $\mathrm{Th}(\mathbf{A})$ is a lambda-$\beta\eta$-theory, then $\mathbf{A} \models \mathbf{1} = \lambda xy.xy = \lambda x.x = \mathbf{I}$. □

Hence Curry algebras are to the lambda-$\beta\eta$-calculus what lambda algebras are to the lambda-$\beta$-calculus.

### 2.6.2 Extensional models

An applicative structure is ***extensional*** if for all $a, b \in \mathbf{A}$,

$$\frac{\forall x \in \mathbf{A}.ax = bx}{a = b}.$$

Extensional combinatory algebras are Curry algebras, and hence models of the $\lambda\beta\eta$-calculus. Although extensionality is an intuitive property, extensional models do not form an algebraic variety: *e.g.* the closed term algebra of the lambda-$\beta\eta$-calculus is extensional, but the subalgebra of closed terms is not (see [5, Thm. 20.1.2] and [46]). In fact, a Curry algebra is extensional if and only if it is a lambda model, since the Meyer-Scott axiom from Proposition 2.22 is equivalent to extensionality in the presence of the equation $\mathbf{1} = \mathbf{I}$.

## 2.7 Lambda algebras and categorical models

### 2.7.1 Reflexive ccc models

In this section, we relate the combinatory models of the lambda calculus to the models that arise from a reflexive object in a cartesian-closed category. An object $D$ in a cartesian-closed category $\mathbf{C}$ is called ***reflexive*** if there exists morphisms $e$ and $p$ such that

$$\begin{tikzcd} D^D \arrow[d, "p"'] \arrow[dr, "\mathsf{id}"] & \\ D \arrow[r, "e"'] & D^D \end{tikzcd}$$

The triple $\langle D, e, p\rangle$ is called a ***reflexive* C*-model*** or a ***categorical model of β-conversion***. If also $p \circ e = \mathsf{id}_D$, we speak of a ***categorical model of βη-conversion***.

One defines an interpretation $[\![M]\!]_{x_1,\dots,x_n}$ of each lambda term $M$ with $\mathrm{FV}(M) \subseteq \{x_1, \dots, x_n\}$ as a morphism $D^n \to D$. Assume that bound variables are renamed as appropriate. Recall that $g^\star$ and $e_\star$ are our notations for the curry and uncurry operations, respectively.

$$\begin{array}{rcl}
[\![x_i]\!]_{x_1,\dots,x_n} & = & D^n \xrightarrow{\pi_i} D \qquad \text{(the } i\text{th projection)} \\
[\![MN]\!]_{x_1,\dots,x_n} & = & D^n \xrightarrow{\langle [\![M]\!]_{x_1,\dots,x_n}, [\![N]\!]_{x_1,\dots,x_n}\rangle} D \times D \xrightarrow{e_\star} D \\
[\![\lambda x_{n+1}.M]\!]_{x_1,\dots,x_n} & = & D^n \xrightarrow{([\![M]\!]_{x_1,\dots,x_{n+1}})^\star} D^D \xrightarrow{p} D.
\end{array}$$

**Proposition 2.25.** *The following are properties of categorical models of* $\beta$*-conversion:*

1. **Permutation.** The interpretation is independent of the ordering of the free variables, or of the addition of dummy variables, in the following sense: If $\sigma : \{1,\ldots,n\} \to \{1,\ldots,m\}$ is injective and $\mathrm{FV}(M) \subseteq \{x_{\sigma 1},\ldots,x_{\sigma n}\}$, then

$$\begin{tikzcd} D^m \arrow[rr, "{[\![M]\!]_{x_1,\ldots,x_m}}"] \arrow[dr, "{\langle \pi_{\sigma 1},\ldots,\pi_{\sigma n}\rangle}"'] & & D. \\ & D^n \arrow[ur, "{[\![M]\!]_{x_{\sigma 1},\ldots,x_{\sigma n}}}"'] & \end{tikzcd}$$

2. **Substitution.** Let $\mathrm{FV}(M) \subseteq \{x_1,\ldots,x_n\}$ and $FV(N_1,\ldots,N_n) \subseteq \{y_1,\ldots,y_m\}$, and let $M[\bar{N}/\bar{x}]$ denote the simultaneous substitution of $N_1,\ldots,N_n$ for $x_1,\ldots,x_n$ in $M$. Then

$$\begin{tikzcd} D^m \arrow[rr, "{[\![M[\bar{N}/\bar{x}]]\!]_{\bar{y}}}"] \arrow[dr, "{\langle [\![N_1]\!]_{\bar{y}},\ldots,[\![N_n]\!]_{\bar{y}}\rangle}"'] & & D. \\ & D^n \arrow[ur, "{[\![M]\!]_{x_1,\ldots,x_n}}"'] & \end{tikzcd}$$

3. **Soundness.** If $M =_\beta N$, then $[\![M]\!]_{\bar{x}} = [\![N]\!]_{\bar{x}}$. In a categorical model of $\beta\eta$-conversion, if $M =_{\beta\eta} N$, then $[\![M]\!]_{\bar{x}} = [\![N]\!]_{\bar{x}}$.

*Proof.* 1. and 2. are straightforward by induction on the term $M$. For 3., define $M \sim N$ iff $[\![M]\!]_{\bar{x}} = [\![N]\!]_{\bar{x}}$; by 1., this is independent of the sequence of variables $\bar{x}$, as long as $\mathrm{FV}(M,N) \subseteq \bar{x}$. Clearly, $\sim$ satisfies the properties (*refl*), (*symm*), (*trans*) from Table 2.1. Moreover, it satisfies (*cong*) and ($\xi$) by 2.; to see that it also satisfies ($\beta$), first note that $e_\star \circ (p \times \mathsf{id}_D) = (e \circ p)_\star = (\mathsf{id}_{D^D})_\star = \varepsilon$. Hence

$$\begin{tikzcd} & D \times D \arrow[dr, "e_\star"] & \\ D^n \arrow[ur, "{\langle [\![\lambda x.M]\!]_{\bar{x}},[\![N]\!]_{\bar{x}}\rangle}"] \arrow[r, "{\langle ([\![M]\!]_{\bar{x},x})^\star,[\![N]\!]_{\bar{x}}\rangle}"] \arrow[dr, "{\langle \mathsf{id}_{D^n},[\![N]\!]_{\bar{x}}\rangle}"'] & D^D \times D \arrow[u, "p\times\mathsf{id}_D"] \arrow[r, "\varepsilon"] & D. \\ & D^n \times D \arrow[u, "{\langle ([\![M]\!]_{\bar{x},x})^\star,\mathsf{id}_D\rangle}"] \arrow[ur, "{[\![M]\!]_{\bar{x},x}}"'] & \end{tikzcd}$$

By definition, the composition along the top is $[\![(\lambda x.M)N]\!]_{\bar{x}}$, while by the Substitution Property 2., the composition along the bottom is $[\![M[N/x]]\!]_{\bar{x}}$. This shows $(\lambda x.M)N \sim M[N/x]$. Hence $=_\beta \subseteq \sim$, and we are done with the first claim. The case for $\eta$-conversion follows by a similar diagram chase. □

### 2.7.2 Reflexive ccc models and lambda algebras

From a categorical model $\langle D, e, p\rangle$, one can define a lambda algebra $\langle \mathbf{A}, \cdot, k, s\rangle$:

$$\begin{array}{rcll} \mathbf{A} & = & (1, D) & \text{(the hom-set)} \\ a \cdot b & = & 1 \xrightarrow{\langle a,b\rangle} D \times D \xrightarrow{e_\star} D & \\ k & = & 1 \xrightarrow{[\![\lambda xy.x]\!]} D & \\ s & = & 1 \xrightarrow{[\![\lambda xyz.xz(yz)]\!]} D. & \end{array}$$

**Lemma 2.26.** *$\langle \mathbf{A}, \cdot, k, s\rangle$ is a well-defined lambda algebra.*

*Proof.* It is easy to show by induction on terms that for any combinatory term $A$,

$$[\![A]\!]_\rho \quad = \quad 1 \xrightarrow{\langle \rho x_1,\ldots \rho x_n\rangle} D^n \xrightarrow{[\![A_\lambda]\!]_{x_1,\ldots,x_n}} D.$$

Hence, $A_\lambda =_\beta B_\lambda \Rightarrow [\![A_\lambda]\!]_{\bar{x}} = [\![B_\lambda]\!]_{\bar{x}} \Rightarrow [\![A]\!]_\rho = [\![B]\!]_\rho \Rightarrow \mathbf{A} \models A = B$. ☐

*Remark.* Every lambda algebra arises from a reflexive model. The construction of a cartesian-closed category from a lambda algebra is due to Scott [54], and it is also described in [31].

The following proposition relates various concepts of lambda algebras to corresponding concepts of the categorical interpretation. An object $D$ in a category is ***well-pointed*** if for all $f, g : D \to E$,

$$\left(\forall x.\ \begin{tikzcd} 1 \arrow[r, "x"] & D \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[r, phantom, "\div"] & E \end{tikzcd}\right) \qquad \Rightarrow \qquad f = g.$$

We say that $D$ is ***locally well-pointed*** if the same holds for all $f, g : D \to D$.

**Proposition 2.27.**

| | | | |
|---|---|---|---|
| *1.* | $\mathbf{A}$ *is a lambda model* | *iff* | $D$ *is locally well-pointed.* |
| *2.* | $\mathbf{A}$ *is a Curry algebra* | *iff* | $p \circ e = \mathsf{id}_D$. |
| *3.* | $\mathbf{A}$ *is extensional* | *iff* | $D$ *is locally well-pointed and* $p \circ e = \mathsf{id}_D$. |
| *4.* | $\mathbf{A}[x]$ | $\cong$ | $(1, D^D) \cong (D, D)$. |
| *5.* | $\mathbf{A}[x_1, \ldots, x_n]$ | $\cong$ | $(D^n, D)$. |
| *6.* | $\mathbf{A} \models M = N$ | *iff* | $M = N \colon (1, D)^n \to (1, D)$. |
| *7.* | $\mathbf{A} \models^{\mathsf{abs}} M = N$ | *iff* | $M = N \in (D^n, D)$. |

*Proof.* In $\mathbf{A}$, one first computes $\mathbf{I} = [\![\lambda x.x]\!] = p \circ (\mathsf{id}_D)^\star$, $\mathbf{1} = [\![\lambda xy.xy]\!] = p \circ (p \circ e)^\star$, and for all $a \in \mathbf{A}$, $\mathbf{1} \cdot a = p \circ e \circ a$.

1. $\Rightarrow$: Suppose $\mathbf{A}$ is a lambda model and suppose $f, g : D \to D$ such that for all $x : 1 \to D$, $f \circ x = g \circ x$. Let $\hat{f} = p \circ f^\star$ and $\hat{g} = p \circ g^\star \in \mathbf{A}$. Then $\hat{f} \cdot x = f \circ x = g \circ x = \hat{g} \cdot x$ for all $x \in \mathbf{A}$, hence $\mathbf{1} \cdot \hat{f} = \mathbf{1} \cdot \hat{g} \Rightarrow p \circ e \circ p \circ f^\star = p \circ e \circ p \circ g^\star \Rightarrow f = g$.

   $\Leftarrow$: Suppose $D$ is locally well-pointed. We show the Meyer-Scott axiom (see Proposition 2.22). Suppose $a, b \in \mathbf{A}$ such that for all $x \in \mathbf{A}$, $ax = bx$. This implies for all $x : 1 \to 1 \times D$,

   $$\begin{tikzcd} & & D \times D \arrow[dr, "e_\star"] & \\ 1 \arrow[r, "x"] & 1 \times D \arrow[ur, "a \times \mathsf{id}_D"] \arrow[dr, "b \times \mathsf{id}_D"'] & \div & D, \\ & & D \times D \arrow[ur, "e_\star"'] & \end{tikzcd}$$

   hence, by local well-pointedness, the square commutes. Currying the square, we get $e \circ a = e \circ b$, hence $\mathbf{1} \cdot a = \mathbf{1} \cdot b$.

2. $\Rightarrow$: Suppose $\mathbf{A}$ is a Curry algebra. Then

   $$(p \circ e)^\star = e \circ p \circ (p \circ e)^\star = e \circ \mathbf{1} = e \circ \mathbf{I} = e \circ p \circ (\mathsf{id}_D)^\star = (\mathsf{id}_D)^\star,$$

   hence $p \circ e = \mathsf{id}_D$.

   $\Leftarrow$: Suppose $p \circ e = \mathsf{id}_D$, then $\mathbf{1} = p \circ (p \circ e)^\star = p \circ (\mathsf{id}_D)^\star = \mathbf{I}$.

3. From 1. and 2.

4. and 5. Consider the following two retracts:

$$
\begin{tikzcd}
\mathbf{A}[x] \arrow[d, "\lambda^* x"'] \arrow[dr, "\mathsf{id}"] & \\
\mathbf{A} \arrow[r, "(-)\cdot x"'] & \mathbf{A}[x]
\end{tikzcd}
\qquad \text{and} \qquad
\begin{tikzcd}
D^D \arrow[d, "p"'] \arrow[dr, "\mathsf{id}"] & \\
D \arrow[r, "e"'] & D^D.
\end{tikzcd}
$$

As we have seen in Section 2.3, the set $\mathbf{A}[x]$ can be identified with those $a \in \mathbf{A}$ such that $\mathbf{1} \cdot a = a$. On the other hand, arrows $1 \to D^D$ can be identified with those $a : 1 \to D$ such that $p \circ e \circ a = a$, which is again just $\mathbf{1} \cdot a = a$. This gives a one-to-one correspondence between the points of $\mathbf{A}[x]$ and $(1, D^D) \cong (D, D)$. The correspondence $\mathbf{A}[x_1, \dots, x_n] \cong (D^n, D)$ is similar. Moreover, this correspondence induces a lambda algebra structure on $(D^n, D)$, which turns out to be the natural "pointwise" one given by

$$
\begin{array}{rcl}
a \cdot b & = & D^n \xrightarrow{\langle a,b \rangle} D \times D \xrightarrow{e_\star} D \\
k & = & D^n \to 1 \xrightarrow{[\![\lambda xy.x]\!]} D \\
s & = & D^n \to 1 \xrightarrow{[\![\lambda xyz.xz(yz)]\!]} D.
\end{array}
$$

6. We have $\mathbf{A} \models M = N$ iff $[\![M]\!]_\rho = [\![N]\!]_\rho$ for all $\rho : \mathcal{V} \to (1, D)$, iff $[\![M]\!]_{\bar{x}} \circ f = [\![N]\!]_{\bar{x}} \circ f \in (1, D)$ for all $f \in (1, D^n)$.

7. Follows from 5. and Lemma 2.13. $\square$

# Chapter 3

# Unorderability

The formulation of the untyped lambda calculus, by Church and Curry in the 1930's, has preceded its modern semantic theory by more than 30 years. It was not until the 1960's that Dana Scott constructed the first truly "mathematical" models of the lambda calculus. Scott discovered that such models can be constructed by a combination of order-theoretic and topological methods. Specifically, he observed that there exist non-trivial diagrams of the form

$$\begin{tikzcd} D^D \arrow[d, "p"'] \arrow[dr, "\mathsf{id}"] & \\ D \arrow[r, "e"'] & D^D \end{tikzcd}$$

in certain cartesian-closed categories of complete partial orders and Scott-continuous functions. Recall that an object $D$ in this situation is called *reflexive*, and that it gives rise to a model of the lambda calculus in a canonical way as described in Chapter 2.

The question now arises whether all models of the lambda calculus can be constructed in this way. This question must be modified, since a simple analysis reveals that every reflexive **CPO**-model is uncountable [48], while there are some countable models of the lambda calculus. Instead, one can ask the refined question: can every model of the lambda calculus be *embedded* in a reflexive **CPO**-model? Alternatively: does every lambda theory arise as the theory of a reflexive **CPO**-model? The answer is known to be negative: Honsell and Ronchi Della Rocca [27] have exhibited a lambda theory that does not arise from such a model. One may now further relax this question by asking:

(i) Can every model of the lambda calculus be partially ordered?

(ii) Can every model of the lambda calculus be embedded in one that admits a partial order?

These two questions are the subject of this chapter. Let us call a lambda algebra *unorderable* if it does not admit a non-trivial partial order that is compatible with the algebra structure. Unorderable algebras are known to exist. Plotkin has recently constructed a *finitely separable* algebra, a property with implies unorderability. In Section 3.1, however, we show that one does not have to look very far to find unorderable algebras: the most natural term models of the lambda calculus, namely the standard open and closed term algebras, are unorderable. An application to reflexive **CPO**-models is discussed in Section 3.2.

Question (ii) is more difficult to answer, as indicated by the fact that the answer is still unknown. Let us call a lambda algebra *absolutely unorderable* if it cannot be embedded in an orderable one. In Section 3.3, we give an algebraic characterization of absolutely unorderable $\mathbf{T}$-algebras in any algebraic variety $\mathbf{T}$. We show that a $\mathbf{T}$-algebra is absolutely unorderable if and only if it has a family of so-called *generalized Mal'cev operators*. The question (ii) thereby reduces to the syntactic question whether it is consistent to add such Mal'cev operators to the lambda calculus. This is an open problem in general, but we discuss some special cases in Section 3.4. Finally, in Section 3.5, we relate various different notions of unorderability.

## 3.1 Lambda terms cannot be ordered

In this section, we investigate unorderable models of the lambda calculus. Let us first fix some terminology. Preorders and partial orders were defined in Section 1.2.1. The unique minimal preorder on any set $X$ is called ***discrete***, the unique maximal preorder is called ***indiscrete***, and discrete or indiscrete preorders are called ***trivial***. By convention, if we wish to refer to a preorder that satisfies $x \leqslant y \Rightarrow y \leqslant x$, we will not call it trivial, but ***symmetric***. Of course, a partial order is symmetric iff it is discrete iff it is trivial.

Applicative structures and combinatory algebras were defined in Section 2.2.1. Let $(X, \cdot)$ be an applicative structure. Recall that a preorder $\leqslant$ on $X$ is called ***compatible*** if the application operation is monotone in both arguments with respect to $\leqslant$, *i.e.* if

$$\forall a, b, a', b' \in X.\ a \leqslant a' \text{ and } b \leqslant b' \Rightarrow a \cdot b \leqslant a' \cdot b'.$$

An applicative structure is called ***unorderable*** if it does not allow a non-trivial compatible partial order.

Notice that if $(X, \cdot, k, s)$ is a combinatory algebra, then a preorder $\leqslant$ is compatible if and only if application is monotone in just the second argument. Monotonicity in the first argument then follows by considering $f = \lambda^* x.x \cdot b$, because $a \leqslant a'$ implies $a \cdot b = f \cdot a \leqslant f \cdot a' = a' \cdot b$.

Recall that the ***open term algebra*** of the $\lambda\beta$-calculus is the combinatory algebra $(\Lambda_C/{=_\beta}, \cdot, K, S)$, where $\Lambda_C$ is the set of untyped lambda terms with constants from $C$, $\cdot$ is the application operation on terms, and $K$ and $S$ are the terms $\lambda xy.x$ and $\lambda xyz.xz(yz)$, respectively. The ***closed term algebra*** $(\Lambda^0_C/{=_\beta}, \cdot, K, S)$ is defined analogously, and similarly for the $\lambda\beta\eta$-calculus.

### 3.1.1 Plotkin's unorderable algebra: Separability

In a recent paper [50], G. Plotkin has constructed a finitely separable lambda, a property which implies unorderability. Following an idea of Flagg and Myhill [43], Plotkin calls a subset $A$ of a lambda algebra $X$ ***separable*** if every function $\varphi\colon A \to X$ is realized by some element $\hat{\varphi} \in X$, meaning that for all $a \in A$, $\varphi(a) = \hat{\varphi} \cdot a$. A lambda algebra is said to be ***finitely separable*** if every finite subset is separable. Flagg and Myhill noticed that finitely separable algebras do not allow non-trivial compatible preorders: This is because if $a < b$ are two distinct comparable elements in $X$, then all pairs $x, y$ of elements are comparable via some $\hat{\varphi} \in X$ with $\hat{\varphi} \cdot a = x$ and $\hat{\varphi} \cdot b = y$.

### 3.1.2 The standard term algebras are unorderable

We will now show that the standard open and closed term algebras cannot be non-trivially partially ordered. Notice that these are not finitely separable. For instance, consider the terms $\omega = (\lambda x.xx)(\lambda x.xx)$ and $I = \lambda x.x$. The term $\omega$ is unsolvable, while $I$ is in normal form. Let $T$ be another term in normal form. By the Genericity Lemma (Barendregt [5], Proposition 14.3.24), whenever $R\omega = T$, then $RI = T$. Hence, $\omega$ and $I$ cannot be separated.

How would one go about constructing a partial order on, say, the open term algebra of the $\lambda\beta$-calculus? As a first approximation, one might take two distinct variables $x$ and $y$, and let $\sqsubseteq$ be the preorder generated by a single inequality $x \sqsubseteq y$. For this preorder, one has $M \sqsubseteq N$ iff $N$ is obtained from $M$ up to $\beta$-equivalence by replacing some, but not necessarily all occurrences of the variable $x$ by $y$. In other words, $M \sqsubseteq N$ iff there is a term $P$ (not itself containing $x$ or $y$) such that $M =_\beta Pxxy$ and $N =_\beta Pxyy$. This preorder is non-trivial, because $y \not\sqsubseteq x$. But is it a partial order? The following lemma, to be proved in Section 4.4, shows that this is not the case:

**Lemma 3.1.** *There exists a closed term $A$ of the untyped lambda calculus with $Axxxy =_\beta Axyyy$, but $Axxxy \neq_{\beta\eta} Axxyy \neq_{\beta\eta} Axyyy$ for variables $x \neq y$.* □

Notice that in the preorder that we have just defined, $Axxxy \sqsubseteq Axxyy \sqsubseteq Axyyy = Axxxy$. But since $Axxxy \neq Axxyy$, the preorder $\sqsubseteq$ is not antisymmetric, hence not a partial order. By the same reasoning, $x$ and $y$ cannot be related in *any* compatible partial order on open terms. To show this section's main result, we need to replace the variables $x$ and $y$ by arbitrary terms $u$ and $t$. This is achieved by the following lemma, which states that, if $s$ is a fresh variable, then $su$ and $st$ behave essentially like indeterminates: any equation that holds for $su$ and $st$ will hold for variables $x$ and $y$. Let $\mathcal{T}$ be one of the theories $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$.

**Lemma 3.2.** *Let $u_1, \ldots, u_n$ be terms that are distinct with respect to $\mathcal{T}$. If $s$ is a variable not free in $u_1, \ldots, u_n$, then $su_1, su_2, \ldots, su_n$ behave like generic arguments. More precisely, for all terms $M, N$ with $s \notin \mathrm{FV}(M, N)$, and for variables $x_1, \ldots, x_n$,*

$$\begin{array}{c} M(su_1)(su_2)\ldots(su_n) =_{\mathcal{T}} N(su_1)(su_2)\ldots(su_n) \\ \textit{implies} \\ Mx_1x_2\ldots x_n =_{\mathcal{T}} Nx_1x_2\ldots x_n. \end{array}$$

*Proof.* Let $\{z_1, z_2, \ldots, z_m\}$ be a list of all the free variables of the terms $u_1, \ldots, u_n$. Choose fresh, distinct constants $c_1, \ldots, c_m$ and $d_1, \ldots, d_n$. For $i = 1 \ldots n$, let $\overline{u_i}$ be the closed term obtained from $u_i$ by replacing free variables with the appropriate constants: $\overline{u_i} = u_i[c_1/z_1, \ldots, c_m/z_m]$. Further, add to the lambda calculus a new constant $\sigma \notin C$ and equations $\sigma\overline{u_i} = d_i$, for $i = 1 \ldots n$. Let $\mathcal{T} + \sigma$ denote the theory that is obtained in this way on $\Lambda_{C+\{\sigma\}}$. Then

$$\begin{array}{rcll} M(su_1)(su_2)\ldots(su_n) & =_{\mathcal{T}} & N(su_1)(su_2)\ldots(su_n) & \\ \Rightarrow\ M(\sigma\overline{u_1})\ldots(\sigma\overline{u_n}) & =_{\mathcal{T}+\sigma} & N(\sigma\overline{u_1})\ldots(\sigma\overline{u_n}) & \text{(by renaming)} \\ \Rightarrow\ Md_1\ldots d_n & =_{\mathcal{T}+\sigma} & Nd_1\ldots d_n & \end{array}$$

The claim now follows from the fact that $\mathcal{T} + \sigma$ is conservative over $\mathcal{T}$. This is a consequence of Plotkin's separability result [50]: the closed term algebra can be embedded in a separable algebra. Let $\iota : \Lambda^0_C/\mathcal{T} \to \mathbf{A}$ be such an embedding. Then choose $\hat{\varphi} \in \mathbf{A}$ such that $\hat{\varphi} \cdot \iota u_i = \iota d_i$, for $i = 1 \ldots n$. There is a unique extension $\iota' : \Lambda^0_{C+\sigma}/\mathcal{T} \to \mathbf{A}$ of $\iota$, sending $\sigma$ to $\hat{\varphi}$. Clearly, the theory induced by $\iota'$ is a conservative extension of $\mathcal{T}$ satisfying the additional equations. $\square$

**Theorem 3.3.** *Let $\mathcal{M}$ be the open or the closed term algebra of the $\lambda\beta$- or $\lambda\beta\eta$-calculus. Then $\mathcal{M}$ does not allow a non-trivial compatible partial order.*

*Proof.* Let $\leqslant$ be a compatible partial order on $\mathcal{M}$. Let $u \neq t \in \mathcal{M}$, and assume, by way of contradiction, that $u \leqslant t$. Let $A$ be as in Lemma 3.1, and let $s$ be a fresh variable. Then by compatibility,

$$\begin{array}{rcl} \lambda s.A(su)(su)(su)(st) & \leqslant & \lambda s.A(su)(su)(st)(st) \\ & \leqslant & \lambda s.A(su)(st)(st)(st) \\ & = & \lambda s.A(su)(su)(su)(st), \end{array}$$

hence, by antisymmetry,

$$A(su)(su)(su)(st) = A(su)(su)(st)(st)$$

Applying Lemma 3.2 to $M = \lambda xy.Axxxy$ and $N = \lambda xy.Axxyy$, one gets $Axxxy = Axxyy$ for variables $x$ and $y$, contradicting the choice of $A$. Consequently, the order is trivial. $\square$

## 3.2 The Topological Completeness Problem

Recall that, for any cartesian-closed category $\mathbf{C}$, a ***reflexive* C*-model*** is a model of the lambda calculus that arises from a diagram

$$\begin{tikzcd} D^D \arrow[d, "p"'] \arrow[dr, "\mathsf{id}"] & \\ D \arrow[r, "e"'] & D^D \end{tikzcd}$$

in the category **C** (see Section 2.7). These models have been particularly well studied in the category **CPO** of cpo's and Scott-continuous functions. Reflexive **CPO**-models are sometimes referred to as ***continuously complete***, because every Scott-continuous function $f : D \to D$ is definable by an element $\hat{f} \in D$. Honsell and Ronchi Della Rocca [27] also use the term ***topological model***. The following is a long standing open problem ([27]):

**Open Problem. (Topological Completeness)** Is there a reflexive **CPO**-model whose theory is $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$?

Two related questions have been answered: Honsell and Ronchi Della Rocca [27] have shown that there is a lambda theory $\mathcal{C}_{\Lambda_0}$ which is not induced by any reflexive **CPO**-model. The reflexive **CPO**-models are thus incomplete for arbitrary lambda theories. On the other hand, Di Gianantonio *et al.* [16] have shown that $\underline{\lambda\beta\eta}$ can arise as the theory of a reflexive $\mathbf{CPO}_1$-model. If $\omega_0$ and $\omega_1$ denote, respectively, the first infinite ordinal and the first uncountable ordinal, then $\mathbf{CPO}_1$ is the category whose objects are $\omega_0$- and $\omega_1$-complete partial orders, and whose morphisms preserve limits of $\omega_1$-chains (but not necessarily of $\omega_0$-chains). However, the construction given in [16] makes decisive use of non-Scott-continuous functions.

We will now explore some consequences of Theorem 3.3 for topological completeness. First, one notices that in all models whose theory is $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$, the denotations of closed lambda terms necessarily form a discrete subset:

**Corollary 3.4.** *In any partially ordered lambda algebra whose theory is $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$, the denotations of closed terms are pairwise incomparable.*

*Proof.* The set of closed term denotations is a sub-lambda algebra which is isomorphic to the closed term algebra; hence the partial order is discrete on it by Theorem 3.3. □

Recall that two elements $x, y$ of a partially ordered set $D$ are called ***compatible*** if there exists $z \in D$ with $x \leqslant z$ and $y \leqslant z$. We can now show that any complete reflexive $\mathbf{CPO}_\bot$-model, if such a model exists, must satisfy one of two peculiar properties:

**Theorem 3.5.** *Suppose $D$ is a reflexive $\mathbf{CPO}_\bot$-model whose theory is $\underline{\lambda\beta}$ or $\underline{\lambda\beta\eta}$. Then either:*

1. *The denotations of closed terms are pairwise incompatible, or*
2. *There exist closed lambda terms $M$ and $N$ such that, for all $x, y \in D$,*

$$(x \leqslant y \text{ or } y \leqslant x) \qquad \Longleftrightarrow \qquad Mxy = Nxy.$$

*Proof.* Suppose 1. does not hold, *i.e.* there are two distinct closed terms $u$ and $t$ whose denotations have an upper bound $v \in D$. Let $A$ be as in Lemma 3.1, and let $M = \lambda xyrs.A(s(rx))(s(rx))(s(rx))(s(ry))$ and $N = \lambda xys.A(s(rx))(s(rx))(s(ry))(s(ry))$.

$\Rightarrow$: Suppose $x \leqslant y$ or $y \leqslant x$. Then $Mxy = Nxy$ by the same reasoning as in the proof of Theorem 3.3.

$\Leftarrow$: Suppose, by way of contradiction, that $Mxy = Nxy$ for incomparable $x, y \in D$. Define $r : D \to D$ by

$$r(z) = \begin{cases} \bot & \text{if } z \leqslant x \text{ and } z \leqslant y \\ u & \text{if } z \leqslant x \text{ and } z \not\leqslant y \\ t & \text{if } z \not\leqslant x \text{ and } z \leqslant y \\ v & \text{if } z \not\leqslant x \text{ and } z \not\leqslant y \end{cases}$$

Then $r$ is continuous; suppose it is represented by $\hat{r} \in D$. Then

$$D \models \lambda s.A(su)(su)(su)(st) = Mxy\hat{r} = Nxy\hat{r} = \lambda s.A(su)(su)(st)(st).$$

But the proof of Theorem 3.3 shows that the first and the last term are $\underline{\lambda\beta\eta}$-different, contradicting the assumption that $D$ was a complete model. □

## 3.3 A characterization of absolutely unorderable algebras

In Section 3.1, we have shown that the combinatory algebra of open lambda terms cannot be non-trivially ordered. However, it can be *embedded* in an orderable algebra; this follows *e.g.* from the work of Di Gianantonio *et al.* [16]. Plotkin conjectures in [50] that there exists a combinatory algebra which is *absolutely unorderable*, *i.e.* which cannot be embedded in an orderable combinatory algebra. In this section, we characterize, for any algebraic variety $\mathbf{T}$, those $\mathbf{T}$-algebras which are absolutely unorderable.

Let $\mathbf{T}$ be an algebraic variety. Recall that a preorder $\leqslant$ on a $\mathbf{T}$-algebra $\mathbf{A}$ is ***compatible*** if whenever $a_i \leqslant b_i$ for $i = 1 \ldots k$, then $fa_1 \ldots a_k \leqslant fb_1 \ldots b_k$, for each $k$-ary function symbol $f \in \Omega_k$. Notice that monotone preorders are closed under arbitrary intersections. If $\leqslant$ is monotone, then so is the dual preorder $\geqslant$. Every monotone preorder determines a congruence $\sim_{\leqslant}$ on $\mathbf{A}$, which is the intersection of $\leqslant$ and $\geqslant$. Also notice that $\leqslant$ naturally defines a partial order on $\mathbf{A}/{\sim_{\leqslant}}$.

A $\mathbf{T}$-algebra $\mathbf{A}$ is said to be ***unorderable*** if it does not allow a non-trivial compatible partial order. Also, $\mathbf{A}$ is said to be ***absolutely unorderable*** if for any embedding $\mathbf{A} \to \mathbf{B}$ of $\mathbf{T}$-algebras, $\mathbf{B}$ is unorderable.

### 3.3.1 Absolutely unorderable algebras and generalized Mal'cev operators

Consider a $\mathbf{T}$-algebra $\mathbf{A}$. Let $\preccurlyeq$ be the smallest compatible preorder on $\mathbf{A}[u,t]$ such that $u \preccurlyeq t$.

**Lemma 3.6.** *$\preccurlyeq$ is trivial on $\mathbf{A}$, i.e. $a \preccurlyeq b \Rightarrow a = b$ for $a, b \in \mathbf{A}$.*

*Proof.* Let $\sim$ be the kernel of the canonical morphism $\mathbf{A}[u,t] \to \mathbf{A}[x]$ which sends both $u$ and $t$ to $x$. Then $\sim$ is a congruence, hence in particular a compatible preorder on $\mathbf{A}[u,t]$. Since also $u \sim t$, by definition $\preccurlyeq$ is contained in it. But $\sim$, hence $\preccurlyeq$, is trivial on $\mathbf{A}$. $\square$

**Lemma 3.7.** *$\mathbf{A}$ is absolutely unorderable if and only if $t \preccurlyeq u$.*

*Proof.* $\Rightarrow$: Suppose $\mathbf{A}$ is absolutely unorderable. Consider the natural map $\mathbf{A} \to \mathbf{A}[u,t] \to \mathbf{A}[u,t]/{\sim_{\preccurlyeq}}$. Lemma 3.6 implies that, the composition is an embedding, hence $\preccurlyeq$ must be trivial as a partial order on $\mathbf{A}[u,t]/{\sim_{\preccurlyeq}}$. Equivalently, $\preccurlyeq$ as a preorder on $\mathbf{A}[u,t]$ is symmetric. Since $u \preccurlyeq t$, it follows that $t \preccurlyeq u$.

$\Leftarrow$: Suppose $\mathbf{A}$ is not absolutely unorderable. Then there is an embedding $F : \mathbf{A} \to \mathbf{B}$ of $\mathbf{T}$-algebras where $\mathbf{B}$ has a non-trivial compatible partial order $\leqslant$. Hence there are $U \neq T \in \mathbf{B}$ such that $U \leqslant T$. Consider the unique map $G : \mathbf{A}[u,t] \to \mathbf{B}$ such that $u \mapsto U$, $t \mapsto T$ and $G|_{\mathbf{A}} = F$. Define $a \leqslant b$ in $\mathbf{A}[u,t]$ iff $G(a) \leqslant G(b)$ in $\mathbf{B}$. Then $\leqslant$ is a compatible preorder on $\mathbf{A}[u,t]$. But $u \leqslant t$, hence $\preccurlyeq$ is contained in $\leqslant$. But $t \not\leqslant u$, hence $t \not\preccurlyeq u$. $\square$

Further, $\preccurlyeq$ has the following explicit description: On $\mathbf{A}[u,t]$, define $a \lhd b$ if and only if there is a polynomial $A(x_1, x_2, x_3) \in \mathbf{A}[x_1, x_2, x_3]$ such that $A(t,u,u) = a$ and $A(t,t,u) = b$.

**Lemma 3.8.** *$\preccurlyeq$ is the transitive closure of $\lhd$.*

*Proof.* Notice that $\lhd$ is reflexive. Let $\lhd^*$ be the transitive closure. Clearly, $\lhd$, and hence $\lhd^*$, is contained in $\preccurlyeq$. On the other hand, $\lhd^*$ is a preorder on $\mathbf{A}[u,t]$ satisfying $u \lhd^* t$. To see that $\lhd^*$ is compatible, let $f$ be a $k$-ary function symbol in $\Sigma$. First assume $a_i \lhd b_i$ for $i = 1 \ldots k$. Then there are $A_i(x_1, x_2, x_3) \in \mathbf{A}[x_1, x_2, x_3]$, for $i = 1 \ldots k$, such that $A_i(t,u,u) = a_i$ and $A_i(t,t,u) = b_i$ for each $i$. By considering $A(x_1, x_2, x_3) = f(A_1(x_1, x_2, x_3) \ldots A_k(x_1, x_2, x_3))$, it follows that $fa_1 \ldots a_k \lhd fb_1 \ldots b_k$. Hence $\lhd$ is compatible, which readily implies compatibility for $\lhd^*$. Therefore, $\preccurlyeq$ is contained in $\lhd^*$. $\square$

Putting this together with Lemma 3.7, we get the following characterization of absolutely unorderable algebras. Recall that an equation $p(u,t) = q(u,t)$ holds ***absolutely*** in $\mathbf{A}$ iff it holds in $\mathbf{A}[u,t]$.

**Theorem 3.9. Characterization of absolutely unorderable T-algebras.** *Let* $\mathbf{T}$ *be an algebraic variety. A* $\mathbf{T}$*-algebra* $\mathbf{A}$ *is absolutely unorderable if and only if, for some* $n \geqslant 1$*, there exist polynomials* $\mathbf{M}_i(x_1, x_2, x_3) \in \mathbf{A}[x_1, x_2, x_3]$*, for* $i = 1 \ldots n$*, such that the following equations hold absolutely in* $\mathbf{A}$*:*

$$\begin{array}{rcl} t & = & \mathbf{M}_1(t,u,u) \\ \mathbf{M}_1(t,t,u) & = & \mathbf{M}_2(t,u,u) \\ \mathbf{M}_2(t,t,u) & = & \mathbf{M}_3(t,u,u) \\ & \vdots & \\ \mathbf{M}_n(t,t,u) & = & u \end{array} \tag{3.1}$$

*Proof.* By Lemmas 3.7 and 3.8, $\mathbf{A}$ is absolutely unorderable if and only if there are $t_1, \ldots, t_{n-1} \in \mathbf{A}[u,t]$ such that $t \lhd t_1 \lhd \ldots \lhd t_n \lhd u$. The corollary follows by definition of $\lhd$. □

In the case $n = 1$, the equations (3.1) have the simple form $t = \mathbf{M}(t,u,u)$ and $\mathbf{M}(t,t,u) = u$. These equations were first studied by A.I. Mal'cev [37] to characterize varieties of congruence-permutable algebras (so-called Mal'cev varieties). A ternary operator $\mathbf{M}$ satisfying these equations is called a Mal'cev operator. Accordingly, we call $\mathbf{M}_1, \ldots, \mathbf{M}_n$ satisfying (3.1) a family of ***(generalized) Mal'cev operators***, and we call the equations (3.1) the ***(generalized) Mal'cev axioms***. Hagemann and Mitschke [25] have shown that an algebraic variety has $n$-permutable congruences if and only if it has operators satisfying the axioms (3.1). It was proved by W. Taylor [63, 11] that algebras in a variety with $n$-permutable congruences are unorderable; however, the converse, to the best of my knowledge, is a new result. Also note that Theorem 3.9 characterizes *individual* algebras that are absolutely unorderable, rather than varieties of unorderable algebras.

### 3.3.2 An application to ordered algebras and dcpo-algebras

Recall from Section 1.3.5 that an algebraic signature $\Sigma$ and a set of inequations $\mathcal{I}$ define a variety $\mathbf{O}$ of ordered algebras. The free ordered $(\Sigma,\mathcal{I})$-algebra over any poset $P$ was denoted by $\mathbf{O}_P$. One may ask under which circumstances the canonical map $\jmath : P \to \mathbf{O}_P$ is order-reflecting. The following theorem shows that the answer depends only on the presence of Mal'cev operators in $(\Sigma,\mathcal{I})$. Recall that a $k$-ary operation in $\Sigma$ is simply a $\Sigma$-term $t(x_1, \ldots, x_k)$.

**Theorem 3.10.** *Let* $\Sigma$ *be a signature and* $\mathcal{I}$ *a set of inequations. Let* $P$ *be a non-trivially ordered poset and let* $\jmath : P \to \mathbf{O}_P$ *be the canonical map from* $P$ *into the free ordered* $(\Sigma,\mathcal{I})$*-algebra over* $P$*. The following are equivalent:*

1. $\jmath$ is *not* order-reflecting.
2. Every ordered $(\Sigma,\mathcal{I})$-algebra is trivially ordered.
3. There are ternary operations $\mathbf{M}_1, \ldots, \mathbf{M}_n$ in $\Sigma$ such that $\mathcal{I}$ entails

$$\begin{array}{rcl} t & \leqslant & \mathbf{M}_1(t,u,u) \\ \mathbf{M}_1(t,t,u) & \leqslant & \mathbf{M}_2(t,u,u) \\ \mathbf{M}_2(t,t,u) & \leqslant & \mathbf{M}_3(t,u,u) \\ & \vdots & \\ \mathbf{M}_n(t,t,u) & \leqslant & u \end{array} \tag{3.2}$$

*Proof.* 1. ⇒ 2.: Suppose $\mathbf{B}$ is a non-trivially ordered $(\Sigma,\mathcal{I})$-algebra with elements $a < b$. We show that $\jmath$ is order-reflecting. Let $x, y \in P$ with $\jmath(x) \leqslant \jmath(y)$. Define $g : P \to \mathbf{B}$ by

$$g(z) = \begin{cases} a & \text{if } z \leqslant y \\ b & \text{if } z \not\leqslant y \end{cases}$$

Then $g$ is monotone; therefore, by the universal property of $\mathbf{O}_P$, there exists a unique homomorphism of ordered algebras $h : \mathbf{O}_P$ such that $g = h \circ \jmath$. By monotonicity of $h$, we get $g(x) = h(\jmath(x)) \leqslant h(\jmath(y)) = g(y) = a$, hence $x \leqslant y$.

2. $\Rightarrow$ 3.: Suppose every ordered $(\Sigma, \mathcal{I})$-algebra is trivially ordered. Then, in particular, $\mathbf{O}_{\mathcal{V}}$ is trivially ordered, and hence $\mathcal{I} \vdash_{\mathit{ineq}} s \leqslant t$ iff $\mathcal{I} \vdash_{\mathit{ineq}} t \leqslant s$. We can therefore regard $\mathcal{I}$ as a set of equations. The claim follows by applying Theorem 3.9 to $\mathbf{A} = \mathbf{O}_\emptyset$.

3. $\Rightarrow$ 2.: Suppose $(\Sigma, \mathcal{I})$ has operators satisfying (3.2). Then for any $(\Sigma, \mathcal{I})$-algebra $\mathbf{B}$, if $a \leqslant b \in \mathbf{B}$, then $b \leqslant \mathbf{M}_1(b, a, a) \leqslant \mathbf{M}_1(b, b, a) \leqslant \ldots \leqslant \mathbf{M}_n(b, b, a) \leqslant a$, hence $\mathbf{B}$ is trivially ordered.

2. $\Rightarrow$ 1.: A map $\jmath : P \to \mathbf{O}_P$ from a non-trivially ordered set into a trivially ordered one cannot be order-reflecting. □

*Remark.* Notice that the proof of 3. $\Rightarrow$ 2. shows that the inequalities (3.2) already imply the corresponding equalities (3.1).

The equivalent of Theorem 3.10 holds for dcpo-algebras as well. This is due to the following lemma, which relates the existence of non-trivial dcpo-algebras to the existence of non-trivial ordered algebras:

**Lemma 3.11.** *Let $\Sigma$ be a signature and $\mathcal{I}$ a set of inequations. There exists a non-trivially ordered $(\Sigma, \mathcal{I})$-dcpo-algebra if and only if there exists a non-trivially ordered $(\Sigma, \mathcal{I})$-algebra.*

*Proof.* $\Rightarrow$: Trivial, since every dcpo-algebra is an ordered algebra.

$\Leftarrow$: Let $\langle \mathbf{A}, \leqslant \rangle$ be an ordered $(\Sigma, \mathcal{I})$-algebra. We consider the ideal completion of $\mathbf{A}$: A subset $I \subseteq \mathbf{A}$ is an ***ideal*** if it is downward closed and directed. Let $\mathcal{J} = \mathrm{Idl}(\mathbf{A})$, the ***ideal completion*** of $\mathbf{A}$, be the set of all ideals, ordered by inclusion. Abramsky and Jung [3] prove that $\mathcal{J}$ is a $(\Sigma, \mathcal{I})$-dcpo-algebra. Moreover, the map $\mathbf{A} \to \mathcal{J} : x \mapsto \downarrow x$ is order preserving and reflecting, and hence $\mathcal{J}$ is non-trivially ordered if $\mathbf{A}$ is.

**Corollary 3.12.** *Let $\Sigma$ be a signature and $\mathcal{I}$ a set of inequations. Let $D$ be a non-trivially ordered dcpo and let $\jmath : D \to \mathbf{D}_D$ be the canonical map from $D$ into the free ordered $(\Sigma, \mathcal{I})$-algebra over $D$. The following are equivalent:*

1. $\jmath$ is *not* order-reflecting.
2. Every $(\Sigma, \mathcal{I})$-dcpo-algebra is trivially ordered.
3. There are ternary operations $\mathbf{M}_1, \ldots, \mathbf{M}_n$ in $\Sigma$ such that $\mathcal{I}$ entails (3.2).

*Proof.* The equivalence of 2. and 3. follows from Theorem 3.10 and Lemma 3.11. The implication 2. $\Rightarrow$ 1. is trivial, and 1. $\Rightarrow$ 2. follows as in the proof of Theorem 3.10; notice that the function $g$ defined there is continuous. □

## 3.4 Absolutely unorderable combinatory algebras

If $\mathbf{A}$ is a combinatory algebra, then the statement of Theorem 3.9 takes a particularly simple form, due to combinatory completeness: A combinatory algebra $\mathbf{A}$ is absolutely unorderable if and only if there are a number $n \geqslant 1$ and elements $\mathbf{M}_1, \ldots, \mathbf{M}_n \in \mathbf{A}$ such that the following hold absolutely in $\mathbf{A}$:

$$\begin{array}{rcl} t & = & \mathbf{M}_1 tuu \\ \mathbf{M}_1 ttu & = & \mathbf{M}_2 tuu \\ \mathbf{M}_2 ttu & = & \mathbf{M}_3 tuu \\ & \vdots & \\ \mathbf{M}_n ttu & = & u \end{array} \tag{3.3}$$

Note that if $\mathbf{A}$ is a lambda algebra, one can replace these equations by the closed equations $\lambda^* tu.t = \mathbf{M}_1 tuu$ etc.

But does an absolutely unorderable combinatory algebra exist? Unfortunately, this is not known. Clearly, an absolutely unorderable combinatory algebra exists if and only if the equations (3.3) are consistent with the axioms of combinatory algebras for some $n$. The answer is only known in the cases $n = 1$ and $n = 2$. In these cases, (3.3) is inconsistent with combinatory logic, as we will now show. Notice that if the axioms are consistent for some $n$, then also for all $m \geqslant n$, by letting $\mathbf{M}_{n+1}, \dots, \mathbf{M}_m = \lambda xyz.z$

Let $Y$ be any fixpoint operator of combinatory logic, for instance the paradoxical fixpoint combinator $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$. Write $\mu x.M$ for $Y(\lambda x.M)$. The operator $\mu$ satisfies the fixpoint property:

$$\mu x.A(x) = A(\mu x.A(x)). \qquad \textit{(fix)}$$

The ***diagonal axiom*** is

$$\mu y.\mu x.A(x, y) = \mu x.A(x, x). \qquad (\Delta)$$

The following lemma is due to G. Plotkin and A. Simpson:

**Lemma 3.13 (Plotkin, Simpson).** *Assuming the diagonal axiom, the Mal'cev axioms (3.3) are inconsistent with combinatory logic for all $n$.*

*Proof.* Let $x$ be arbitrary. Let $A = \mu z.\mathbf{M}_1 xzz$. Then $A = \mu z.x = x$. Also,

$$\begin{array}{rcll} x \;=\; A & = & \mu z.\mathbf{M}_1 xzz & \\ & = & \mu y.\mu z.\mathbf{M}_1 xyz & \text{by } (\Delta) \\ & = & \mu z.\mathbf{M}_1 xxz & \text{by } \textit{(fix)} \\ & = & \mu z.\mathbf{M}_2 xzz & \text{by } (3.3) \\ & = & \dots & \\ & = & \mu z.\mathbf{M}_{n-1} xxz & \\ & = & \mu z.z & \text{by } (3.3) \end{array}$$

Hence $x = \mu z.z$ for all $x$, which is an inconsistency. □

**Theorem 3.14 (Plotkin, Simpson).** *For $n = 1$, the Mal'cev axioms are inconsistent with combinatory logic.*

*Proof.* Suppose $\mathbf{M}$ is a Mal'cev operator. Let $x$ be arbitrary and let $A = \mu y.\mu z.\mathbf{M}xyz$. Then

$$A \overset{\textit{(fix)}}{=} \mu z.\mathbf{M}xAz \overset{\textit{(fix)}}{=} \mathbf{M}xAA \overset{(Malcev_1)}{=} x,$$

hence $x = \mu z.\mathbf{M}xAz = \mu z.\mathbf{M}xxz = \mu z.z$. □

**Theorem 3.15 (Plotkin, Selinger).** *For $n = 2$, the Mal'cev axioms are inconsistent with combinatory logic.*

*Proof.* Suppose $\mathbf{M}_1$ and $\mathbf{M}_2$ are operators satisfying the Mal'cev axioms (3.3). Define $A$ and $B$ by mutual recursion such that

$$\begin{array}{rcl} A & = & \mu x.f(\mathbf{M}_1 xAB)(\mathbf{M}_1 xAB) \\ B & = & \mu y.\mu z.f(\mathbf{M}_2 ABy)(\mathbf{M}_2 ABz). \end{array}$$

Then

$$\begin{array}{rcll} B & = & f(\mathbf{M}_2 ABB)(\mathbf{M}_2 ABB) & \text{by } \textit{(fix)} \\ & = & f(\mathbf{M}_1 AAB)(\mathbf{M}_1 AAB) & \text{by } (3.3) \\ & = & A. & \text{by } \textit{(fix)} \end{array}$$

So $\mu x.fxx = \mu x.f(\mathbf{M}_1 xAA)(\mathbf{M}_1 xAA) = A = B = \mu y.\mu z.f(\mathbf{M}_2 AAy)(\mathbf{M}_2 AAz) = \mu y.\mu z.fyz$, which is the diagonal axiom. By Lemma 3.13, this leads to an inconsistency. □

## 3.5 Relating different notions of unorderability

### 3.5.1 Local notions

We defined a combinatory algebra to be unorderable if it does not allow a non-trivial compatible partial order. There are other notions of unorderability that are worth investigating. For instance, one can ask for the existence of non-symmetric preorders instead of partial orders. Or one can ask for the (pre)order to be compatible with abstraction as well as with application: $\leqslant$ is called a ***lambda-(pre)order*** if it is compatible and

$$\frac{\forall x \in \mathbf{A}.ax \leqslant bx}{\mathbf{1}a \leqslant \mathbf{1}b}.$$

We thus arrive at the following four unorderability notions for a combinatory algebra $\mathbf{A}$:

1. ***unorderable*** if every compatible partial order on $\mathbf{A}$ is trivial.
2. ***un-preorderable*** if every compatible preorder on $\mathbf{A}$ is symmetric.
3. ***un-$\boldsymbol{\lambda}$-orderable*** if every lambda-order on $\mathbf{A}$ is trivial.
4. ***un-$\boldsymbol{\lambda}$-preorderable*** if every lambda-preorder on $\mathbf{A}$ is symmetric.

Between these notions, only the obvious implications hold:

$$\begin{tikzcd}
& \text{unorderable} \arrow[dr, Rightarrow] & \\
\text{un-preorderable} \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & & \text{un-}\lambda\text{-orderable} \\
& \text{un-}\lambda\text{-preorderable} \arrow[ur, Rightarrow] &
\end{tikzcd}$$

To see that no other implications hold, first observe that the open term algebra is unorderable, but $\lambda$-pre-orderable: let $M \leqslant N$ iff for all valuations $\rho$, $[\![M]\!]_\rho \leqslant [\![N]\!]_\rho$ in the standard $D_\infty$-model. It follows from [29, 64] that this preorder is non-trivial; it is a lambda-preorder because the order on the standard $D_\infty$-model is pointwise.

The counterexample in the other direction is due to G. Plotkin, and it is given in the following theorem:

**Theorem 3.16. (G. Plotkin)** *There is an extensional, partially ordered lambda algebra* $\mathbf{A}$ *that does not allow a non-trivial lambda-preorder.*

*Proof.* The idea of the construction is to work in a category where the order relation on function spaces is not pointwise. We use the category of meet cpo's and stable functions $\mathbf{CPO}^{\wedge}$, which was defined in Section 1.2.4. Recall that the objects of this category are cpo's with bounded binary meets which act continuously, and that the morphisms are stable functions, *i.e.* continuous functions preserving the bounded meets. As we outlined in Section 1.2.5, the usual Scott $D_\infty$-construction of models of the lambda-$\beta\eta$-calculus goes through in this category.

Let $D_0$ be the cpo with two elements $\bot \leqslant \top$. Define $D_{n+1}$ to be the stable function space $D_n^{D_n}$. Then $D_1$ has three elements $B, I, T$, where $B$ is the constantly $\bot$ function, $T$ is the constantly $\top$ function, and $I$ is the identity. Notice that the stable order on $D_1$ is as shown:

$$\begin{tikzcd}
\top & & T \arrow[dr, dash] & & I \arrow[dl, dash] \\
\bot \arrow[u, dash] & & & B & \\
D_0 & & & D_1 &
\end{tikzcd}$$

Define $e_{01} : D_0 \to D_1$ to send $\top$ to $T$ and $\perp$ to $B$, and define $p_{10} : D_1 \to D_0$ to send $f$ to $f(\perp)$. The pair $\langle e_{01}, p_{10}\rangle$ is is an embedding-projection pair in the category **CPO**$^\wedge$, in particular, $e_{01} \circ p_{10}$ is stably less than the identity. From this, one constructs the other embedding projection pairs and takes the inverse limit $D$ as usual. Then $D \cong D^D$, and as a lambda algebra, $D$ is extensional by Proposition 2.27. Clearly the order $\leqslant$ on $D$ is non-trivial and compatible.

For convenience, we identify all relevant function spaces with the corresponding subspaces of $D$. Let $p_n : D \to D_n$ be the projection of $D$ onto $D_n$, and $e_n$ the corresponding embedding.

Now suppose that $\sqsubseteq$ is a lambda-preorder. We will show it is trivial, *i.e.* it is either discrete or indiscrete. First notice that, since we are in an extensional model, $\mathbf{1} = \mathbf{I}$ and hence

$$\frac{\forall x \in D. ax \sqsubseteq bx}{a \sqsubseteq b}. \tag{3.4}$$

Chasing the definition of the $D_\infty$-model, one calculates that for $f \in D_{n+1} = D_n^{D_n}$ and $x \in D$, the application $f \cdot x$ is given by $e_n \circ f \circ p_n(x)$. From this and (3.4), it follows that $f \sqsubseteq g \in D_{n+1}$ iff $f \cdot x \sqsubseteq g \cdot x$ for all $x \in D_n$. One distinguishes three cases:

**Case 1:** $I \sqsubseteq T$. For any pair of elements $x, y \in D$, define $f : D_1 \to D$ by $f(I) = x$ and $f(T) = y$ and $f(B) = \perp$. This is stable and therefore realized by some $\hat{f} \in D$. We get that $x \sqsubseteq y$ and hence $\sqsubseteq$ is indiscrete.

**Case 2:** $T \sqsubseteq I$. Similar.

**Case 3:** Neither $I \sqsubseteq T$ nor $T \sqsubseteq I$. Suppose, by way of contradiction, that there are distinct elements $x, y \in D$ such that $x \sqsubseteq y$. Then for some $n$ the projections $x_n = p_n(x)$ and $y_n = p_n(y)$ are distinct. Since the projection $p_n$ itself is realized by some $\hat{p}_n \in D$, one gets $x_n \sqsubseteq y_n$. But then, since $x_n \neq y_n$, there are $z_{n-1}, \ldots, z_0$ in $D_{n-1}, \ldots, D_0$ such that $a = x_n z_{n-1} \ldots z_0$ and $a' = y_n z_{n-1} \ldots z_0$ are distinct elements of $D_0$. One then knows that $a \sqsubseteq a'$, and hence it must be the case that either $\perp \sqsubseteq \top$ or $\top \sqsubseteq \perp$. In the first case, one has

$$\begin{aligned} \top = {} & I \cdot \top \sqsubseteq T \cdot \top & = \top \\ \perp = {} & I \cdot \perp \sqsubseteq T \cdot \perp & = \top, \end{aligned}$$

hence, since $\sqsubseteq$ is a lambda-preorder, $I \sqsubseteq T$. Similarly, in the second case, one has $T \sqsubseteq I$, the required contradiction. $\square$

### 3.5.2 Absolute notions

There is a multitude of notions of absolute unorderability that one can consider. Fortunately, we will see that all of them coincide. Recall that we defined a combinatory algebra $\mathbf{A}$ to be absolutely unorderable if for every embedding $\mathbf{A} \hookrightarrow \mathbf{B}$, the algebra $\mathbf{B}$ is unorderable. First, one can adapt this with respect to preorders, lambda-orders etc. Second, one can replace the word "embedding" by "homomorphism". Third, one can restrict attention to certain subcategories, *e.g.* lambda algebras or lambda models.

Instead of cataloging some 30 different notions and showing them all to be equivalent, we start with some simple observations. If $P$ is some property of objects in a category, we say that an object $\mathbf{A}$ ***absolutely satisfies*** $P$ if for all $\mathbf{A} \to \mathbf{B}$, $\mathbf{B}$ satisfies $P$.

First notice that, since lambda algebras are defined by closed equations, their full subcategory is right-closed in the category of combinatory algebras: *i.e.*, if $\mathbf{A} \to \mathbf{B}$ is a homomorphism of combinatory algebras, and $\mathbf{A}$ is a lambda algebra, then so is $\mathbf{B}$. Hence, a lambda algebra $\mathbf{A}$ satisfies some property absolutely *as a lambda algebra* iff it does so *as a combinatory algebra*. The corresponding property is true for Curry algebras.

Next, there are some obvious implications: if $\mathbf{A}$ absolutely satisfies $P$ with respect to homomorphisms, then also with respect to embeddings. We also have the implications that were discussed in Section 3.5.1.

It therefore suffices to show, for each of the categories of combinatory algebras, lambda algebras, and Curry algebras, that the weakest notion that we are considering implies the strongest one. This is done in the following proposition.

**Proposition 3.17.** *For a combinatory algebra* $\mathbf{A}$*, the following are equivalent:*

1. *There is* $\mathbf{A} \to \mathbf{B}$ *for some non-symmetrically preordered combinatory algebra* $\mathbf{B}$.
2. *There is* $\mathbf{A} \hookrightarrow \mathbf{B}$ *for some non-trivially partially ordered combinatory algebra* $\mathbf{B}$.

*For a lambda algebra* $\mathbf{A}$*, the following are equivalent:*

3. *There is* $\mathbf{A} \to \mathbf{B}$ *for some non-symmetrically preordered lambda algebra* $\mathbf{B}$.
4. *There is* $\mathbf{A} \hookrightarrow \mathbf{B}$ *for some non-trivially lambda-ordered lambda model* $\mathbf{B}$.

*For a Curry algebra* $\mathbf{A}$*, the following are equivalent:*

5. *There is* $\mathbf{A} \to \mathbf{B}$ *for some non-symmetrically preordered Curry algebra* $\mathbf{B}$.
6. *There is* $\mathbf{A} \hookrightarrow \mathbf{B}$ *for some non-trivially lambda-ordered extensional algebra* $\mathbf{B}$.

*Proof.* 1. $\Rightarrow$ 2.: Suppose $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B}$ is non-symmetrically preordered. Let $\mathbf{B}' = \mathbf{B}/(\leqslant \cap \geqslant)$, then $\mathbf{B}'$ is non-trivially partially ordered and $\mathbf{A} \to \mathbf{B}'$. Now let $\mathbf{B}'' = \mathbf{B}' \times \mathbf{A}$, which is non-trivially ordered by the componentwise order where $\mathbf{A}$ is discrete. We have $\mathbf{A} \hookrightarrow \mathbf{B}''$.

3. $\Rightarrow$ 4.: Suppose $\mathbf{A} \to \mathbf{B}$ and $\mathbf{B}$ is non-symmetrically preordered. First, construct $\mathbf{A} \hookrightarrow \mathbf{B}''$ as in 2.; then consider $\mathbf{A} \hookrightarrow \mathbf{B}'' \hookrightarrow \mathbf{B}''[X]$ for a countable set $X$. We know that $\mathbf{B}''[X]$ is a lambda model by Proposition 2.23. It has a non-trivial lambda order by Lemma 3.18 below.

5. $\Rightarrow$ 6.: Same as 3.$\Rightarrow$4. $\square$

**Lemma 3.18.** *Suppose* $\leqslant$ *is a non-trivial partial order on a lambda algebra* $\mathbf{B}$*. Then* $\leqslant$ *extends naturally to a lambda-order on* $\mathbf{B}[X]$*, for countable* $X$.

*Proof.* First, consider the case of adjoining a single indeterminate $\mathbf{A} \subseteq \mathbf{A}[x]$. Let $\leqslant$ be a partial order on $\mathbf{A}$, and define on $\mathbf{A}[x]$ the partial order $a \leqslant b$ iff $\lambda^* x.a \leqslant \lambda^* x.b$. Notice that if $a$ and $b$ were in $\mathbf{A}$, then $\lambda^* x.a = \mathbf{K}a$ and $\lambda^* x.b = \mathbf{K}b$, hence $a \leqslant b$ in $\mathbf{A}[x]$ iff $\mathbf{K}a \leqslant \mathbf{K}b$ in $\mathbf{A}$ iff $a \leqslant b$ in $\mathbf{A}$, *i.e.* the order on $\mathbf{A}[x]$ is an extension of the order on $\mathbf{A}$. Now consider $\mathbf{B}[X]$, which can be regarded as a union of an ascending chain of subsets $\mathbf{B} \subseteq \mathbf{B}[x_1] \subseteq \mathbf{B}[x_1, x_2] \subseteq \cdots$. Starting with a partial order on $\mathbf{B}$, one can extend it step by step to all of $\mathbf{B}[X]$. In the limit, we obtain a lambda-order, because if $ax \leqslant bx$ for all $x$, then $a, b \in \mathbf{A} = \mathbf{B}[x_1, \ldots, x_{n-1}]$ for some $n$ and one can take $x = x_n$. But $ax_n = bx_n$ in $\mathbf{A}[x_n]$ iff $\lambda^* x_n.ax_n = \lambda^* x_n.bx_n$ in $\mathbf{A}$, *i.e.* $\mathbf{1}a = \mathbf{1}b$. $\square$

Finally, notice that none of the local notions of unorderability that we have considered implies absolute unorderability: Plotkin's finitely separable algebra [50], although it cannot be non-trivially preordered, can still be embedded in an orderable algebra (for example by Theorem 3.9).

# Chapter 4

# Finite Lambda Models

It has long been known that a model of the untyped lambda calculus, in the traditional sense, can never be finite or even recursive [5]. For instance, no consistent lambda theory equates any two of the countably many Church numerals $\bar{0} = \lambda xy.y$, $\bar{1} = \lambda xy.xy$, $\bar{2} = \lambda xy.x(xy)$, etc.; hence, these terms must have distinct denotations in any non-trivial model. Consequently, model constructions of the lambda calculus typically involve passing to an infinite limit, yielding unwieldy models in which term denotations or equality of terms are not effectively computable.

By contrast, we introduce a notion of finite models for the lambda calculus. These finite models are models of reduction, rather than of conversion. Therefore, as we shall see, they are not subject to the usual limitations on size and complexity. Informally, by a *model of conversion*, we mean a model with a soundness property of the form

$$M \cong N \Rightarrow [\![M]\!] = [\![N]\!],$$

where $\cong$ is *e.g.* $\beta$- or $\beta\eta$-convertibility, and $[\![\ ]\!]$ is the function that carries a lambda term to its interpretation in the model. On the other hand, a *model of reduction* has an underlying partial order and a soundness property of the form

$$M \longrightarrow N \Rightarrow [\![M]\!] \leqslant [\![N]\!],$$

where $\longrightarrow$ is *e.g.* $\beta$- or $\beta\eta$-reduction. Models of reduction have been considered by different authors [23, 30, 49]. We will focus here on a formulation which was given by Plotkin [49] in the spirit of the familiar *syntactical lambda models* [5]. The key observation here is that models of reduction, unlike models of conversion, may be finite, and that they can be easily constructed. In special cases, models of reduction allow a limited form of reasoning about convertibility of terms. This is the case for instance if the underlying partial order is a tree.

We begin by reviewing syntactical and categorical models of reduction in Section 4.1. In Section 4.2, we introduce a reasoning principle for models whose underlying order is a tree. We also give a method for efficiently constructing such models. In Section 4.3, this is further specialized to the case where the underlying order is flat. Examples are given in Section 4.4. Some reflections on completeness properties follow in Section 4.5. In Section 4.6, we investigate the connection between models of reduction and the $D_\infty$-construction.

## 4.1 Models of reduction

### 4.1.1 Syntactical models of reduction

**Definition. (Plotkin [49])** An ***ordered applicative structure*** $\langle P, \cdot \rangle$ is a poset $P$, together with a monotone binary operation $\cdot\colon P \times P \to P$. Let $P^{\mathcal{V}}$ be the set of all ***valuations***, *i.e.* functions from variables to $P$. A

***syntactical model of β-reduction*** $\langle P, \cdot, [\![\ ]\!] \rangle$ is an ordered applicative structure together with an ***interpretation function***

$$[\![\ \cdot\ ]\!]_{\cdot} \colon \Lambda \times P^{\mathcal{V}} \to P$$

with the following properties:

1. $[\![x]\!]_\rho = \rho(x)$
2. $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$
3. $[\![\lambda x.M]\!]_\rho \cdot a \leqslant [\![M]\!]_{\rho(x:=a)}$, for all $a \in P$
4. $\rho|_{\mathrm{FV}(M)} = \rho'|_{\mathrm{FV}(M)} \Rightarrow [\![M]\!]_\rho = [\![M]\!]_{\rho'}$
5. $(\forall a.[\![M]\!]_{\rho(x:=a)} \leqslant [\![N]\!]_{\rho(x:=a)}) \Rightarrow [\![\lambda x.M]\!]_\rho \leqslant [\![\lambda x.N]\!]_\rho$

Moreover, we say $\langle P, \cdot, [\![\ ]\!] \rangle$ is a ***syntactical model of βη-reduction***, if it also satisfies the property

6. $[\![\lambda x.Mx]\!]_\rho \leqslant [\![M]\!]_\rho$, if $x \notin \mathrm{FV}(M)$.

A ***syntactical model of conversion*** is a syntactical model of $\beta$-reduction $\langle X, \cdot, [\![\ ]\!] \rangle$, where $X$ is discretely ordered, *i.e.*, a set. Notice that this notion coincides with the familiar syntactical lambda models as defined *e.g.* in [5].

*Remark.* Properties 1.–3. do not form an inductive definition; rather they state properties of a function $[\![\ ]\!]$ which is given *a priori*. In particular, 3. does not uniquely determine the interpretation of a lambda abstraction $[\![\lambda x.M]\!]_\rho$.

We have seen in Chapter 3 that many models of conversion are equipped with a partial order. This, however, is entirely different from the partial order we consider on a model of reduction. Models of conversion have an *approximation order*, where $a \leqslant b$ is often understood to mean that $a$ is "less defined" or "diverges more often" than $b$. On the other hand, models of reduction have a *reduction order*, where $a \leqslant b$ means $a$ reduces to $b$. More precisely, one has the following soundness theorem:

**Proposition 4.1 (Plotkin [49]).** *The following are properties of syntactical models of $\beta$-reduction:*

1. **Monotonicity.** If $\rho(x) \leqslant \rho'(x)$ for all $x$, then $[\![M]\!]_\rho \leqslant [\![M]\!]_{\rho'}$.
2. **Substitution.** $[\![M[N/x]]\!]_\rho = [\![M]\!]_{\rho(x:=[\![N]\!]_\rho)}$.
3. **Soundness for reduction.** If $M \xrightarrow{\beta} N$, then $[\![M]\!]_\rho \leqslant [\![N]\!]_\rho$. In a syntactical model of $\beta\eta$-reduction: If $M \xrightarrow{\beta\eta} N$, then $[\![M]\!]_\rho \leqslant [\![N]\!]_\rho$. □

Syntactical models of $\beta$-reduction are easily constructed. One may, for example, start with any pointed poset $P$ and monotone function $\cdot \colon P \times P \to P$, and define, somewhat uningeniously, $[\![\lambda x.M]\!]_\rho = \bot$. Among the possible interpretation functions on a given ordered applicative structure, this choice is the minimal one. Much more interesting is the situation in which there exists a maximal such choice. We will explore such a situation in Section 4.2.2.

### 4.1.2 Categorical models of reduction

Let $\mathcal{D}$ be a cartesian-closed category of posets and monotone functions, with the pointwise order on hom-sets.

**Definition.** A ***categorical model of β-reduction*** $\langle P, e, p \rangle$ is given by an object $P \in \mathcal{D}$, together with a pair of morphisms $e \colon P \to P^P$ and $p \colon P^P \to P$, such that

$$\begin{tikzcd}
P^P \arrow[d, "p"'] \arrow[dr, "\mathsf{id}"] \arrow[dr, phantom, "\leqslant" near end, shift right=3ex] & \\
P \arrow[r, "e"'] & P^P.
\end{tikzcd}$$

If moreover $p \circ e \leqslant \mathsf{id}_P$, then $\langle P, e, p \rangle$ is a ***categorical model of βη-reduction***.

Categorical models of reduction have been studied by various authors, *e.g.* by Girard [23] for the case of qualitative domains, or by Jacobs *et al.* [30], where they are called models of expansion. For a detailed discussion of these and other references, see Plotkin [49].

From a categorical model of reduction $\langle P, e, p\rangle$, one can construct a syntactical model of reduction $\langle P, \cdot, [\![\ ]\!]\rangle$ by letting $a \cdot b = e(a)(b)$ and by defining $[\![\ ]\!]$ inductively:

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x), \\ [\![MN]\!]_\rho &= e([\![M]\!]_\rho)([\![N]\!]_\rho), \\ [\![\lambda x.M]\!]_\rho &= p(\lambda a.[\![M]\!]_{\rho(x:=a)}). \end{aligned}$$

**Proposition 4.2.** *If $\langle P, e, p\rangle$ is a categorical model of $\beta$-reduction, then the above construction yields a well-defined syntactical model of $\beta$-reduction $\langle P, \cdot, [\![\ ]\!]\rangle$. Moreover, $\langle P, e, p\rangle$ is a categorical model of $\beta\eta$-reduction, then $\langle P, \cdot, [\![\ ]\!]\rangle$ is a syntactical model of $\beta\eta$-reduction.*

*Proof.* To see that the inductive definition is well-defined, and in particular that the function $\lambda a.[\![M]\!]_{\rho(x:=a)}$ indeed defines an element in $P^P$, it is best to work directly in the category $\mathcal{D}$ and to define an interpretation $[\![M]\!]_{x_1,\dots,x_n}$ of each lambda term $M$ with FV$(M) \subseteq \{x_1, \dots, x_n\}$ as a morphism $P^n \to P$, just as we did for categorical models of conversion:

$$\begin{array}{rcl} [\![x_i]\!]_{x_1,\dots,x_n} &=& P^n \xrightarrow{\pi_i} P \qquad \text{(the $i$th projection)} \\ [\![MN]\!]_{x_1,\dots,x_n} &=& P^n \xrightarrow{\langle [\![M]\!]_{x_1,\dots,x_n}, [\![N]\!]_{x_1,\dots,x_n}\rangle} P \times P \xrightarrow{p_\star} P \\ [\![\lambda x_{n+1}.M]\!]_{x_1,\dots,x_n} &=& P^n \xrightarrow{([\![M]\!]_{x_1,\dots,x_{n+1}})^\star} P^P \xrightarrow{e} P. \end{array}$$

It is easily seen that the two definitions coincide in the sense that

$$[\![M]\!]_\rho = 1 \xrightarrow{\langle \rho(x_1),\dots,\rho(x_n)\rangle} P^n \xrightarrow{[\![M]\!]_{x_1,\dots,x_n}} P.$$

The verification that this is a syntactical model of $\beta$-, respectively, $\beta\eta$-reduction is now routine. ☐

### 4.1.3 Models of $\beta\eta$-reduction: Order-extensionality

We have seen in Chapter 2 that an extensional model of $\beta$-conversion is always a model of $\beta\eta$-conversion. The property that corresponds to extensionality for models of reduction is order-extensionality: An ordered applicative structure $\langle P, \cdot\rangle$ is called ***order-extensional*** if

$$\frac{\forall x \in P.\ ax \leqslant bx}{a \leqslant b}.$$

**Lemma 4.3.** *If a syntactical model of $\beta$-reduction $\langle P, \cdot, [\![\ ]\!]\rangle$ is order-extensional, then it is a model of $\beta\eta$-reduction.*

*Proof.* Suppose $x \in$ FV$(M)$. Then for all $a \in P$, $[\![\lambda x.Mx]\!]_\rho \cdot a \leqslant [\![Mx]\!]_{\rho(x:=a)} = [\![M]\!]_{\rho(x:=a)} \cdot [\![x]\!]_{\rho(x:=a)} = [\![M]\!]_\rho \cdot a$, hence $[\![\lambda x.Mx]\!]_\rho \leqslant [\![M]\!]_\rho$. ☐

## 4.2 Tree models

### 4.2.1 Recapturing convertibility

The soundness property for models of reduction does not in general yield useful information about convertibility, since interconvertible terms $M \cong N$ may have different denotations. However, if the reduction under consideration is Church-Rosser, then $M \cong N$ implies that there is a term $Q$ with $M \longrightarrow Q$ and $N \longrightarrow Q$.

Therefore, the denotations $[\![M]\!]_\rho$ and $[\![N]\!]_\rho$ must be compatible. Recall that $a$ and $b$ are ***compatible***, in symbols $a \frown b$, if there exists $c$ with $a \leqslant c$ and $b \leqslant c$. In a model of reduction, one has the following restricted form of soundness for convertibility:

$$M \cong N \Rightarrow [\![M]\!]_\rho \frown [\![N]\!]_\rho. \tag{4.1}$$

The latter property is especially useful if the underlying poset $P$ has many pairs of incompatible elements. Therefore, we will pay special attention to the cases where $P$ is a tree or a flat partial order.

**Definition.** A pointed poset $P$ is called a ***tree*** if for all $a, b \in P$, $a \frown b$ implies $a \leqslant b$ or $a \geqslant b$. Equivalently, for each $x \in P$, the downdeal $\downarrow x$ is linearly ordered. A tree $P$ is said to be ***bounded*** if there is a number $n \in \mathbb{N}$ such that each $\downarrow x$ has at most $n$ elements. The smallest such $n$ is called the ***height*** of $P$.

A model of reduction is called a ***tree model*** if the underlying poset is a tree.

### 4.2.2 A method for constructing models

In general, there may be many different ways of defining an interpretation function $[\![\ ]\!]$ that makes a given ordered applicative structure $\langle P, \cdot\rangle$ into a syntactical model of reduction. Even if one restricts attention to those cases where $[\![\ ]\!]$ is defined inductively from a categorical model $\langle P, e, p\rangle$, with $e(a)(b) = a \cdot b$, there is a choice involved in determining the morphism $p : P^P \to P$. In general, the greater $p$ is chosen with respect to the pointwise order, the greater the resulting interpretation $[\![\ ]\!]$ will be, and the better one will be able to make use of the soundness property for convertibility 4.1.

The best possible situation arises if we can find a right adjoint $p$ of $e$, because if $p$ is such a right adjoint, then it is maximal with the property $e \circ p \leqslant \mathsf{id}$. It is well-known that if $P$ is a complete lattice, then $e : P \to Q$ has a right adjoint if and only if $e$ preserves suprema. In this case, one can define $p(y) = \bigvee\{x \in P \mid e(x) \leqslant y\}$. But following the remarks in Section 4.2.1, we are interested in posets $P$ that have incompatible pairs of elements, and which can therefore not be complete lattices. In the case of bounded trees, the existence of a right adjoint is characterized by a property which we call strong extensionality:

**Definition.** Let $P$ be a bounded tree. We say that an ordered applicative structure $\langle P, \cdot\rangle$ is ***strongly extensional*** if for all $a, b \in P$,

$$\frac{\forall x \in P.\ ax \frown bx}{a \frown b}.$$

**Proposition 4.4.** *Let $\langle P, \cdot\rangle$ be an ordered applicative structure, where $P$ is a bounded tree. Let $e : P \to P^P$ be the map defined by $e(a)(b) = a \cdot b$. Then $e$ has a right adjoint in the category of posets if and only if $\langle P, \cdot\rangle$ is strongly extensional.*

*Proof.* $\Rightarrow$: Suppose $e$ has a right adjoint $p : P^P \to P$. Let $a, b \in P$ such that $ax \frown bx$ for all $x$. Since $P$ is a tree, one has $ax \leqslant bx$ or $ax \geqslant bx$ for every $x$. Define a monotone map $f : P \to P$ by $f(x) = \max(ax, bx)$. Since $e(a)(x) = ax \leqslant f(x)$ for all $x$, one has $e(a) \leqslant f$ and hence $a \leqslant p(f)$, and similarly for $b$. Hence $a \frown b$, and $\langle P, \cdot\rangle$ is strongly extensional.

$\Leftarrow$: Suppose $\langle P, \cdot\rangle$ is strongly extensional. For any $f \in P^P$, consider the subset $P_f = \{x \in P \mid e(x) \leqslant f\} \subseteq P$. Notice that for any $a, b \in P_f$, $e(a) \frown e(b)$, hence $ax \frown bx$ for all $x$, hence $a \frown b$ by strong extensionality. Since $P$ is a tree, either $a \leqslant b$ or $a \geqslant b$. Therefore $P_f$ is linearly ordered. Since $P$ is bounded, the set $P_f$ is finite, and hence it has a maximal element $p(f)$. Clearly, the function $p$ thus defined is monotone, and $x \leqslant p(f)$ iff $x \in P_f$ iff $e(x) \leqslant f$. Therefore $e \dashv p$. □

**Corollary 4.5.** *The last proposition yields a practical method for constructing a tree model of reduction: Begin with a tree $P$ and a monotone binary operation $\cdot : P \times P \to P$, such that $\langle P, \cdot\rangle$ is strongly extensional. Define $[\![\ ]\!]$ inductively as follows:*

1. $[\![x]\!]_\rho = \rho(x)$

2. $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$

3. *$[\![\lambda x.M]\!]_\rho$ is the maximal $b \in P$ such that $b \cdot a \leqslant [\![M]\!]_{\rho(x:=a)}$ for all $a \in P$.*

*Then $\langle P, \cdot, [\![\ ]\!]\rangle$ is a well-defined model of $\beta$-reduction.*

*Proof.* Proposition 4.4, together with Proposition 4.2, ensures that this is well-defined, in particular, that a maximal $b$ exists in 3. □

The following lemma is sometimes useful for reasoning about such a model:

**Lemma 4.6.** *If $[\![\ ]\!]$ is defined as in Corollary 4.5, then for all $n \geqslant 1$, the denotation of an $n$-fold lambda abstraction $[\![\lambda x_1 \ldots x_n.M]\!]_\rho$ is the maximal $b \in P$ such that for all $a_1 \ldots a_n \in X$,*

$$b \cdot a_1 \cdots a_n \leqslant [\![M]\!]_{\rho(x_1:=a_1)\ldots(x_n:=a_n)}.$$

*Proof.* By induction on $n$. □

If $\langle P, \cdot\rangle$ is order-extensional, then the construction in Corollary 4.5 yields a model of $\beta\eta$-reduction by Lemma 4.3. We end this section with a lemma that relates order-extensionality to strong extensionality for tree models:

**Lemma 4.7.** *If $P$ is a tree, and if $\langle P, \cdot\rangle$ is strongly extensional and extensional, then it is also order-extensional.*

*Proof.* Suppose for all $x$, $ax \leqslant bx$, hence $ax \frown bx$, hence $a \frown b$ by strong extensionality. Since $P$ is a tree, either $a \leqslant b$ or $a \geqslant b$. In the first case, we are done; in the second case, $ax \geqslant bx$, and hence $ax = bx$, for all $x$, which implies $a = b$ by extensionality. □

## 4.3 Partial models

As the examples in Section 4.4 will show, it often suffices to consider tree models whose underlying poset $P$ is flat, *i.e.* $P = X_\bot$ for a discrete set $X$. If one also assumes that the application operation $\cdot \colon P \times P \to P$ is strict in each argument, then one can think of $\bot$ as the *undefined* element, and of $\cdot$ and $[\![\ ]\!]$ as *partial* functions. Since it is sometimes convenient to think in terms of these partial operations, we restate the definition of a model of reduction in this special case. The venturi-tube $\succeq$ denotes ***directed equality***: $A \succeq B$ means that if $A$ is defined, then so is $B$, and they are equal.

**Definition.** A ***partial applicative structure*** $\langle X, \cdot\rangle$ is a set $X$ with a partial binary operation $\cdot \colon X \times X \rightharpoonup X$. Let $\mathrm{Val}(X)$ be the set of partial valuations $\mathcal{V} \rightharpoonup X$. A ***partial syntactical lambda model*** $\langle X, \cdot, [\![\ ]\!]\rangle$, or ***partial model*** for short, is given by a partial applicative structure together with a partial map

$$[\![\cdot]\!]_\cdot \colon \Lambda_X \times \mathrm{Val}(X) \rightharpoonup X,$$

such that

1. $[\![x]\!]_\rho = \rho(x)$
2. $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$
3. $[\![\lambda x.M]\!]_\rho \cdot a \succeq [\![M]\!]_{\rho(x:=a)}$, for all $a \in X$
4. $\rho|_{\mathrm{FV}(M)} = \rho'|_{\mathrm{FV}(M)} \Rightarrow [\![M]\!]_\rho = [\![M]\!]_{\rho'}$
5. $(\forall a. [\![M]\!]_{\rho(x:=a)} \succeq [\![N]\!]_{\rho(x:=a)}) \Rightarrow [\![\lambda x.M]\!]_\rho \succeq [\![\lambda x.N]\!]_\rho$

Moreover, if

6. $[\![\lambda x.Mx]\!]_\rho \succeq [\![M]\!]_\rho$, if $x \notin \mathrm{FV}(M)$.

then $\langle X, \cdot, [\![\ ]\!]\rangle$ is a ***partial $\beta\eta$-model***.

Here, equality is understood to be ***Kleene equality***, meaning $A = B$ if and only if $A$ and $B$ are either both undefined or both defined and equal. Notice that the directed equality $\succeq$ on $X$ is just the partial order on the flat poset $X_\perp$. Thus, the axioms 1–5 and 6 correspond exactly to the axioms for a syntactical model of $\beta$-, respectively, $\beta\eta$-reduction.

In a partial model, the denotation of some terms may be undefined. The idea of using partiality in models for the lambda calculus is not new. In fact, Kleene's "first model", which consists of Gödel numbers of partial recursive functions and their application, is partial. The models we consider here are even "more" partial; we do not even assume that the interpretations of basic combinators such as $S$ and $K$ are defined. The following soundness properties ensure that the class of terms whose denotation is defined is closed under reduction, and that interconvertible terms have the same denotation if they are both defined.

**Proposition 4.8.** *The following are properties of partial models:*

1. **Soundness for reduction.** If $M \xrightarrow{\beta} N$, then $[\![M]\!]_\rho \succeq [\![N]\!]_\rho$.
2. **Soundness for convertibility.** If $M =_\beta N$, and if $[\![M]\!]_\rho$ and $[\![N]\!]_\rho$ are both defined, then $[\![M]\!]_\rho = [\![N]\!]_\rho$.
3. In a partial $\beta\eta$-model, the respective properties hold for $\xrightarrow{\beta\eta}$ and $=_{\beta\eta}$.

*Proof.* Soundness for reduction follows from Proposition 4.1. Soundness for convertibility follows from the Church-Rosser property. □

Partial applicative structures are particularly easy to manipulate in practice, since they are just given by a set $X$ and a "multiplication table" such as the one in Table 4.1. It is easy to read properties such as strong extensionality off the table: A partial applicative structure is strongly extensional if no two rows of the multiplication table are compatible, and it is order-extensional if no row is subsumed by another. In particular, if the table is everywhere defined, *i.e.* if $\langle X, \cdot\rangle$ is a total applicative structure, then both strong extensionality and order-extensionality coincide with (ordinary) extensionality.

## 4.4 Examples

### 4.4.1 A class of finite models to distinguish the terms $\Omega_n$

Let $x$ be a variable and define $x^1 = x$ and $x^{n+1} = x^n x$ for $n \geqslant 1$. Let $\omega_n = \lambda x.x^n$ and $\Omega_n = \omega_n\omega_n$. None of these terms has a normal form, *e.g.* $\Omega_2 = (\lambda x.xx)(\lambda x.xx)$ reduces only to itself. The terms $\Omega_n$ are unsolvable; therefore, their interpretations coincide with $\perp$ in the $D_\infty$-model [29, 64]. We will now give a class of finite partial models that distinguishes these terms.

Fix an integer $p \geqslant 1$ and let $X = \mathbb{Z}_p = \{1, 2, \dots, p\}$. Addition and subtraction in $X$ are modulo $p$; let $=_p$ denote equality modulo $p$. Define $\cdot\colon X \to X$ by

$$n \cdot m =_p \begin{cases} n+1 & \text{if } m =_p 1 \\ m+1 & \text{if } m \neq_p 1. \end{cases}$$

A "multiplication table" for this operation is shown in Table 4.1. Clearly, $\langle X, \cdot\rangle$ is a strongly extensional applicative structure. Define $[\![\ ]\!]$ as in Corollary 4.5 to get a partial model. For $n \geqslant 2$, we calculate $1^n =_p n$ and $m^n =_p m+1$ for $m \neq 1$. Hence, for all $x \in X$ and $n \geqslant 2$,

$$\begin{array}{rrcl} & x^n & =_p & (n-1)\cdot x \\ \Rightarrow & [\![\omega_n]\!] = [\![\lambda x.x^n]\!] & =_p & n-1 \\ \Rightarrow & [\![\Omega_n]\!] = [\![\omega_n\omega_n]\!] & =_p & (n-1)\cdot(n-1) =_p n. \end{array}$$

Hence, $[\![\Omega_n]\!]$ is always defined for $n \geqslant 2$, and we have $[\![\Omega_n]\!] = [\![\Omega_m]\!]$ iff $n =_p m$.

Table 4.1: Multiplication table for a partial model

| $\cdot$ | 1 | 2 | 3 | $\cdots$ | $p-1$ | $p$ |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | $\cdots$ | $p$ | 1 |
| 2 | 3 | 3 | 4 | $\cdots$ | $p$ | 1 |
| 3 | 4 | 3 | 4 | $\cdots$ | $p$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $p-1$ | $p$ | 3 | 4 | $\cdots$ | $p$ | 1 |
| $p$ | 1 | 3 | 4 | $\cdots$ | $p$ | 1 |

### 4.4.2 A non-trivial 3-element model

In this section, we provide the proof of Lemma 3.1 from Chapter 3. At the heart of the proof is a 3-element partial model which distinguishes two appropriately chosen unsolvable terms $Auuut$ and $Auutt$.

**Lemma.** *There is a closed term $A$ of the untyped lambda calculus with $Auuut =_\beta Auttt$, but $Auuut \neq_{\beta\eta} Auutt \neq_{\beta\eta} Auttt$ for variables $u \neq t$.*

*Proof.* Define terms

$$\begin{array}{rcl} h & = & \lambda zyx.zzy(zzy(zzyx)) \\ f & = & hh \\ A & = & \lambda uvwt.\lambda x.fu(fv(fw(ftx))). \end{array}$$

Then for all $x$, $y$:

$$fyx \xrightarrow{\beta} fy(fy(fyx)),$$

hence for all $u$, $t$:

$$\begin{array}{rcl} \lambda x.fu(ftx) & \xrightarrow{\beta} & \lambda x.fu(fu(fu(ftx))) = Auuut \\ \lambda x.fu(ftx) & \xrightarrow{\beta} & \lambda x.fu(ft(ft(ftx))) = Auttt. \end{array}$$

To see that $Auuut \neq_{\beta\eta} Auutt$ for variables $u$ and $t$, we will construct a partial model. Let $X = \{\mathrm{k}, 0, 1\}$, and let $\cdot$ be defined by the following "multiplication table":

| $\cdot$ | k | 0 | 1 |
|---|---|---|---|
| k | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |

Then $\langle X, \cdot\rangle$ is a strongly extensional applicative structure. Define $[\![\ ]\!]$ inductively as in Corollary 4.5. Although $\langle X, \cdot\rangle$ is total, $[\![\ ]\!]$ will be partial.

Consider the function $\psi(c, b, a) := [\![zzy(zzy(zzyx))]\!]_{\rho(z:=c)(y:=b)(x:=a)} = ccb(ccb(ccba))$. Table 4.2 shows the values of this function, and one observes that $\psi(c, b, a) = \mathrm{k} \cdot c \cdot b \cdot a$ for all $c, b, a \in X$. Hence by Corollary 4.5, $[\![h]\!] = [\![\lambda zyx.zzy(zzy(zzyx))]\!]$ is defined and equal to k, and consequently $[\![f]\!] = [\![hh]\!] = \mathrm{kk} = 0$. If $\rho(u) = \rho(x) = 0$ and $\rho(t) = 1$, then

$$\begin{array}{rcl} [\![fu(fu(fu(ftx)))]\!]_\rho & = & 1 \\ [\![fu(fu(ft(ftx)))]\!]_\rho & = & 0. \end{array}$$

By soundness, $fu(fu(fu(ftx))) \neq_{\beta\eta} fu(fu(ft(ftx))) \Rightarrow Auuut \neq_{\beta\eta} Auutt.$ $\square$

Table 4.2: Values for $\psi(c, b, a)$ and $\mathrm{k} \cdot c \cdot b \cdot a$

| $c$ | $b$ | $a$ | $\psi(c, b, a)$ | $\mathrm{k} \cdot c \cdot b \cdot a$ |
|---|---|---|---|---|
| k or 0 or 1 | k | k | 0 | 0 |
| | k | 0 | 0 | 0 |
| | k | 1 | 1 | 1 |
| | 0 | k | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| | 0 | 1 | 1 | 1 |
| | 1 | k | 0 | 0 |
| | 1 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 0 |

## 4.5 Completeness

Given a syntactical model of $\beta$- or $\beta\eta$-reduction $\langle P, \cdot, [\![\ ]\!]\rangle$, one can define its ***lift*** $\langle P_\perp, \bullet, [\![\ ]\!]'\rangle$ as follows:

$$a \bullet b \quad = \quad \begin{cases} a \cdot b & \text{if } a, b \neq \perp \\ \perp & \text{else,} \end{cases}$$

$$[\![M]\!]'_\rho \quad = \quad \begin{cases} [\![M]\!]_\rho & \text{if } \rho(x) \neq \perp \text{ for all } x \in \mathrm{FV}(M) \\ \perp & \text{else.} \end{cases}$$

It is easily checked that this is again a model of $\beta$-, respectively, $\beta\eta$-reduction. As a trivial consequence, one has the following completeness theorem for partial models:

**Proposition 4.9. Completeness:** *If $M \neq_\beta N$, then there is a partial model and $\rho$ for which $[\![M]\!]_\rho$, $[\![N]\!]_\rho$ are defined and $[\![M]\!]_\rho \neq [\![N]\!]_\rho$. If $M \neq_{\beta\eta} N$, then the model can be chosen to be strongly extensional.* □

*Proof.* Take a model of conversion such that $[\![M]\!]_\rho \neq [\![N]\!]_\rho$ for some $\rho$, *e.g.* a term model. Then its lift is a partial model with $[\![M]\!]'_\rho \not\subset [\![N]\!]'_\rho$. □

Of course much more interesting questions can be asked, *e.g.* how close one can come to a *finite completeness theorem* for models of reduction? In other words: can every inequality $M \neq_\beta N$ be demonstrated in a finite model of reduction? The answer to this question must be no, since such a finite completeness theorem would yield a decision procedure for convertibility of lambda terms, which is known to be an undecidable problem. It is an open problem to identify subclasses of terms for which a finite completeness property holds, or to describe the class of equations that hold in all finite models of reduction, tree models, partial models etc.

## 4.6 Relating models of reduction to $D_\infty$-models

Consider a finite categorical model of reduction $\langle P, e, p\rangle$, such that $e \circ p \leqslant \mathsf{id}_{P^P}$ and $p \circ e = \mathsf{id}_P$. Since $P$ is finite, it is a dcpo and $e$ and $p$ form a Scott-continuous embedding-projection pair. Therefore, one can take $P$, $e$ and $p$ as the basis for carrying out the $D_\infty$-construction in the category **CPO**, as outlined in Section 1.2.6.

Let $D_0 = P$ and $D_{n+1} = D_n^{D_n}$. Let $e_0 = e : D_0 \to D_1$ and $p_0 = p : D_1 \to D_0$. From this, construct the other embedding-projection pairs and take the bilimit $D_\infty$ as usual. Let $\iota_n : D_n \to D_\infty$ and $\pi_n : D_\infty \to D_n$

be the limiting morphisms. For each $n \geqslant 0$ one has

$$\begin{tikzcd} D_n \arrow[r, "e_n"] \arrow[d, "\iota_n"'] & D_n^{D_n} \arrow[d, "\iota_n^{\pi_n}"] \\ D_\infty \arrow[r, "e_\infty"'] & D_\infty^{D_\infty} \end{tikzcd} \quad \text{and} \quad \begin{tikzcd} D_n & D_n^{D_n} \arrow[l, "p_n"'] \\ D_\infty \arrow[u, "\pi_n"] & D_\infty^{D_\infty} \arrow[l, "p_\infty"] \arrow[u, "\pi_n^{\iota_n}"'] \end{tikzcd}.$$

Note that each $\langle D_n, e_n, p_n\rangle$ and $\langle D_\infty, e_\infty, p_\infty\rangle$ is a categorical model of reduction. Let $[\![\ ]\!]^n$ and $[\![\ ]\!]^\infty$ be the respective interpretation functions. How are they related? For a valuation of variables $\rho : \mathcal{V} \to D_\infty$, denote by $\rho_n$ the valuation $\pi_n \circ \rho : \mathcal{V} \to D_n$. One may expect that $[\![M]\!]^n_{\rho_n} = \pi_n[\![M]\!]^\infty_\rho$. However, this is in general *not* the case. The following proposition relates $[\![\ ]\!]^n$ and $[\![\ ]\!]^\infty$:

**Proposition 4.10.** *For all lambda terms $M$,*

$$[\![M]\!]^\infty_\rho = \bigvee^{\uparrow}_{n\geqslant 0} \iota_n[\![M]\!]^n_{\rho_n}.$$

*Proof.* First recall from Proposition 1.7 that $\mathrm{id}_{D_\infty} = \bigvee^{\uparrow}_n \iota_n\circ\pi_n$. Also note that $\pi_n\circ p_\infty\circ\iota_n^{\pi_n} = p_n\circ\pi_n^{\iota_n}\circ\iota_n^{\pi_n} = p_n$. The proposition is proved by induction on $M$. There are three cases:

**Case 1:** $[\![x]\!]^\infty_\rho = \rho(x) = \bigvee^{\uparrow}_n \iota_n \circ \pi_n \circ \rho(x) = \bigvee^{\uparrow}_n \iota_n \circ \rho_n(x) = \bigvee^{\uparrow}_n \iota_n[\![x]\!]^n_{\rho_n}$.

**Case 2:**
$$\begin{aligned} [\![MN]\!]^\infty_\rho &= e_\infty([\![M]\!]^\infty_\rho)([\![N]\!]^\infty_\rho) \overset{\text{(IH)}}{=} e_\infty(\textstyle\bigvee^{\uparrow}_n \iota_n[\![M]\!]^n_{\rho_n})(\bigvee^{\uparrow}_n \iota_n[\![N]\!]^n_{\rho_n}) \\ &= \textstyle\bigvee^{\uparrow}_n e_\infty(\iota_n[\![M]\!]^n_{\rho_n})(\iota_n[\![N]\!]^n_{\rho_n}) = \bigvee^{\uparrow}_n \iota_n^{\pi_n}(e_n[\![M]\!]^n_{\rho_n})(\iota_n[\![N]\!]^n_{\rho_n}) \\ &= \textstyle\bigvee^{\uparrow}_n(\iota_n \circ (e_n[\![M]\!]^n_{\rho_n}) \circ \pi_n)(\iota_n[\![N]\!]^n_{\rho_n}) = \bigvee^{\uparrow}_n \iota_n(e_n([\![M]\!]^n_{\rho_n})([\![N]\!]^n_{\rho_n})) \\ &= \textstyle\bigvee^{\uparrow}_n \iota_n[\![MN]\!]^n_{\rho_n}. \end{aligned}$$

**Case 3:**
$$\begin{aligned} [\![\lambda x.M]\!]^\infty_\rho &= p_\infty(\lambda a \in D_\infty.[\![M]\!]^\infty_{\rho(x:=a)}) \overset{\text{(IH)}}{=} p_\infty(\lambda a.\textstyle\bigvee^{\uparrow}_n \iota_n[\![M]\!]^n_{\rho_n(x:=\pi_n a)}) \\ &= \textstyle\bigvee^{\uparrow}_n p_\infty(\lambda a.\iota_n[\![M]\!]^n_{\rho_n(x:=\pi_n a)}) = \bigvee^{\uparrow}_n p_\infty \circ \iota_n^{\pi_n}(\lambda b \in D_n.[\![M]\!]^n_{\rho_n(x:=b)}) \\ &= \textstyle\bigvee^{\uparrow}_n \iota_n \circ \pi_n \circ p_\infty \circ \iota_n^{\pi_n}(\lambda b \in D_n.[\![M]\!]^n_{\rho_n(x:=b)}) \\ &= \textstyle\bigvee^{\uparrow}_n \iota_n \circ p_n(\lambda b \in D_n.[\![M]\!]^n_{\rho_n(x:=b)}) = \bigvee^{\uparrow}_n \iota_n[\![\lambda x.M]\!]^n_{\rho_n}. \end{aligned}$$
□

In particular, it follows that $\iota_n[\![M]\!]^n_{\rho_n} \leqslant [\![M]\!]^\infty_\rho$ for every $M$, and by applying $\pi_n$ to both sides, it also follows that $[\![M]\!]^n_{\rho_n} \leqslant \pi_n[\![M]\!]^\infty_\rho$. To see that equality does not in general hold, notice that $D_\infty$, by construction, is a model of conversion. Hence for all $M =_\beta N$, one has $\pi_n[\![M]\!]^\infty_\rho = \pi_n[\![N]\!]^\infty_\rho$. On the other hand, $D_n$ is finite and hence a proper model of reduction. Therefore, it is possible to find $M, N$ with $M =_\beta N$ and $[\![M]\!]^n_{\rho_n} \neq [\![N]\!]^n_{\rho_n}$.

**Corollary 4.11.** *If $M$ and $N$ are lambda terms such that, for some $n$, $[\![M]\!]^n_{\rho_n} \not\supset\!\!\!\subset [\![N]\!]^n_{\rho_n}$, then $[\![M]\!]^\infty_\rho \not\supset\!\!\!\subset [\![N]\!]^\infty_\rho$. The converse holds if $D_\infty$ is bounded complete (this is the case, for instance, if $P$ is a tree).*

*Proof.* Suppose $[\![M]\!]^\infty_\rho \supset\!\!\!\subset [\![N]\!]^\infty_\rho$. Let $c \in D_\infty$ such that $[\![M]\!]^\infty_\rho, [\![N]\!]^\infty_\rho \leqslant c$. Then $[\![M]\!]^n_{\rho_n} \leqslant \pi_n[\![M]\!]^\infty_\rho \leqslant \pi_n c$, and similarly for $[\![N]\!]^n_{\rho_n}$. For the converse, assume $D_\infty$ is bounded complete. Assume that for all $n \geqslant 0$, $[\![M]\!]^n_{\rho_n} \supset\!\!\!\subset [\![N]\!]^n_{\rho_n}$. Then $\iota_n[\![M]\!]^n_{\rho_n} \supset\!\!\!\subset \iota_n[\![N]\!]^n_{\rho_n}$ for all $n$. Let $c_n = \iota_n[\![M]\!]^n_{\rho_n} \vee \iota_n[\![N]\!]^n_{\rho_n}$ in $D_\infty$. Then $(c_n)_{n\geqslant 0}$ is an increasing sequence and $[\![M]\!]^\infty_\rho = \bigvee^{\uparrow}_n \iota_n[\![M]\!]^n_{\rho_n} \leqslant \bigvee^{\uparrow}_n c_n$, and similarly $[\![N]\!]^\infty_\rho \leqslant \bigvee^{\uparrow}_n c_n$, hence $[\![M]\!]^\infty_\rho \supset\!\!\!\subset [\![N]\!]^\infty_\rho$. □

# Chapter 5

# Henkin Representations, Polymorphism, and Empty Types

The polymorphic lambda calculus was independently discovered by Girard [22] and Reynolds [51]. It has been extensively studied as a prototypical programming language because of its great expressive power and economy of syntax. The basic idea is to augment the simply-typed lambda calculus with type variables $\alpha, \beta, \ldots$ and with explicit universal quantification over types. This allows the formulation of algorithms that uniformly handle data of more than one type. Type instantiation and type abstraction is made explicit on terms: If $t$ is a term of type $\forall\alpha.\tau$, then $t\sigma$ is a term of type $\tau[\sigma/\alpha]$, for all types $\sigma$. Conversely, if $s$ is a term of type $\tau$, then $\Lambda\alpha.s$ is a term of type $\forall\alpha.\tau$. Now consider for instance the type

$$\mathit{Polybool} = \forall\alpha.\alpha \to \alpha \to \alpha.$$

A term $t$ of this type yields a term $t\sigma$ of type $\sigma \to \sigma \to \sigma$, for every type $\sigma$. Moreover, following Strachey's concept of *parametric polymorphism* [62], one expects the behavior of $t\sigma$ to vary uniformly with the choice of $\sigma$. In the polymorphic lambda calculus, there are only two such uniform functions of type *Polybool*, *i.e.* there are exactly two closed terms of type *Polybool*, corresponding to the first and second projections:

$$p_1 = \Lambda\alpha.\lambda x{:}\alpha.\lambda y{:}\alpha.x \quad \text{and} \quad p_2 = \Lambda\alpha.\lambda x{:}\alpha.\lambda y{:}\alpha.y.$$

Several notions of models for the polymorphic lambda calculus have been proposed in the 1980's. These models follow one of two basic designs:

1. Environment-style models, which have been considered by Bruce and Meyer [10], extend the familiar Henkin models of the simply-typed lambda calculus. These models are *non-strict*, in the sense that a function type $\sigma \to \tau$ is interpreted as a subset of the set of functions from $\sigma$ to $\tau$, and similarly a universal type $\forall\alpha.\tau$ is interpreted as a subset of an infinite product $\prod_\sigma \tau[\sigma/\alpha]$.

2. Categorical models, introduced by Seely [56], are based on general principles for the interpretation of quantifiers in categorical hyperdoctrines. Seely's *PL*-categories are a canonical extension of the ccc interpretation of the simply-typed lambda calculus. These interpretations are *strict*, in the sense that both function types and universal types are interpreted directly by their categorical counterparts.

These two classes of models do not readily mesh, because it is known that strict interpretations collide with the classical foundations: Reynolds showed that there are no set-theoretic strict models of the polymorphic lambda calculus [52].

The aim of this chapter is to reconcile the categorical and the set-theoretical approaches by giving a categorical treatment of non-strict models. This generalizes both Seely's models and the models of Bruce and Meyer. The central concept is that of a *Henkin representation*: a functor $H$ between ccc's is a Henkin representation if it preserves finite products and if for all objects $A, B$, the canonical morphism $H(B^A) \rightarrowtail H(B)^{H(A)}$ is monic.

In Section 5.1, we prove three Henkin representation theorems characterizing those ccc's which can be Henkin-represented, respectively, in the category of non-empty sets $\mathscr{S}^+$, the category of sets $\mathscr{S}$, and a category $\mathscr{S}^P$ of presheaves over some poset $P$. After reviewing the simply-typed lambda calculus in Section 5.2, we show in Section 5.3 that the three Henkin representation theorems correspond naturally to completeness theorems for three different classes of non-strict models: Friedman's set-theoretic models with non-empty types [21], set-theoretic models with possibly empty types, as investigated by Meyer *et al.* [39], and Mitchell and Moggi's Kripke lambda models [42], respectively. Sections 5.4 through 5.6 are devoted to Henkin representations of *PL*-categories and their relationship to completeness theorems for the polymorphic lambda calculus.

# 5.1 Henkin representations of cartesian-closed categories

## 5.1.1 Henkin representations

**Definition.** Let $\mathbf{C}$ and $\mathbf{D}$ be cartesian-closed categories. A functor $H : \mathbf{C} \to \mathbf{D}$ is called a ***Henkin representation*** if it preserves terminator and binary products, and if for all objects $A, B \in \mathbf{C}$, the canonical arrow $(H\varepsilon_{A,B})^\star : H(B^A) \rightarrowtail HB^{HA}$ is monic.

Recall that a ccc-representation $F : \mathbf{C} \to \mathbf{D}$ is a functor that preserves all ccc structure, and in particular $F(B^A) = FB^{FA}$ and $(F\varepsilon_{A,B})^\star = F({\varepsilon_{A,B}}^\star) = \mathsf{id}$. Thus, every ccc-representation is a Henkin representation, but not *vice versa*. Henkin representations arise naturally as the forgetful functors of various concrete ccc's into $\mathscr{S}$. Even though Henkin representations do not in general preserve exponentials, they are 'compatible' with ccc structure in an essential way: their kernels are ccc-congruences. This is why they correspond to useful notions of 'model' for typed lambda calculi.

**Definition 5.1.** A ***ccc-congruence*** $\sim$ on a ccc $\mathbf{C}$ is given by an equivalence relation $\sim_{A,B}$ on each hom-set $(A, B)$, such that the following hold:

$$1.\ \frac{f \sim_{A,B} f' \qquad g \sim_{B,C} g'}{g \circ f \sim_{A,C} g' \circ f'} \qquad 2.\ \frac{f \sim_{A,B} f' \qquad g \sim_{A,C} g'}{\langle f, g\rangle \sim_{A,B\times C} \langle f', g'\rangle} \qquad 3.\ \frac{f \sim_{A\times B,C} f'}{f^\star \sim_{A,C^B} f'^\star}$$

The ***kernel*** of a functor $F : \mathbf{C} \to \mathbf{D}$ is defined by $f \sim_{A,B} f'$ iff $Ff = Ff'$, for all $f, f' : A \to B$. Clearly, the kernel of a ccc-representation is a ccc-congruence. The same is true for Henkin representations:

**Lemma 5.2.** *The kernel of a Henkin representation $H : \mathbf{C} \to \mathbf{D}$ is a ccc-congruence.*

*Proof.* 1. and 2. are obvious, since $H$ preserves binary products. For 3., suppose $f \sim_{A\times B,C} f'$, *i.e.* $Hf = Hf'$. One has

$$\begin{tikzcd} C^B \times B \arrow[r, "\varepsilon"] & C \\ A \times B \arrow[u, "f^\star \times \mathsf{id}_B"] \arrow[ur, "f"'] \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} H(C^B) \times HB \arrow[r, "H\varepsilon"] & HC \\ HA \times HB \arrow[u, "H(f^\star) \times \mathsf{id}_{HB}"] \arrow[ur, "Hf"'] \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} H(C^B) \arrow[r, tail, "(H\varepsilon)^\star"] & HC^{HB}, \\ HA \arrow[u, "H(f^\star)"] \arrow[ur, "(Hf)^\star"'] \end{tikzcd}$$

and similarly for $f'$. Since $(Hf)^\star = (Hf')^\star$, and since $(H\varepsilon)^\star$ is monic, one gets $H(f^\star) = H(f'^\star)$. ☐

*Remark 5.3.* Henkin representations do not form a category, since they do not in general compose. If $H_1$ and $H_2$ are Henkin representations, then the composition $H_2 \circ H_1$ will be a Henkin representation if $H_2$ preserves monics or if $H_1$ is a ccc-representation.

Henkin representations can also be described in terms of partial exponential diagrams. We say that a diagram $D \times B \xrightarrow{f} C$ is a ***partial exponential diagram*** if for every morphism $g : A \times B \to C$, there is *at most* one $h : A \to D$ such that

$$\begin{tikzcd} D \times B \arrow[r, "f"] & C. \\ A \times B \arrow[u, "h\times\mathsf{id}_B"] \arrow[ur, "g"'] & \end{tikzcd}$$

We have dropped the condition for the existence of $h$ from the definition of exponential diagrams in Section 1.1.6. In general, the word “partial” stipulates that one requires uniqueness, but not existence, while the word “weak” or the prefix “pre-” indicates the opposite.

A Henkin representation of a ccc $\mathbf{C}$ can now be characterized as a finite product preserving functor $H : \mathbf{C} \to \mathbf{D}$ such that for all $A, B \in \mathbf{C}$, the arrow $H(B^A) \times HA \xrightarrow{H\varepsilon_{A,B}} HB$ is a partial exponential diagram. The advantage of this definition is that it makes sense for a category $\mathbf{D}$ with finite products, even if it is not cartesian-closed. Our definition of Henkin representations for *PL*-categories in Section 5.4.2 will make use of a similar notion of partial $\forall$-diagrams.

### 5.1.2 Henkin representations and well-pointed ccc’s

**Definition.** An object $A$ is ***well-pointed*** if for every $f \neq g : A \to B$, there is a point $p : 1 \to A$ such that $f \circ p \neq g \circ p$. A category $\mathbf{D}$ is well-pointed if all its objects are well-pointed.

Note that for a ccc $\mathbf{D}$, the following are equivalent:

1. $\mathbf{D}$ is well-pointed.
2. The point functor $\Gamma = (1, -)$ is an embedding.
3. $\Gamma$ is a Henkin representation.

**Proposition 5.4.** *Every ccc representation $F : \mathbf{C} \to \mathbf{D}$ from a ccc $\mathbf{C}$ into a well-pointed ccc $\mathbf{D}$ gives rise to a Henkin representation $H = \Gamma \circ F : \mathbf{C} \to \mathscr{S}$. Conversely, every Henkin representation $H : \mathbf{C} \to \mathscr{S}$ arises in this way.*

*Proof.* If $\mathbf{D}$ is well-pointed, then $\Gamma \circ F : \mathbf{C} \to \mathscr{S}$ is a Henkin representation by Remark 5.3. For the converse, suppose $H : \mathbf{C} \to \mathscr{S}$ is a Henkin representation. Define $\mathbf{D}$ by $|\mathbf{D}| = |\mathbf{C}|$ and $\mathbf{D}(A, B) = H(B^A)$. Composition and identities are given by the respective $H$-images of the canonical morphisms $\circ : C^B \times B^A \to C^A$ and $\mathsf{id}^\star : 1 \to A^A$ in $\mathbf{C}$. Associativity and the identity laws follow from the commutativity of the following diagrams in $\mathbf{C}$, and of their images under $H$:

$$\begin{tikzcd} D^C \times C^B \times B^A \arrow[r, "\mathsf{id}\times\circ"] \arrow[d, "\circ\times\mathsf{id}"'] & D^C \times C^A \arrow[d, "\circ"] \\ D^B \times B^A \arrow[r, "\circ"'] & D^A \end{tikzcd} \qquad \begin{tikzcd} B^A \arrow[r, "\langle\mathsf{id},\mathsf{id}^\star\rangle"] \arrow[d, "\langle\mathsf{id}^\star,\mathsf{id}\rangle"'] \arrow[dr, "\mathsf{id}"] & B^A \times A^A \arrow[d, "\circ"] \\ B^B \times B^A \arrow[r, "\circ"'] & B^A \end{tikzcd}$$

Define $F : \mathbf{C} \to \mathbf{D}$ as the identity on objects, and by sending $f : A \to B$ to $H(f^\star) : 1 \to H(B^A) = \mathbf{D}(A, B)$. It is routine to check that $\mathbf{D}$ is a well-pointed ccc, that $F$ is a ccc representation, and $\Gamma \circ F = H$. $\square$

### 5.1.3 Freely adjoining arrows to a ccc

If $A$ is an object of a ccc $\mathbf{C}$, let $\mathbf{C}[1 \xrightarrow{x} A]$ be the ccc obtained from $\mathbf{C}$ by freely adjoining an indeterminate arrow $x : 1 \to A$. The category $\mathbf{C}[1 \xrightarrow{x} A]$, together with the canonical ccc-representation $\jmath : \mathbf{C} \to \mathbf{C}[1 \xrightarrow{x} A]$, is uniquely determined by the following universal property: for every ccc-representation $F : \mathbf{C} \to \mathbf{D}$ and every arrow $f : 1 \to FA$ in $\mathbf{D}$, there is a unique ccc-representation $\hat{F} : \mathbf{C}[1 \xrightarrow{x} A] \to \mathbf{D}$ such that

$$\begin{tikzcd}
\mathbf{C} \arrow[r, "\jmath"] \arrow[dr, "F"'] & \mathbf{C}[1 \xrightarrow{x} A] \arrow[d, "\hat{F}"] \\
& \mathbf{D}
\end{tikzcd}$$

and $\hat{F}x = f$. The category $\mathbf{C}' = \mathbf{C}[1 \xrightarrow{x} A]$ has a concrete description as the Kleisli category of the comonad $T(B) = A \times B$ (see Lambek and Scott [34]). This means, the objects of $\mathbf{C}'$ are those of $\mathbf{C}$, and the hom-sets are given by $\mathbf{C}'(B, C) = \mathbf{C}(A \times B, V)$. The identity at $B$ in $\mathbf{C}'$ is $\pi' : A \times B \to B$ in $\mathbf{C}$, and the composition $g \circ f$ in $\mathbf{C}'$ is $A \times B \xrightarrow{\langle \pi, f \rangle} A \times C \xrightarrow{g} D$ in $\mathbf{C}$. $x : 1 \to A$ in $\mathbf{C}'$ is $\mathsf{id} : A \to A$ in $\mathbf{C}$. The canonical functor $\jmath : \mathbf{C} \to \mathbf{C}'$ sends $f : B \to C$ to $f \circ \pi' : A \times B \to C$.

It is an interesting question to ask which properties are preserved or reflected by the canonical functor $\jmath : \mathbf{C} \to \mathbf{C}[1 \xrightarrow{x} A]$. We will pay particular attention to the question under what conditions $\jmath$ is an embedding, and under what conditions it is faithful (*i.e.*, isomorphism reflecting).

**Definition.** In any category, a morphism $f : A \to B$ is called a ***cover*** if, whenever $f$ factors through a monic $m$,

$$\begin{tikzcd}
A \arrow[r, "f", -{Triangle[open]}] \arrow[dr] & B \\
& U, \arrow[u, tail, "m"']
\end{tikzcd}$$

then $m$ is necessarily iso. We sometimes write $f : A \twoheadrightarrow B$ for a cover. Notice that any morphism $f : A \to B$ with a right inverse $f \circ g = \mathsf{id}_B$ is a cover, called a ***split cover***. Also notice that any $f$ is iso iff it is a monic cover. An object $A$ is called ***well-supported*** if for each object $B$, the second projection $\pi' : A \times B \twoheadrightarrow B$ is a cover.

An object $A$ is ***partially initial*** if every hom-set $(A, B)$ has at most one element.

**Lemma 5.5.** *Suppose $F : \mathbf{C} \to \mathbf{D}$ has a right adjoint. Then $F$ preserves epics and partial initial objects. Moreover, if $\mathbf{C}$ has pullbacks, then $F$ preserves covers.*

*Proof.* Let $\varphi : F \dashv G$ be the adjunction. Suppose $e : B \twoheadrightarrow C$ is epic and

$$\begin{tikzcd}
FB \arrow[r, "Fe"] & FC \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & D & \Rightarrow & B \arrow[r, "e"] & C \arrow[r, shift left, "\varphi g"] \arrow[r, shift right, "\varphi h"'] & GD,
\end{tikzcd}$$

which implies $\varphi g = \varphi h$, hence $g = h$. Hence $Fe$ is epic. Dually, right adjoints preserve monics. Now suppose $A$ is partially initial. Then $|(FA, B)| \cong |(A, GB)| \leqslant 1$, hence $FA$ is partially initial. Now assume $\mathbf{C}$ has pullbacks, and suppose $f : A \twoheadrightarrow B$ is a cover. Assume $Ff = FA \xrightarrow{g} U \overset{m}{\rightarrowtail} FB$. Since $G$ is a right adjoint, $Gm : GU \rightarrowtail GFB$ is monic, and we can consider

$$\begin{tikzcd}
A \arrow[dr] \arrow[drr, "f", -{Triangle[open]}] \arrow[ddr, "\varphi g"'] & & & & & & & \\
& \cdot \arrow[r, "m'"] \arrow[d, "u'"'] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "u"] & \Rightarrow & B \arrow[dl] \arrow[d, "u"] & \Rightarrow & FB \arrow[dl] \arrow[d, "\mathsf{id}"] \\
& GU \arrow[r, tail, "Gm"'] & GFB & GU \arrow[r, tail, "Gm"'] & GFB & U \arrow[r, tail, "m"'] & FB,
\end{tikzcd}$$

where $u$ is the unit of the adjunction. Notice that pullbacks always preserve monics, hence $m'$ is monic, and it must be iso since $f$ is a cover. The last diagram implies that $m$ is iso, and it follows that $Ff$ is a cover. $\square$

**Lemma 5.6.** *Let $\mathbf{C}' = \mathbf{C}[1 \xrightarrow{x} A]$ and $\jmath : \mathbf{C} \to \mathbf{C}'$ the canonical functor. The following hold:*

1. *$\jmath$ preserves epics, monic cones, partial initial objects, and well-pointed objects.*
2. *$\jmath$ is an embedding if and only if the unique morphism $A \twoheadrightarrow 1$ is epic in $\mathbf{C}$.*
3. *$\jmath$ is faithful if and only if it is an embedding and $A$ is well-supported.*

*Proof.* 1.: Recall that in a ccc, the product functor $T(B) = A \times B$ has a right adjoint; hence it preserves epics and partial initial objects by Lemma 5.5. Now suppose $f : B \twoheadrightarrow C$ is epic in $\mathbf{C}$ and $g \circ \jmath f = h \circ \jmath f$ in $\mathbf{C}'$. By the characterization of $\mathbf{C}[1 \xrightarrow{x} A]$, this means

$$\begin{tikzcd} A \times B \arrow[r, two heads, "\mathsf{id}_A \times f"] & A \times C \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & D, \end{tikzcd}$$

holds in $\mathbf{C}$, and hence $g = h$. Thus, $\jmath f$ is epic. Now, suppose $B$ is a partial initial object in $\mathbf{C}$. Then $A \times B$ is also partially initial, hence $|\mathbf{C}'(B, C)| = |\mathbf{C}(A \times B, C)| \leqslant 1$, hence $B$ is partially initial in $\mathbf{C}'$. Moreover, $T$, and hence $\jmath$, preserves monic cones. Now, suppose $B$ is well-pointed in $\mathbf{C}$, and suppose $f \neq g : B \to C$ in $\mathbf{C}'$. Then $f \neq g : A \times B \to C$, hence $f^\star \neq g^\star : B \to C^A$ in $\mathbf{C}$, and since $B$ is well-pointed, there is $p : 1 \to B$ with $f^\star \circ p \neq g^\star \circ p$. This implies $f \circ (\mathsf{id}_A \times p) \neq g \circ (\mathsf{id}_A \times p)$, hence $f \circ \jmath p \neq g \circ \jmath p$ in $\mathbf{C}'$.

2.: $\Rightarrow$: Certainly $A \twoheadrightarrow 1$ is epic in $\mathbf{C}[1 \xrightarrow{x} A]$ (it splits). But embeddings reflect epics.

$\Leftarrow$: Suppose $A \twoheadrightarrow 1$ is epic in $\mathbf{C}$. Then $\pi' : A \times B \twoheadrightarrow B$ is epic for all $B$. Consider $f, g : B \to C$ in $\mathbf{C}$ with $\jmath f = \jmath g$. Then $f \circ \pi' = g \circ \pi' : A \times B \to C$, hence $f = g$.

3.: First notice that any monic-preserving embedding $F$ reflects isos iff it reflects covers: Suppose $F$ reflects isos and $Ff$ is a cover. Suppose $f$ factors through a monic $m$, then $Ff$ factors through $Fm$, hence $Fm$ is iso, hence $m$ is iso, hence $f$ is a cover. Conversely, suppose $F$ reflects covers. If $Ff$ is iso, then $Ff$, hence $f$, is a monic cover, hence an iso. Since $\jmath : \mathbf{C} \to \mathbf{C}[1 \xrightarrow{x} A]$ preserves monics, it suffices to show that $\jmath$ reflects covers iff $A$ is well-supported.

$\Rightarrow$: Suppose $\jmath$ reflects covers. Clearly, $A \times B \to\!\!\triangleright\, B$ is a split cover in $\mathbf{C}[1 \xrightarrow{x} A]$, hence a cover in $\mathbf{C}$, making $A$ well-supported in $\mathbf{C}$.

$\Leftarrow$: Suppose $A$ is well-supported in $\mathbf{C}$, and suppose $f : C \to B$ is such that $Ff$ is a cover in $\mathbf{C}[1 \xrightarrow{x} A]$. Suppose $f$ factors through a monic $m : D \rightarrowtail C$. Then $Ff$ factors through $Fm$, hence $Fm$ is iso in $\mathbf{C}[1 \xrightarrow{x} A]$. This means, there is $m^{-1} \in \mathbf{C}'(C, D) = \mathbf{C}(A \times C, D)$ such that in $\mathbf{C}$,

$$\begin{tikzcd} A \times C \arrow[r, "\pi'"] \arrow[dr, "m^{-1}"'] & C \\ & D. \arrow[u, tail, "m"'] \end{tikzcd}$$

But $\pi'$ is a cover, therefore $m$ an iso in $\mathbf{C}$. This shows $f$ is a cover. $\square$

**Lemma 5.7.** *If $f \neq g : A \to B$ in $\mathbf{C}$, then $\jmath f \circ x \neq \jmath g \circ x$ in $\mathbf{C}[1 \xrightarrow{x} A]$.*

*Proof.* $\jmath f \circ x = f \circ \pi' \circ \langle \mathsf{id}_A, \mathsf{id}_A \rangle = f$ in $\mathbf{C}$, and similarly for $\jmath g \circ x$. $\square$

**Proposition 5.8.** *Let $\mathbf{C}$ be a small ccc. If $\mathcal{A} \subseteq |\mathbf{C}|$ is a set of objects such that $A \twoheadrightarrow 1$ for all $A \in \mathcal{A}$, then there is a ccc-embedding $F_\mathcal{A} : \mathbf{C} \to \mathbf{D}$ such that $FA$ is well-pointed for all $A \in \mathcal{A}$. $F_\mathcal{A}$ preserves monic cones. Moreover, if each $A \in \mathcal{A}$ is well-supported, then $F_\mathcal{A}$ can be chosen to be faithful, i.e. isomorphism-reflecting.*

*Proof.* We adjoin countably many arrows $1 \to A$ to each $A \in \mathcal{A}$. More precisely, let $I$ be the directed poset of finite subsets $X \subseteq \mathbb{N} \times \mathcal{A}$, ordered by inclusion. Let $\Delta : I \to \mathcal{CCC}$ be the diagram that associates to each $X = \{\langle i_1, A_1\rangle, \dots, \langle i_n, A_n\rangle\} \in I$ a ccc $\mathbf{C}[X] := \mathbf{C}[1 \xrightarrow{x_{i_1,A_1}} A_1, \dots, 1 \xrightarrow{x_{i_n,A_n}} A_n]$, and to each inclusion $\iota : X \hookrightarrow Y \in I$ the canonical ccc-representation $\Delta_\iota : \mathbf{C}[X] \to \mathbf{C}[Y]$. Notice that by Lemma 5.6, $\Delta_\iota$ is a monic-preserving embedding, and moreover, if each $A \in \mathcal{A}$ is well-supported, then $\Delta_\iota$ is faithful. We can take $\mathbf{D}$ to be the colimit of the diagram $\Delta$. Concretely, assume that each $\mathbf{C}[X]$ has the same objects as $\mathbf{C}$, and that the embeddings $\Delta_\iota$ are actual inclusions on hom-sets. Then $\mathbf{D}$ can be described as follows: the objects of $\mathbf{D}$ are those of $\mathbf{C}$, and the hom-set $\mathbf{D}(B, C)$ is the directed union of the hom-sets $\mathbf{C}[X](B, C)$, where $X \in I$. One checks that $\mathbf{D}$ is a ccc and that the inclusion $F : \mathbf{C} \hookrightarrow \mathbf{D}$ is a ccc-embedding preserving collective monics, and moreover, $F$ is faithful if the $\Delta_\iota$ are. To show that $A$ is well-pointed in $\mathbf{D}$, let $A \in \mathcal{A}$ and assume $f \neq g : A \to B$ in $\mathbf{D}$. Then there is $X \in I$ with $f, g \in \mathbf{C}[X]$. Let $\langle i, A\rangle \notin X$ and consider $\mathbf{C}[X, 1 \xrightarrow{x_{i,A}} A]$: one has $f \circ x_{i,A} \neq g \circ x_{i,A}$ by Lemma 5.7. Hence $A$ is well-pointed in $\mathbf{D}$. □

### 5.1.4 Henkin representation theorems

A Henkin representation theorem characterizes those ccc's which can be Henkin embedded in a given category, or in a category from a given class. We consider Henkin representations into the category of non-empty sets $\mathscr{S}^+$, into the category of sets $\mathscr{S}$, and into a category $\mathscr{S}^P$ of presheaves over a poset $P$. We will see in Section 5.3 how each of these target categories corresponds to a certain class of models of the simply-typed lambda calculus. The first one corresponds to Friedman's set-theoretic models with non-empty types [21]; the second one corresponds to set-theoretic models with possibly empty types, as investigated by Meyer *et al.* [39], and the third one corresponds to Mitchell and Moggi's Kripke models [42]. Our Henkin representation theorems will translate into completeness theorems for each of these classes of models.

Representation Theorems for cartesian-closed categories have been considered in the papers of Čubrić [14] and Simpson [60]. The difference to our representation theorems is that Čubrić and Simpson work with strict ccc representations rather than Henkin representations, and they only consider representations of a *free* cartesian-closed category.

#### Henkin representations in $\mathscr{S}^+$

**Theorem 5.9.** *A small ccc $\mathbf{C}$ can be Henkin-embedded in $\mathscr{S}^+$ if and only if for every object $A$, the morphism $A \twoheadrightarrow 1$ is epic.*

*Proof.* $\Rightarrow$: In $\mathscr{S}^+$, one has $A \twoheadrightarrow 1$ for all $A$; moreover, embeddings reflect epics.

$\Leftarrow$: Consider the ccc-embedding $F_{\mathcal{A}} : \mathbf{C} \to \mathbf{D}$ from Proposition 5.8, with $\mathcal{A} = |\mathbf{C}|$. Then $\mathbf{C} \xrightarrow{F_{\mathcal{A}}} \mathbf{D} \xrightarrow{\Gamma} \mathscr{S}^+$ is a Henkin-embedding. □

**Corollary 5.10.** *If for every object $A$ in a small ccc $\mathbf{C}$, the morphism $A \twoheadrightarrow 1$ is epic and $A$ is well-supported, then there is a faithful (i.e., isomorphism-reflecting) Henkin-embedding $H : \mathbf{C} \to \mathscr{S}^+$.* □

#### Henkin representations in $\mathscr{S}$

**Definition.** A cartesian-closed category $\mathbf{C}$ is called ***special*** if for every object $A$, either the morphism $A \twoheadrightarrow 1$ is epic, or $A$ is partially initial.

**Theorem 5.11.** *A small ccc $\mathbf{C}$ can be Henkin-embedded in $\mathscr{S}$ if and only if it is special.*

*Proof.* $\Rightarrow$: The category $\mathscr{S}$ is special, because each non-empty set $A$ satisfies $A \twoheadrightarrow 1$, while the empty set is (partially) initial. Moreover, embeddings reflect epics and partial initial objects, and hence specialness.

$\Leftarrow$: Suppose $\mathbf{C}$ is special. Consider the ccc-embedding $F_{\mathcal{A}} : \mathbf{C} \to \mathbf{D}$ from Proposition 5.8, with $\mathcal{A} = \{A \in |\mathbf{C}| \mid A \twoheadrightarrow 1\}$. Then each $A \in \mathcal{A}$ is well-pointed in $\mathbf{D}$ by construction of $\mathbf{D}$; moreover, each $A \notin \mathcal{A}$ is partially initial and therefore trivially well-pointed. Hence $\mathbf{C} \xrightarrow{F_{\mathcal{A}}} \mathbf{D} \xrightarrow{\Gamma} \mathscr{S}$ is a Henkin-embedding. □

**Corollary 5.12.** *Let $M$ be a monoid, i.e. a one-object category. A small ccc $\mathbf{C}$ can be Henkin-embedded in $\mathscr{S}^M$ if and only if $\mathbf{C}$ is special.*

*Proof.* $\Rightarrow$: $\mathscr{S}^M$ is special, and embeddings reflect specialness.

$\Leftarrow$: There is an obvious ccc-embedding $H : \mathscr{S} \to \mathscr{S}^M$ which preserves monics. If $\mathbf{C}$ is special, it can be Henkin-embedded in $\mathscr{S}$ and hence, by Remark 5.3 in $\mathscr{S}^M$. $\square$

**Corollary 5.13.** *Let $I$ be a set. A small ccc $\mathbf{C}$ can be Henkin-embedded in $\mathscr{S}^I$ if and only if there is a family $(\sim_i)_{i\in I}$ of ccc-congruences on $\mathbf{C}$ such that each quotient $\mathbf{C}/{\sim_i}$ is special, and such that $\bigcap_{i\in I} \sim_i$ is the identity relation.* $\square$

*Remark.* If a ccc $\mathbf{C}$ has an object $A$ such that $A$ is partially initial *and* $A \twoheadrightarrow 1$, then $\mathbf{C}$ is a preorder, *i.e.* every hom-set has at most one element. Indeed, if $f, g : B \to C$, then

$$\begin{tikzcd} A \arrow[r, two heads] & 1 \arrow[r, shift left, "f^\star"] \arrow[r, shift right, "g^\star"'] & C^B, \end{tikzcd}$$

hence $f = g$. As a consequence, if $\mathbf{C}$ has a non-trivial hom-set, then any Henkin embedding $\mathbf{C} \to \mathscr{S}$ not only reflects, but also preserves partial initial objects and epics $A \twoheadrightarrow 1$.

### Henkin representations in $\mathscr{S}^P$

Any small ccc $\mathbf{C}$ can be ccc-embedded in a category of presheaves $\mathscr{S}^{\mathbf{A}}$, for instance by the Yoneda embedding (see Example 1.5). If one takes $\mathbf{A}$ to be a poset, it is still possible to obtain a Henkin embedding:

**Theorem 5.14.** *Any small ccc $\mathbf{C}$ can be Henkin-embedded in $\mathscr{S}^P$ for some poset $P$. Moreover, the embedding preserves monic cones.*

Let $\mathbf{A}$ and $\mathbf{B}$ be small categories, and let $F : \mathbf{A} \to \mathbf{B}$ be a functor. This induces a functor $\mathscr{S}^F : \mathscr{S}^{\mathbf{B}} \to \mathscr{S}^{\mathbf{A}}$, which we denote by $F_*$. Note that $F_*$ always preserves monic cones and limits, since these are taken pointwise in $\mathscr{S}^{\mathbf{A}}$ and $\mathscr{S}^{\mathbf{B}}$. The following two lemmas give sufficient conditions for $F_*$ to be a Henkin representation, respectively, an embedding.

**Definition.** A functor $F : \mathbf{A} \to \mathbf{B}$ is called ***left-full*** if for all $g : FA \to B$ in $\mathbf{B}$, there exists $f : A \to A'$ in $\mathbf{A}$ such that $B = FA'$ and $g = Ff$.

**Lemma 5.15.** *If $F : \mathbf{A} \to \mathbf{B}$ is left-full, then $F_* : \mathscr{S}^{\mathbf{B}} \to \mathscr{S}^{\mathbf{A}}$ is a Henkin representation.*

*Proof.* We need to show that the canonical natural transformation $\varphi : F_*(Q^P) \to F_*Q^{F_*P}$ is monic for all $P, Q \in \mathscr{S}^{\mathbf{B}}$. Let $A \in \mathbf{A}$. Unraveling the definition of exponentiation in a functor category yields that $\varphi_A : \mathscr{S}^{\mathbf{B}}(\mathbf{B}(FA,-) \times P, Q) \to \mathscr{S}^{\mathbf{A}}(\mathbf{A}(A,-) \times F_*P, F_*Q)$ is given by $(\varphi_A\eta)_{A'}(f,x) = \eta_{FA'}(Ff,x)$, where $\eta : \mathbf{B}(FA,-) \times P \to Q$, $A' \in A$, $f : A \to A'$ and $x \in (F_*P)A' = P(FA')$. To show that $\varphi_A$ is one-to-one, assume $\eta \neq \eta' \mathbf{A}(FA,-) \times P \to Q$. Then there are $B \in \mathbf{B}$, $g : FA \to B$ and $x \in PB$ such that $\eta_B(g,x) \neq \eta'_B(g,x)$. Since $F$ is left-full, there is $f : A \to A'$ in $\mathbf{A}$ such that $B = FA'$ and $g = Ff$, hence

$$(\varphi_A\eta)_{A'}(f,x) = \eta_{FA'}(Ff,x) = \eta_B(g,x) \neq \eta'_B(g,x) = \eta'_{FA'}(Ff,x) = (\varphi_A\eta')_{A'}(f,x),$$

and therefore $\varphi_A\eta \neq \varphi_A\eta'$. This shows that $\varphi_A$ is one-to-one for every $A$, hence $\varphi$ is monic. $\square$

**Lemma 5.16.** *If $F : \mathbf{A} \to \mathbf{B}$ is onto objects, then $F_* : \mathscr{S}^{\mathbf{B}} \to \mathscr{S}^{\mathbf{A}}$ is an embedding.*

*Proof.* Let $P, Q \in \mathscr{S}^{\mathbf{B}}$ and $\eta \neq \eta' : P \to Q$. Then $\eta_B \neq \eta'_B : PB \to QB$ for some $B \in \mathbf{B}$. Let $A \in \mathbf{A}$ with $B = FA$. Then $(F_*\eta)_A = \eta_{FA} \neq \eta'_{FA} = (F_*\eta')_A$, hence $F_*\eta \neq F_*\eta'$. $\square$

Table 5.1: Typing rules for the simply-typed lambda calculus

$$(var)\quad \frac{}{\Gamma, x{:}\sigma \rhd x : \sigma}$$

$$(const)\quad \frac{}{\Gamma \rhd c^{\sigma} : \sigma}$$

$$(*)\quad \frac{}{\Gamma \rhd * : 1}$$

$$(pair)\quad \frac{\Gamma \rhd M : \sigma \qquad \Gamma \rhd N : \tau}{\Gamma \rhd \langle M, N\rangle : \sigma \times \tau}$$

$$(\pi_1)\quad \frac{\Gamma \rhd M : \sigma \times \tau}{\Gamma \rhd \pi_1 M : \sigma}$$

$$(\pi_2)\quad \frac{\Gamma \rhd M : \sigma \times \tau}{\Gamma \rhd \pi_2 M : \tau}$$

$$(app)\quad \frac{\Gamma \rhd M : \sigma \to \tau \qquad \Gamma \rhd N : \sigma}{\Gamma \rhd MN : \tau}$$

$$(abs)\quad \frac{\Gamma, x{:}\sigma \rhd M : \tau}{\Gamma \rhd \lambda x{:}\sigma.M : \sigma \to \tau}$$

$$(weaken)\quad \frac{\Gamma \rhd M : \sigma \qquad \Gamma \subseteq \Gamma'}{\Gamma' \rhd M : \sigma}$$

The proof of Theorem 5.14 now rests on the fact that every small category $\mathbf{A}$ is, in the terminology of Freyd and Scedrov [20], ***dominated*** by some poset $P$, *i.e.* there is a left-full functor $F : P \to \mathbf{A}$ which is onto objects.

**Lemma 5.17 (Freyd, Scedrov [20]).** *Every small category* $\mathbf{A}$ *is dominated by some poset* $P$.

*Proof.* Let the objects of $P$ be finite sequences of objects and morphisms $A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{n-1}} A_n$, ordered by the prefix ordering. Then $P$ is a poset, and the obvious functor $F : P \to \mathbf{A}$ is left-full and onto objects. $\square$

*Proof of Theorem 5.14:* Let $\mathbf{C}$ be a small ccc. Then $\mathbf{C}^{op}$ is dominated by some poset $P$ by Lemma 5.17; let $F : P \to \mathbf{C}^{op}$. By Lemmas 5.15 and 5.16, the functor $F_* : \mathscr{S}^{\mathbf{C}^{op}} \to \mathscr{S}^P$ is a Henkin embedding; moreover it preserves monic cones. By precomposing $F_*$ with the Yoneda embedding, one obtains a Henkin embedding $F_* \circ Y : \mathbf{C} \to \mathscr{S}^P$. $\square$

## 5.2 The interpretation of the simply-typed lambda calculus

### 5.2.1 The simply-typed lambda calculus

Let $TC$ be a set of ***type constants*** $t, u, \ldots$. ***Simple types*** $\sigma, \tau, \ldots$ are given by the grammar:

$$\sigma ::= t \mid 1 \mid \sigma \times \tau \mid \sigma \to \tau$$

Let $\mathcal{V}$ be an infinite set of ***variables*** $x, y, \ldots$. For each type $\sigma$, let $C_\sigma$ be a set of ***individual constants*** $c^\sigma, d^\sigma, \ldots$. The collection $\langle TC, (C_\sigma)_\sigma\rangle$ is also called a ***simply-typed signature***. ***Raw typed lambda terms*** $M, N, \ldots$ are given by the grammar:

$$M ::= x \mid c^\sigma \mid * \mid \langle M, N\rangle \mid \pi_1 M \mid \pi_2 M \mid MN \mid \lambda x{:}\sigma.M \mid M\sigma$$

We have the usual notions of free and bound variables, and we write $\mathrm{FV}(M)$ for the free variables of a term $M$. We identify raw terms up to renaming of bound variables, and we write $M[N/x]$ for the result of substituting $N$ for $x$ in $M$.

Table 5.2: Equational rules for the simply-typed lambda calculus

$$(\textit{refl})\quad \frac{}{\Gamma \rhd M = M : \sigma}$$

$$(\textit{symm})\quad \frac{\Gamma \rhd M = N : \sigma}{\Gamma \rhd N = M : \sigma}$$

$$(\textit{trans})\quad \frac{\Gamma \rhd M = N : \sigma \qquad \Gamma \rhd N = P : \sigma}{\Gamma \rhd M = P : \sigma}$$

$$(\textit{cong}_1)\quad \frac{\Gamma \rhd M = M' : \sigma \qquad \Gamma \rhd N = N' : \tau}{\Gamma \rhd \langle M, N\rangle = \langle M', N'\rangle : \sigma \times \tau}$$

$$(\textit{cong}_2)\quad \frac{\Gamma \rhd M = M' : \sigma \times \tau}{\Gamma \rhd \pi_1 M = \pi_1 M' : \sigma}$$

$$(\textit{cong}_3)\quad \frac{\Gamma \rhd M = M' : \sigma \times \tau}{\Gamma \rhd \pi_2 M = \pi_2 M' : \tau}$$

$$(\textit{cong}_4)\quad \frac{\Gamma \rhd M = M' : \sigma \to \tau \quad \Gamma \rhd N = N' : \sigma}{\Gamma \rhd MN = M'N' : \tau}$$

$$(\textit{cong}_5)\quad \frac{\Gamma, x{:}\sigma \rhd M = M' : \tau}{\Gamma \rhd \lambda x{:}\sigma.M = \lambda x{:}\sigma.M' : \sigma \to \tau}$$

$$(\textit{unit})\quad \frac{}{\Gamma \rhd M = * : 1}$$

$$(\textit{proj}_1)\quad \frac{}{\Gamma \rhd \pi_1\langle M, N\rangle = M : \sigma}$$

$$(\textit{proj}_2)\quad \frac{}{\Gamma \rhd \pi_2\langle M, N\rangle = N : \tau}$$

$$(\textit{surj})\quad \frac{}{\Gamma \rhd \langle \pi_1 M, \pi_2 M\rangle = M : \sigma \times \tau}$$

$$(\beta)\quad \frac{}{\Gamma \rhd (\lambda x{:}\sigma.M)N = M[N/x] : \tau}$$

$$(\eta)\quad \frac{x \notin \mathbf{FV}(M)}{\Gamma \rhd \lambda x{:}\sigma.(Mx) = M : \sigma \to \tau}$$

$$(\textit{add-var})\quad \frac{\Gamma \rhd M = M' : \sigma \qquad \Gamma \subseteq \Gamma'}{\Gamma' \rhd M = M' : \sigma}$$

A ***type assignment*** $\Gamma = x_1{:}\sigma_1, x_2{:}\sigma_2, \ldots, x_m{:}\sigma_m$ is a finite, possibly empty sequence of pairs of a variable and a type, such that $x_i \neq x_j$ for all $i \neq j$. We write $\Gamma \subseteq \Gamma'$ if $\Gamma$ is contained in $\Gamma'$ *as a set*. A ***valid typing judgment*** is an expression of the form $\Gamma \rhd M : \sigma$ which can be derived by the rules in Table 5.1. An ***equation*** of the simply-typed lambda calculus is an expression of the form $\Gamma \rhd M = N : \sigma$, where $\Gamma \rhd M : \sigma$ and $\Gamma \rhd N : \sigma$ are valid typing judgments. If $E$ is an equation and $\mathcal{E}$ is a set of equations, we write $\mathcal{E} \vdash_s E$ if $E$ can be derived from $\mathcal{E}$ by the rules in Table 5.2. As usual, $\mathcal{E}$ is called a ***theory*** if it is closed under derivability, *i.e.* if $\mathcal{E} \vdash_s E$ implies $E \in \mathcal{E}$. The smallest theory of the simply-typed lambda calculus (for a fixed signature) is denoted by $\underline{\lambda}$. It is also called the ***pure*** theory.

### 5.2.2 Strict interpretation in a cartesian-closed category

Fix a simply-typed signature. An ***(strict) interpretation*** $I$ of the simply-typed lambda calculus in a ccc $\mathbf{C}$, which we schematically write as $I : \underline{\lambda} \to \mathbf{C}$, consists of an interpretation of types and an interpretation of typing judgments. A type $\sigma$ is interpreted as an object $[\![\sigma]\!]^I$ of $\mathbf{C}$. A valid typing judgment $\Gamma \rhd M : \tau$ is interpreted as a morphism $[\![\Gamma \rhd M : \tau]\!]^I$. A strict interpretation $I$ is uniquely determined by its values on type constants and individual constants.

Let $I : TC \to |\mathbf{C}|$ be an interpretation of type constants as objects of $\mathbf{C}$. This extends uniquely to an interpretation $[\![\sigma]\!]^I$ of every type:

$$\begin{array}{rcl} [\![t]\!]^I & = & I(t) \\ [\![1]\!]^I & = & 1 \\ [\![\sigma \times \tau]\!]^I & = & [\![\sigma]\!]^I \times [\![\tau]\!]^I \\ [\![\sigma \to \tau]\!]^I & = & ([\![\tau]\!]^I)^{[\![\sigma]\!]^I} \end{array}$$

If $\Gamma = x_1{:}\sigma_1, \ldots, x_m{:}\sigma_m$ is a type assignment, we write $[\![\Gamma]\!]^I = [\![\sigma_1]\!]^I \times \cdots \times [\![\sigma_m]\!]^I$. Let $I_\sigma : C_\sigma \to (1, [\![\sigma]\!]^I)$ be an interpretation of term constants as morphisms of $\mathbf{C}$, for each type $\sigma$. This extends uniquely to an interpretation $\llbracket\Gamma \rhd M : \tau\rrbracket^I$ of valid typing judgments:

$$
\begin{array}{rcl}
\llbracket\Gamma \rhd x_j : \sigma_j\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\pi_j} \llbracket\sigma_j\rrbracket^I, \text{ the } j\text{th projection} \\
\llbracket\Gamma \rhd c^\sigma : \sigma\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\bigcirc} 1 \xrightarrow{I_\sigma(c^\sigma)} \llbracket\sigma\rrbracket^I \\
\llbracket\Gamma \rhd * : 1\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\bigcirc} 1 = \llbracket 1\rrbracket^I \\
\llbracket\Gamma \rhd \langle M, N\rangle : \sigma \times \tau\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\langle\llbracket\Gamma\rhd M:\sigma\rrbracket^I, \llbracket\Gamma\rhd N:\tau\rrbracket^I\rangle} \llbracket\sigma\rrbracket^I \times \llbracket\tau\rrbracket^I = \llbracket\sigma \times \tau\rrbracket^I \\
\llbracket\Gamma \rhd \pi_1 M : \sigma\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\llbracket\Gamma\rhd M:\sigma\times\tau\rrbracket^I} \llbracket\sigma\rrbracket^I \times \llbracket\tau\rrbracket^I \xrightarrow{\pi} \llbracket\sigma\rrbracket^I \\
\llbracket\Gamma \rhd \pi_2 M : \tau\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\llbracket\Gamma\rhd M:\sigma\times\tau\rrbracket^I} \llbracket\sigma\rrbracket^I \times \llbracket\tau\rrbracket^I \xrightarrow{\pi'} \llbracket\sigma\rrbracket^I \\
\llbracket\Gamma \rhd MN : \tau\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{\langle\llbracket\Gamma\rhd M:\sigma\to\tau\rrbracket^I, \llbracket\Gamma\rhd N:\sigma\rrbracket^I\rangle} (\llbracket\tau\rrbracket^I)^{\llbracket\sigma\rrbracket^I} \times \llbracket\sigma\rrbracket^I \xrightarrow{\varepsilon} \llbracket\tau\rrbracket^I \\
\llbracket\Gamma \rhd \lambda x{:}\sigma.M : \sigma \to \tau\rrbracket^I & = & \llbracket\Gamma\rrbracket^I \xrightarrow{(\llbracket\Gamma, x:\sigma\rhd M:\tau\rrbracket^I)^\star} (\llbracket\tau\rrbracket^I)^{\llbracket\sigma\rrbracket^I} = \llbracket\sigma \to \tau\rrbracket^I
\end{array}
$$

**Lemma 5.18.** *The interpretation of the simply-typed lambda calculus in a ccc has the following properties, which are proved by induction on $M$:*

1. **Permutation of Individual Variables.** If $s : \{1, \dots, l\} \to \{1, \dots, m\}$ is injective and FV$(M) \subseteq \{x_{s1}, \dots, x_{sl}\}$, and if $\Gamma' = x_1{:}\sigma_1, \dots x_m{:}\sigma_m$ and $\Gamma = x_{s1}{:}\sigma_{s1}, \dots x_{sl}{:}\sigma_{sl}$ then

$$
\begin{tikzcd}
\llbracket\Gamma'\rrbracket^I \arrow[rr, "\llbracket\Gamma'\rhd M:\tau\rrbracket^I"] \arrow[dr, "\langle\pi_{s1},\dots,\pi_{sl}\rangle"'] & & \llbracket\tau\rrbracket^I. \\
& \llbracket\Gamma\rrbracket^I \arrow[ur, "\llbracket\Gamma\rhd M:\tau\rrbracket^I"'] &
\end{tikzcd}
$$

2. **Term Substitution.** Let $\Gamma = x_1{:}\sigma_1, \dots x_m{:}\sigma_m$ and $\Gamma' = y_1{:}\rho_1, \dots y_l{:}\rho_l$, and suppose $\Gamma \rhd M : \tau$ and $\Gamma' \rhd N_j : \sigma_j$ for $j = 1, \dots, m$. Let $M[\bar{N}/\bar{x}]$ denote the simultaneous substitution of $N_1, \dots, N_n$ for $x_1, \dots, x_n$ in $M$. Then

$$
\begin{tikzcd}
\llbracket\Gamma'\rrbracket^I \arrow[rr, "\llbracket\Gamma'\rhd M[\bar{N}/\bar{x}]:\tau\rrbracket^I"] \arrow[dr, "\langle\llbracket\Gamma'\rhd N_1:\sigma_1\rrbracket^I,\dots,\llbracket\Gamma'\rhd N_n:\sigma_n\rrbracket^I\rangle"'] & & \llbracket\tau\rrbracket^I. \\
& \llbracket\Gamma\rrbracket^I \arrow[ur, "\llbracket\Gamma\rhd M:\tau\rrbracket^I"'] &
\end{tikzcd}
$$

□

We say that an interpretation $I$ ***satisfies*** an equation $\Gamma \rhd M = N : \tau$, in symbols, $I \models \Gamma \rhd M = N : \tau$, if $\llbracket\Gamma \rhd M : \tau\rrbracket^I = \llbracket\Gamma \rhd N : \tau\rrbracket^I$. If $\mathcal{E}$ is a set of equations, then we write $I \models \mathcal{E}$ if $I \models E$ for all $E \in \mathcal{E}$. The set of all equations that an interpretation $I$ satisfies is written Th$(I)$. If $\mathcal{M}$ is a class of ccc's, then we write $\mathcal{E} \models_{\mathcal{M}} E$, for an equation $E$ and a set of equations $\mathcal{E}$, if for every strict interpretation $I$ in a ccc $\mathbf{C} \in \mathcal{M}$, $I \models \mathcal{E}$ implies $I \models E$.

**Proposition 5.19. Soundness of the strict ccc interpretation.**

$$\mathcal{E} \vdash_s E \quad \textit{implies} \quad \mathcal{E} \models_{\mathcal{CCC}} E.$$

□

If $\mathcal{T}$ is a theory and $I : \underline{PL} \to \mathbf{C}$ is an interpretation such that $I \models \mathcal{T}$, then we also write $I : \mathcal{T} \to \mathbf{C}$. An interpretation can be post-composed with a ccc-representation in an evident way: $\mathcal{T} \xrightarrow{I} \mathbf{C} \xrightarrow{F} \mathbf{C}$ is the interpretation $J$ defined by $\llbracket\sigma\rrbracket^J = F\llbracket\sigma\rrbracket^I$ and $\llbracket\Gamma \rhd M : \tau\rrbracket^J = F\llbracket\Gamma \rhd M : \tau\rrbracket^I$.

### 5.2.3 The cartesian-closed category associated to a theory

From a theory $\mathcal{T}$ over a simply-typed signature, one constructs a cartesian-closed category $\mathfrak{F}_{ccc}(\mathcal{T})$ as follows: The objects are simple types $\sigma$, and the morphisms $f_M \in (\sigma, \tau)$ are named by terms $M$ such that $x{:}\sigma \rhd M : \tau$ is a valid typing judgment. Two terms $M$ and $N$ name the same morphism if $\mathcal{T} \vdash_s x{:}\sigma \rhd M = N : \tau$.

**Proposition 5.20.** *The above construction yields a well-defined cartesian-closed category $\mathfrak{F}_{ccc}(\mathcal{T})$. There is a canonical strict interpretation $I_0 : \mathcal{T} \to \mathfrak{F}_{ccc}(\mathcal{T})$ with $[\![\sigma]\!]^{I_0} = \sigma$ and $[\![x{:}\sigma \rhd M : \tau]\!]^{I_0} = f_M : \sigma \to \tau$. Moreover, $I_0$ has the following universal property: For any strict interpretation $J : \mathcal{T} \to \mathbf{C}$, there is a unique ccc-representation $F : \mathfrak{F}_{ccc}(\mathcal{T}) \to \mathbf{C}$ such that*

$$\begin{tikzcd} \mathcal{T} \arrow[d, "I_0"'] \arrow[dr, "J"] & \\ \mathfrak{F}_{ccc}(\mathcal{T}) \arrow[r, "F"'] & \mathbf{C}. \end{tikzcd}$$

□

**Corollary 5.21. Completeness of the strict ccc interpretation.** *Each theory $\mathcal{T}$ of the simply-typed lambda calculus arises as the theory of some ccc-interpretation. Consequently, for any set of equations $\mathcal{E}$,*

$$\mathcal{E} \models_{\mathcal{CCC}} E \quad \textit{implies} \quad \mathcal{E} \vdash_s E.$$

□

### 5.2.4 Henkin representations of a free ccc

**Definition.** The ***free ccc*** over a simply-typed signature is the cartesian-closed category associated to the *pure* theory $\underline{\lambda}$ over that signature.

Čubrić proved in [14] that for any object $A$ in a free ccc, the unique morphism $A \twoheadrightarrow 1$ is epic, and hence the condition of Theorem 5.9 is satisfied. The proof uses a strongly normalizing system of Mints reductions. Let us remark here that, using these Mints reductions, one can show more about the morphism $A \to 1$:

**Proposition 5.22.** *In a free ccc, the morphism $A \to 1$ is a coequalizer of the diagram*

$$\begin{tikzcd} A \times A \arrow[r, shift left, "\pi"] \arrow[r, shift right, "\pi'"'] & A. \end{tikzcd}$$

*Proof sketch:* Let $f : A \to B$ be named by the term $x{:}A \rhd M : B$, and assume $f \circ \pi = f \circ \pi'$. This means $\vdash_s y{:}A, z{:}A \rhd M[y/x] = M[z/x]$. Suppose $M'$ is the unique normal form of $M$ with respect to the system of Mints reductions. Then $M'[y/x]$ and $M'[z/x]$ are the respective unique normal forms of $M[y/x]$ and $M[z/x]$, hence they are syntactically equal. It follows that $M'$ does not contain $x$ freely, and therefore $f$ factors as $A \to 1 \xrightarrow{M'} B$. We already know that the factorization is unique because $A \twoheadrightarrow 1$. □

As a consequence, in a free ccc, every object $A$ is well-supported, *i.e.* $\pi' : A \times B \twoheadrightarrow B$ for all $B$. Indeed, products preserve coequalizers, and coequalizers are covers. With Corollary 5.10, one gets:

**Corollary 5.23.** *Any free ccc has a faithful (i.e. isomorphism-reflecting) Henkin-embedding into $\mathscr{S}^+$.* □

### 5.2.5 The non-strict interpretation of the simply-typed lambda calculus

Let $\mathbf{C}$ be a ccc. A ***non-strict interpretation*** of the simply-typed lambda calculus $I : \underline{\lambda} \to \mathbf{C}$ is a Henkin representation $H : \mathfrak{F}_{ccc}(\underline{\lambda}) \to \mathbf{C}$. One defines $[\![\sigma]\!]^I = H[\![\sigma]\!]^{I_0}$ and $[\![\Gamma \rhd M : \tau]\!]^I = H[\![\Gamma \rhd M : \tau]\!]^{I_0}$. A non-strict interpretation $I$ ***satisfies*** an equation $\Gamma \rhd M = N : \tau$, in symbols $I \models \Gamma \rhd M = N : \tau$, if $[\![\Gamma \rhd M : \tau]\!]^I = [\![\Gamma \rhd N : \tau]\!]^I$. As before, we denote by $\mathrm{Th}(I)$ the set of equations that are satisfied in $I$. For a class $\mathcal{M}$ of ccc's we write $\mathcal{E} \models_{\mathcal{M}}^{\textit{non-strict}} E$ if any non-strict interpretation in some $\mathbf{C} \in \mathcal{M}$ that satisfies $\mathcal{E}$ also satisfies $E$. The following soundness theorem is an obvious consequence of Lemma 5.2:

**Proposition 5.24. Soundness of the non-strict ccc-interpretation.**

$$\mathcal{E} \vdash_s E \quad \textit{implies} \quad \mathcal{E} \models^{\textit{non-strict}}_{\mathcal{CCC}} E.$$

□

*Remark.* Completeness is also evident, because the strict ccc-interpretations are among the non-strict ones. More interesting are completeness theorems with respect to certain smaller classes of models. This is the subject of the next section.

## 5.3 From Henkin representation theorems to completeness theorems

### 5.3.1 The problem with empty types

By a ***set-theoretic model*** of the simply-typed lambda calculus, we mean a non-strict interpretation in $\mathscr{S}$. The equational rules for the lambda calculus from Table 5.2 are not complete for equational reasoning in set-theoretic models. This was first noticed by Meyer *et al.* in [39]. If in some model, the interpretation $[\![\sigma]\!]^I$ of a type $\sigma$ is the empty set, then the model satisfies *every* equation of the form

$$x{:}\sigma, \Gamma \rhd M = N : \tau. \tag{5.1}$$

On the other hand, if $[\![\sigma]\!]^I$ is non-empty, then the model satisfies the rule

$$(\textit{non-empty}) \qquad \frac{x{:}\sigma, \Gamma \rhd M = N : \tau \qquad x \notin \mathrm{FV}(M, N)}{\Gamma \rhd M = N : \tau} \tag{5.2}$$

for that type $\sigma$. (By this we mean, for every instance of the rule, if the model satisfies the premise, then it satisfies the conclusion. We also say the rule is ***sound*** for the model.) So in any particular set-theoretic model, for each $\sigma$, either (5.1) or (5.2) holds. However, in a general theory of the simply-typed lambda calculus, this is not true. Meyer and his co-authors give the following example: Let $\sigma$ and $\tau$ be type constants, and let $f : (\sigma \to \sigma \to \sigma) \to \tau$ be an individual constant. Let $p_1 = \lambda x{:}\sigma.\lambda y{:}\sigma.x$ and $p_2 = \lambda x{:}\sigma.\lambda y{:}\sigma.y$. Then the following is sound for any set-theoretic interpretation:

$$\frac{x{:}\sigma \rhd fp_1 = fp_2 : \tau}{\rhd fp_1 = fp_2 : \tau}. \tag{5.3}$$

This is because, if $[\![\sigma]\!]^I$ is empty, then $p_1 = p_2$ holds as a consequence of (5.1), while if $[\![\sigma]\!]^I$ is non-empty, then (5.3) follows from (5.2). On the other hand, (5.3) is not sound for arbitrary theories of the lambda calculus: specifically, let **2** be the poset with two elements $\bot \leqslant \top$, and consider the following interpretation $I : \underline{\lambda} \to \mathscr{S}^{\mathbf{2}}$: for types, we let

$$[\![\sigma]\!]^I(\bot) = \emptyset, \qquad [\![\sigma]\!]^I(\top) = \{s_1, s_2\}, \qquad [\![\tau]\!]^I(\bot) = \{t_1, t_2\}, \qquad [\![\tau]\!]^I(\top) = \{u\},$$

with the unique maps $[\![\sigma]\!]^I(\bot) \to [\![\sigma]\!]^I(\top)$ and $[\![\tau]\!]^I(\bot) \to [\![\tau]\!]^I(\top)$. Let $A = [\![\sigma \to \sigma \to \sigma]\!]^I$, and notice that $A(\bot) = A(\top) = \{s_1, s_2\}^{\{s_1,s_2\}^2}$. Let $\pi_2 : \{s_1, s_2\}^2 \to \{s_1, s_2\}$ be the first projection. Now define $[\![f]\!]^I : [\![\sigma \to \sigma \to \sigma]\!]^I \to [\![\tau]\!]^I$ via

$$[\![f]\!]^I(\bot)(x) = \begin{cases} t_1 & \text{if } x = \pi_1 \\ t_2 & \text{else,} \end{cases} \qquad [\![f]\!]^I(\top)(x) = u.$$

With respect to this interpretation, $[\![fp_1]\!]^I(\bot) = t_1 \neq t_2 = [\![fp_2]\!]^I(\bot)$. On the other hand, there is a unique morphism $[\![\sigma]\!]^I \to [\![\tau]\!]^I$. Hence, $x{:}\sigma \rhd fp_1 = fp_2 : \tau$ holds for the interpretation $I$, while $\rhd fp_1 = fp_2 : \tau$ does not. Consequently, the rule (5.3) is not admissible for arbitrary lambda theories.

As the example shows, the equational rules in Table 5.2 are not complete for the class of set-theoretic models. On the other hand, the rule (*non-empty*) is sound only for set-theoretic models with non-empty types. Hence, the need arises to consider the following three classes of models, and their associated completeness theorems, separately:

1. One may consider set-theoretic models where all types are non-empty. This is the classical approach [21]. In this case, the rules in Table 5.2, together with (*non-empty*), are sound and complete.

2. One may consider all set-theoretic models. This necessitates a more elaborate system of inference rules. A sound and complete system was given in [39].

3. One may enlarge the class of models to allow non-set-theoretic ones. The class of Kripke lambda models, introduced in [42], is a convenient such class, and for it, the rules in Table 5.2 are sound and complete.

Each of the three classes of models can be described in terms of Henkin representations, and the completeness theorems in each of the three cases can be derived from the three respective Henkin representation theorems of Section 5.1.4.

### 5.3.2 A categorical analysis of the rule (*non-empty*)

We have already remarked that in a set-theoretic model, the rule (*non-empty*) is sound for a type $\sigma$ if $[\![\sigma]\!]^I$ is non-empty. More generally, if $I : \underline{\lambda} \to \mathbf{C}$ is a non-strict interpretation of the simply-typed lambda calculus, and if $[\![\sigma]\!]^I$ is an object such that $[\![\sigma]\!]^I \twoheadrightarrow 1$ is epic, then the rule (*non-empty*) is sound for $\sigma$ with respect to $I$. Because if $x \notin \mathrm{FV}(M, N)$, and if $[\![x{:}\sigma, \Gamma \rhd M : \tau]\!]^I = [\![x{:}\sigma, \Gamma \rhd N : \tau]\!]^I$, then, using Lemma 5.18,

$$[\![\sigma]\!]^I \times [\![\Gamma]\!]^I \xrightarrow{\pi'} \!\!\!\!\rightarrow [\![\Gamma]\!]^I \underset{[\![\Gamma \rhd N:\tau]\!]^I}{\overset{[\![\Gamma \rhd M:\tau]\!]^I}{\underset{\to}{\overset{\to}{\div}}}} [\![\tau]\!]^I,$$

and hence $I \models \Gamma \rhd M = N : \tau$. Conversely, assume that (*non-empty*) is sound for $\sigma$ in some theory $\mathcal{T}$. Then $[\![\sigma]\!]^{I_0} \twoheadrightarrow 1$ is epic in the ccc $\mathfrak{F}_{ccc}(\mathcal{T})$, for the canonical interpretation $I_0$.

### 5.3.3 Set-theoretic models with non-empty types

Fix a simply-typed signature. By a ***set-theoretic model with non-empty types*** of the simply-typed lambda calculus, we mean a non-strict interpretation $I : \underline{\lambda} \to \mathscr{S}^+$. We write $\mathcal{E} \models^{\textit{non-strict}}_{\mathscr{S}^+} E$ for semantic consequence with respect to that class of models. We write $\mathcal{E} \vdash^{\textit{non-empty}}_{s} E$ if $E$ can be derived from the equations $\mathcal{E}$ by the usual simply-typed lambda calculus rules, together with the rule (*non-empty*).

**Theorem 5.25. Soundness and Completeness for non-strict interpretations in $\mathscr{S}^+$.** *The rule (non-empty) is sound for non-strict interpretations in $\mathscr{S}^+$. Moreover, any theory that is closed under (non-empty) arises from such an interpretation. As a consequence,*

$$\mathcal{E} \models^{\textit{non-strict}}_{\mathscr{S}^+} E \quad \textit{if and only if} \quad \mathcal{E} \vdash^{\textit{non-empty}}_{s} E.$$

*Proof.* Soundness: It follows from the remarks in Section 5.3.2 that the rule (*non-empty*) is sound for $\models^{\textit{non-strict}}_{\mathscr{S}^+}$.

Completeness: Assume $\mathcal{E} \models^{\textit{non-strict}}_{\mathscr{S}^+} E$. Let $\mathcal{T}$ be the theory generated by $\mathcal{E}$ and (*non-empty*). We need to show $E \in \mathcal{T}$. Let $I_0 : \mathcal{T} \to \mathfrak{F}_{ccc}(\mathcal{T})$ be the canonical interpretation. Then $A \twoheadrightarrow 1$ for all objects of $\mathfrak{F}_{ccc}(\mathcal{T})$, hence there is a Henkin embedding $H : \mathfrak{F}_{ccc}(\mathcal{T}) \to \mathscr{S}^+$ by Henkin Representation Theorem 5.9. Let $I = H \circ I_0 : \mathcal{T} \to \mathscr{S}^+$. Then $\mathrm{Th}(I) = \mathrm{Th}(I_0) = \mathcal{T}$, hence $I \models \mathcal{E} \Rightarrow I \models E \Rightarrow E \in \mathcal{T}$. □

### 5.3.4 Set-theoretic models with empty types

For reasoning about possibly empty types, we use the extended proof system of Meyer *et al.* [39]. Fix a simply-typed signature. An ***emptiness assertion*** is an expression $e(\tau)$, where $\tau$ is a type. We use the letter $\Delta$ to denote a sequence of emptiness assertions, and we write $\Delta \subseteq \Delta'$ if $\Delta$ is contained in $\Delta'$ as a set. An

Table 5.3: Rules for the simply-typed lambda calculus with emptiness assertions

$$(\textit{empty}) \qquad \frac{}{\Delta, e(\sigma), x{:}\sigma, \Gamma \rhd M = N : \tau}$$

$$(\textit{cases}) \qquad \frac{\Delta, e(\sigma), \Gamma \rhd M = N : \tau \qquad \Delta, x{:}\sigma, \Gamma \rhd M = N : \tau}{\Delta, \Gamma \rhd M = N : \tau}$$

$$(\textit{add-emp}) \qquad \frac{\Delta, \Gamma \rhd M = N : \tau \qquad \Delta \subseteq \Delta'}{\Delta', \Gamma \rhd M = N : \tau}$$

***extended equation*** is an expression of the form $\Delta, \Gamma \rhd M = N : \tau$, where $\Gamma \rhd M : \tau$ and $\Gamma \rhd N : \tau$ are valid typing judgments (note that no $\Delta$ appears in typing judgments). The intuitive meaning of an extended equation $e(\tau_1), \ldots, e(\tau_k), \Gamma \rhd M = N : \tau$ is: if $\tau_1$ through $\tau_k$ are empty, then $\Gamma \rhd M = N : \tau$ holds. We freely use suggestive notation such as $e(\sigma) \rhd E$ to denote an extended equation whose left-hand side contains an emptiness assertion $e(\sigma)$, and $x{:}\sigma \rhd E$ to denote an extended equation whose left-hand side contains a type assertion $x{:}\sigma$, where $E$ may contain other emptiness or type assertions.

We consider three special rules for extended equations, which are shown in Table 5.3. Notice that in the rule (*cases*), the variable $x$ cannot be free in $M, N$. We write $\mathcal{E} \vdash_s^{\mathit{ext}} E$ for derivability using these rules, together with the equational rules of the simply-typed lambda calculus. Throughout this subsection, we will write $E$ for an extended equation, and $\mathcal{E}$ for a set of extended equations. An ***extended theory*** is a set of extended equations that is closed under derivability. If $\mathcal{T}$ is an extended theory, then we write $\mathcal{T}^\circ$ for its subset of equations, *i.e.* those extended equations of $\mathcal{T}$ that contain no emptiness assertions. $\mathcal{T}^\circ$ is a theory, which we call the ***core*** of $\mathcal{T}$.

Recall that a ccc $\mathbf{C}$ is ***special*** if for every object $A$, either $A \twoheadrightarrow 1$ is epic or $A$ is partially initial. Let $I : \underline{\lambda} \to \mathbf{C}$ be an interpretation of the simply-typed lambda calculus in a special ccc, and let $E$ be an extended equation, say, $e(\tau_1), \ldots, e(\tau_k), \Gamma \rhd M = N : \tau$. We say that $I$ ***satisfies*** $E$, in symbols $I \models E$, if

$$[\![\tau_1]\!]^I, \ldots, [\![\tau_k]\!]^I \text{ partially initial} \qquad \Rightarrow \qquad [\![\Gamma \rhd M : \tau]\!]^I = [\![\Gamma \rhd N : \tau]\!]^I.$$

If $\mathcal{M}$ is a class of special ccc's, we write $\mathcal{E} \models_{\mathcal{M}} E$, respectively $\mathcal{E} \models_{\mathcal{M}}^{\textit{non-strict}} E$, if $I \models \mathcal{E}$ implies $I \models E$ for all strict, respectively non-strict, interpretations $I$ in a ccc in $\mathcal{M}$.

**Definition 5.26.** An extended theory $\mathcal{T}$ is called ***principal*** if for each type $\sigma$, either $\mathcal{T}$ contains *all* extended equations of the form $x{:}\sigma \rhd E'$, or it contains *all* extended equations of the form $e(\sigma) \rhd E'$.

**Proposition 5.27. The correspondence between principal extended theories and special ccc's.**

1. *Let $I : \underline{\lambda} \to \mathbf{C}$ be a strict or non-strict interpretation of the simply-typed lambda calculus in a special ccc. Then the set $\mathcal{T} = \{E \mid I \models E\}$ is a principal extended theory.*

2. *Conversely, every principal extended theory arises in this way from some strict interpretation $I$.*

*Proof.* 1.: First, we need to check that $\mathcal{T}$ is indeed an extended theory. It is easily checked that the rules (*empty*), (*cases*) and (*add-emp*) are sound with respect to any interpretation $I$ in a special ccc $\mathbf{C}$. For (*cases*), one uses the fact that $\mathbf{C}$ is special: the conclusion follows from the first premise if $[\![\sigma]\!]^I$ is partially initial, and from the second premise if $[\![\sigma]\!]^I \twoheadrightarrow 1$. The fact that $\mathcal{T}$ is principal follows directly from the definition of $I \models E$: consider any type $\sigma$. If $[\![\sigma]\!]^I$ is partially initial, then $\mathcal{T}$ contains all extended equations of the form $x{:}\sigma \rhd E'$. If $[\![\sigma]\!]^I$ is *not* partially initial, then $\mathcal{T}$ (trivially) contains all extended equations of the form $e(\sigma) \rhd E'$.

2.: Let $\mathcal{T}$ be a principal extended theory. Let $\mathcal{T}^\circ$ be the core of $\mathcal{T}$, *i.e.* the subset of those extended equations of $\mathcal{T}$ that contain no emptiness assertions. Let $\mathbf{C} = \mathfrak{F}_{ccc}(\mathcal{T}^\circ)$ be the cartesian-closed category associated to the theory $\mathcal{T}^\circ$, and let $I_0 : \mathcal{T}^\circ \to \mathbf{C}$ be the canonical interpretation. We show that $\mathbf{C}$ is a special ccc. Consider any object $\sigma$. If $\mathcal{T}$ contains all extended equations of the form $x{:}\sigma \rhd E'$, then $[\![\sigma]\!]^{I_0}$ is partially initial. Otherwise, $\mathcal{T}$ contains all extended equations of the form $e(\sigma) \rhd E'$. Therefore, the first premise of (*cases*) always holds for the type $\sigma$, and hence the rule (*add-emp*) is sound for $\mathcal{T}$ at type sigma. By the remarks in Section 5.3.2 this means that $[\![\sigma]\!]^{I_0} \twoheadrightarrow 1$. Therefore $\mathbf{C}$ is special.

We now claim that $\mathcal{T} \models E$ iff $I_0 \models E$, for any extended equation $E$. Let $E$ be $e(\tau_1), \ldots, e(\tau_k) \rhd E_0$, where $E_0$ is an equation, *i.e.* $E_0$ contains no more emptiness assertions.

First, assume $\mathcal{T} \models E$. Assume that $[\![\tau_1]\!]^{I_0}, \ldots, [\![\tau_k]\!]^{I_0}$ are partially initial in $\mathbf{C}$. Under this hypothesis, we need to show $I_0 \models E_0$. Since $\mathbf{C}$ is the ccc associated to the theory $\mathcal{T}^\circ$, this implies that $\mathcal{T}^\circ \vdash x_i{:}\tau_i \rhd E_0$ for $i = 1 \ldots k$. With $E$, by repeated application of the rules (*cases*) and (*add-emp*), one gets $\mathcal{T} \vdash_s^{ext} E_0$. Since $E_0$ is a (non-extended) equation, it must be in the core, *i.e.* $\mathcal{T}^\circ \vdash_s^{ext} E_0$, hence $I_0 \models E_0$.

Conversely, assume that $\mathcal{T} \not\models E$. We claim that $I_0 \not\models E$. Since $\mathcal{T}$ is a principal extended theory and $\mathcal{T} \not\vdash_s^{ext} E$, it must be the case that $\mathcal{T}$ contains all extended equations of the form $x{:}\tau_j \rhd E'$, for each $j = 1 \ldots k$. Therefore, each $[\![\tau_j]\!]^{I_0}$ is partially initial in $\mathbf{C}$. Also, from $\mathcal{T} \not\vdash_s^{ext} E$, by (*add-emp*) one has $\mathcal{T} \not\vdash_s^{ext} E_0$, hence $I_0 \not\models E_0$. This shows that $I_0 \not\models E$. □

The proof of the completeness result for set-theoretic models rests on the following lemma, which implies that any extended theory is an intersection of principal ones:

**Lemma 5.28. Maximal extended theories are principal.** *Let $E$ be an extended equation and let $\mathcal{T}$ be a maximal extended theory such that $\mathcal{T} \not\vdash_s^{ext} E$. Then $\mathcal{T}$ is principal.*

*Proof.* Consider the following two hypothetical arguments:

1.: If there is some extended equation $x{:}\sigma \rhd E'$ that is *not* in $\mathcal{T}$, then, by maximality, $\mathcal{T} \cup \{x{:}\sigma \rhd E'\} \vdash_s^{ext} E$. Consider any derivation of $E$ from $\mathcal{T} \cup \{x{:}\sigma \rhd E'\}$. Alter this derivation by adding an emptiness assertion $e(\sigma)$ to each extended equation throughout. An inspection of the proof rules in Tables 5.2 and 5.3 shows that this alteration yields a valid derivation of $e(\sigma) \rhd \mathcal{T} \cup \{e(\sigma), x{:}\sigma \rhd E'\} \vdash_s^{ext} e(\sigma) \rhd E$, where $e(\sigma) \rhd \mathcal{T}$ denotes the set of equations $\{e(\sigma) \rhd E'' \mid E'' \in \mathcal{T}\}$. Applying the rules (*add-emp*) and (*empty*) at the leaves, one gets $\mathcal{T} \vdash_s^{ext} e(\sigma) \rhd E$.

2.: If there is some extended equation $e(\sigma) \rhd E'$ that is *not* in $\mathcal{T}$, then, by the same reasoning as in 1., $\mathcal{T} \vdash_s^{ext} x{:}\sigma \rhd E$.

Now observe that cases 1. and 2. cannot happen simultaneously, since otherwise $\mathcal{T} \vdash_s^{ext} E$ by (*cases*). It follows that $\mathcal{T}$ is principal. □

**Theorem 5.29. Soundness and Completeness for special ccc's.** *Let $\mathcal{CCC}_{spec}$ be the class of special ccc's. Then*

$$\mathcal{E} \models_{\mathcal{CCC}_{spec}} E \quad \textit{iff} \quad \mathcal{E} \models_{\mathcal{CCC}_{spec}}^{\textit{non-strict}} E \quad \textit{iff} \quad \mathcal{E} \vdash_s^{ext} E.$$

*Proof.* Soundness is an immediate consequence of the first part of Proposition 5.27. For completeness, assume $\mathcal{E} \not\vdash_s^{ext} E$. Let $\mathcal{T}$ be a maximal extended theory containing $\mathcal{E}$ such that $\mathcal{T} \not\vdash_s^{ext} E$. $\mathcal{T}$ is principal by Lemma 5.28. By the second part of Proposition 5.27, it follows that $\mathcal{T}$ is the extended theory of some strict interpretation $I : \underline{\lambda} \to \mathbf{C}$. Hence $I \models \mathcal{E}$ but $I \not\models E$, which implies $\mathcal{E} \not\models_{\mathcal{CCC}_{spec}} E$. □

Soundness and completeness for set-theoretic models now follows by applying the Henkin Representation Theorem 5.11. We write $\models_{\mathscr{S}}^{\textit{non-strict}}$ for semantic consequence for extended equations with respect to set-theoretic models.

**Theorem 5.30. Soundness and Completeness for non-strict interpretations in $\mathscr{S}$.**

$$\mathcal{E} \models_{\mathscr{S}}^{\textit{non-strict}} E \quad \textit{if and only if} \quad \mathcal{E} \vdash_s^{ext} E.$$

*Proof.* Soundness is a special case of Theorem 5.29. For completeness, suppose $\mathcal{E} \not\vdash_s^{ext} E$. By Theorem 5.29, there is a special ccc $\mathbf{C}$ and a strict interpretation $I_0 : \underline{\lambda} \to \mathbf{C}$ such that $I_0 \models \mathcal{E}$ but $I_0 \not\models E$. By Theorem 5.11, there is a Henkin embedding $H : \mathbf{C} \to \mathscr{S}$. Let $I = H \circ I_0 : \underline{\lambda} \to \mathscr{S}$. It follows from Remark 5.1.4 that $I$ validates the same extended equations as $I_0$. □

### 5.3.5 Kripke lambda models

By a ***Kripke lambda model***, we mean a a non-strict interpretation $I : \underline{\lambda} \to \mathscr{S}^P$ in a presheaf category over some poset $P$. We write $\mathcal{E} \models^{\textit{non-strict}}_{\textit{Kripke}} E$ for semantic consequence in the class of Kripke lambda models.

**Theorem 5.31. Soundness and Completeness for non-strict interpretations in $\mathscr{S}^P$.** *Each simply-typed lambda theory arises from some non-strict interpretation in a presheaf category $\mathscr{S}^P$ over some poset $P$. As a consequence,*

$$\mathcal{E} \models^{\textit{non-strict}}_{\textit{Kripke}} E \quad \textit{if and only if} \quad \mathcal{E} \vdash_s E.$$

*Proof.* Soundness is a special case of Proposition 5.24. Completeness is an immediate consequence of Theorem 5.14. □

### 5.3.6 A remark on the principal model property

The class of set-theoretic models with non-empty types and the class of Kripke lambda models each have the *principal model property*: any lambda theory that arises from the class actually arises as the theory of a *single* model. However, the principal model property does not hold for interpretations in $\mathscr{S}$. Indeed, among the extended theories, the ones that arise from a single model are the *principal* ones in the sense of Definition 5.26—but not all theories are principal.

The reason for the failure of the principal model property lies with the categorical properties of $\mathscr{S}$. Unlike the category of non-empty sets, the category of sets does not embed its own discrete powers. Notice that any discrete power $(\mathscr{S}^+)^I$ of the category of non-empty sets has enough points to be Henkin-embedded in $\mathscr{S}^+$ via the point functor $\Gamma = (1, -)$. As a consequence, a ccc $\mathbf{C}$ can be Henkin-embedded in $(\mathscr{S}^+)^I$ if and only if it can be Henkin-embedded in $\mathscr{S}^+$, and a lambda theory arises as the theory of a family of models with non-empty types if and only if it arises as the theory of a single such model. A similar property holds for the class of Kripke lambda models, because any discrete power of a presheaf category $\mathscr{S}^P$ is again of this form.

What the proofs of Theorems 5.29 and 5.30 really show about set-theoretic models is that any extended theory is the theory of some interpretation in a discrete power $\mathscr{S}^I$ of the category of sets. The proof is indirect, by first showing that any extended theory is an intersection of principal (namely, maximal) ones. In the process, the categorical meaning of the extended equations gets lost. Is it possible to give a more direct proof in the spirit of categorical logic, via a construction of a category directly from an extended theory? This would be the ultimate form of Theorem 5.30. Presumably such a proof would require a categorical characterization of those ccc's that can be Henkin-embedded in $\mathscr{S}^I$. Unfortunately, the characterization given in Corollary 5.13 is not very elegant, and a more satisfactory Henkin Representation Theorem for the class $\mathscr{S}^I$ is not known.

## 5.4 Henkin representations of *PL*-categories

### 5.4.1 *PL*-categories

Let $U : \mathcal{CCC} \to \mathcal{C}at$ and $|-| : \mathcal{CCC} \to \mathscr{S}$ be the forgetful functors that map a small ccc to its underlying category and to its set of objects, respectively.

**Definition.** A ***PL-category*** $\mathcal{B} = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta \rangle$ consists of

1. a small ***base category*** $\mathbf{B}$ with finite products and a distinguished object $\Omega$,
2. a contravariant ***fiber functor*** $F : \mathbf{B}^{op} \to \mathcal{CCC}$, together with a natural isomorphism

$$\gamma : (V, \Omega) \xrightarrow{\sim}_V |F_V|,$$

3. a natural transformation $\forall : U(F_{V\times\Omega}) \to_V U(F_V)$, together with a natural isomorphism

$$\eta : (F_{\pi_V} C, D)_{V\times\Omega} \xrightarrow{\sim}_{V,C,D} (C, \forall_V D)_V.$$

We sometimes write a *PL*-category as $\langle \mathbf{B}, F, \forall \rangle$ if the remaining parts of the structure are understood.

*Remarks.* We assume that the finite products of the base category are chosen. We use the letters $V, W, \ldots$ for objects of $\mathbf{B}$. The fiber functor $F$ maps an object $V$ to a cartesian-closed category $F_V$, called the ***fiber at V***. We also call $F_{\Omega^n}$ the ***n-fiber***, and in particular, $F_1$ is called the ***0-fiber***. We use the letters $C, D, \ldots$ for objects and $f, g, \ldots$ for morphisms of the fibers, and we denote hom-sets of $F_V$ by $(C, D)_V$. Each morphism of the base $\varphi : V \to W$ gives rise to a ccc-representation of fibers $F_\varphi : F_W \to F_V$. For an object $V \in \mathbf{B}$, let $\pi_V : V \times \Omega \to V$ be the first projection. The resulting functor $F_{\pi_V} : F_V \to F_{V\times\Omega}$ is called the ***dummy functor at V***, and we denote it by $\Delta_V$. Notice that $\Delta_V$, like $\pi_V$, is natural in $V$. Each $\Delta_V$ has a right adjoint $\forall_V : F_{V\times\Omega} \to F_V$. Notice that $\forall_V$ is not assumed to be a ccc-representation. Both the functor $\forall_V$ and the adjunction $\eta_V : \Delta_V \dashv \forall_V$ are assumed to be natural in $V$. The naturality of $\eta$ in $V$ means that for all $C \in F_W$ and $D \in F_{W\times\Omega}$ and for all $\varphi : V \to W$,

$$\begin{tikzcd}
(\Delta_W C, D)_{W\times\Omega} \arrow[r, "\eta_{W,C,D}", "\sim"'] \arrow[d, "F_{\varphi\times\Omega}"'] & (C, \forall_W D)_W \arrow[d, "F_\varphi"] \\
(\Delta_V C', D')_{V\times\Omega} \arrow[r, "\sim", "\eta_{V,C',D'}"'] & (C', \forall_V D')_V,
\end{tikzcd} \tag{5.4}$$

where $C' = F_\varphi C$ and $D' = F_{\varphi\times\Omega} D$. In the literature on hyperdoctrines and universal quantification [55, 15], the condition that $\forall$ is natural in $V$ is sometimes relaxed: one only requires that $F_\varphi \forall_W$ and $\forall_V F_{\varphi\times\Omega}$ are naturally isomorphic as functors $F_{W\times\Omega} \to F_V$. In this case, condition (5.4) is replaced by the so-called ***Beck-Chevalley condition***. In our setting, the Beck-Chevalley condition and (5.4) are equivalent.

The adjunction $\Delta_V \dashv \forall_V$ can be described concretely in terms of its co-unit $\Delta_V \forall_V D \xrightarrow{\theta_{V,D}} D$ by the following property: for every object $C \in F_V$ and every morphism $g : \Delta_V C \to D$, there exists a unique $h = \eta_V g : C \to \forall_V D$ such that

$$\begin{tikzcd}
\Delta_V \forall_V D \arrow[r, "\theta_{V,D}"] & D \\
\Delta_V C. \arrow[u, "\Delta_V h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

In analogy to product diagrams and exponential diagrams (see Section 1.1.6), we call a diagram of the form $\Delta_V E \xrightarrow{f} D$ with the above universal property a ***∀-diagram***. Condition (5.4) is equivalent to the requirement that $F_\varphi$ preserves $\forall$-diagrams, *i.e.* $F_\varphi \theta_{W,D} = \theta_{V,F_\varphi D}$ for all $\varphi : V \to W$ and $D \in F_{W\times\Omega}$.

**Definition.** Let $\mathcal{B} = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta \rangle$ and $\mathcal{B}' = \langle \mathbf{B}', \Omega', F', \gamma', \forall', \eta' \rangle$ be *PL*-categories. A ***PL-representation*** $\langle B, G \rangle : \mathcal{B} \to \mathcal{B}'$ is a finite product preserving functor $B : \mathbf{B} \to \mathbf{B}'$ together with a natural transformation $G : F \to F' \circ B$, such that $B\Omega = \Omega'$ and for all $V \in \mathbf{B}$, $C \in F_V$, and $D \in F_{V\times\Omega}$:

$$\begin{tikzcd}
\mathbf{B}(V,\Omega) \arrow[r, "\gamma_V", "\sim"'] \arrow[d, "B"'] & {|F_V|} \arrow[d, "{|G_V|}"] \\
\mathbf{B}'(BV,\Omega') \arrow[r, "\sim", "\gamma'_{BV}"'] & {|F'_{BV}|,}
\end{tikzcd}
\qquad
\begin{tikzcd}
F_{V\times\Omega} \arrow[r, "\forall_V"] \arrow[d, "G_{V\times\Omega}"'] & F_V \arrow[d, "G_V"] \\
F'_{BV\times\Omega'} \arrow[r, "\forall'_{BV}"'] & F'_{BV},
\end{tikzcd}
\qquad
\begin{tikzcd}
(\Delta_V C, D)_{V\times\Omega} \arrow[r, "\eta_{V,C,D}", "\sim"'] \arrow[d, "G_{V\times\Omega}"'] & (C, \forall_V D)_V \arrow[d, "G_V"] \\
(\Delta'_{BV} C', D')_{BV\times\Omega'} \arrow[r, "\sim", "\eta'_{BV,C',D'}"'] & (C', \forall_{BV} D')_{BV},
\end{tikzcd}$$

where $C' = G_V C$ and $D' = G_{V\times\Omega} D$. The condition that $G$ preserves $\eta$ can be equivalently expressed in terms of $\forall$-diagrams by requiring $G_{V\times\Omega}(\theta_{V,D}) = \theta'_{BV,D'}$.

Notice that $G$ is a natural transformation of functors $\mathbf{B} \to \mathcal{CCC}$; in particular, each $G_V : F_V \to F'_{BV}$ is a ccc-representation.

Small *PL*-categories and *PL*-representations form a category, which we denote by $\mathcal{PL}$.

We will now consider a notion of congruence relation on a *PL*-category. We are only concerned about congruences on the morphisms of the fibers, and not on the morphisms or objects of the base.

**Definition.** A ***PL-congruence*** $\sim$ on a *PL*-category $\mathcal{B}$ is given by a family of equivalence relations on the hom-set of the fibers, *i.e.*, an equivalence relation $\sim_{V,C,D}$ on $(C,D)_V$ for each $V \in \mathbf{B}$ and $C, D \in F_V$, such that for each $V$, $\sim_V$ is a ccc-congruence on $F_V$ (see Definition 5.1), and in addition:

$$\frac{f \sim_{V\times\Omega,\Delta_V C,D} f'}{\eta_V f \sim_{V,C,\forall_V D} \eta_V f'}$$

If $\langle B, G\rangle : \mathcal{B} \to \mathcal{B}'$ is a *PL*-representation, then its ***kernel*** is a *PL*-congruence on $\mathcal{B}$, defined by $f \sim_{V,C,D} g$ iff $G_V f = G_V g$, for all $f, g \in (C,D)_V$.

Conversely, let $\sim$ be a *PL*-congruence on $\mathcal{B}$. One can define the ***quotient*** $\mathcal{B}/{\sim}$ by taking the quotient $(C,D)_V/{\sim_{V,C,D}}$ at each hom-set of the fibers; one checks that this is a well-defined *PL*-category with the same base category as $\mathcal{B}$.

### 5.4.2 Henkin-*PL*-representations

A ***pre-structure*** $\mathcal{P} = \langle \mathbf{P}, M\rangle$ consists of a base category $\mathbf{P}$ with finite products and a contravariant functor $M : \mathbf{P}^{op} \to \mathcal{CCC}$. For any pair of objects $V, W \in \mathbf{P}$, we consider the first projection $\pi_{V,W} : V \times W \to V$, and the associated functor $M_{\pi_{V,W}} : M_V \to M_{V\times W}$, which we again call the ***dummy functor***, and which we denote by $\Delta_{V,W}$. We say that a diagram $\Delta_{V,W} C \xrightarrow{f} D$ in $M_{V\times W}$ is a ***partial*** $\forall$***-diagram*** if for every object $C' \in M_V$ and every morphism $g : \Delta_{V,W} C' \to D$, there exists *at most* one $h : C' \to C$ such that

$$\begin{tikzcd}
\Delta_{V,W} C \arrow[r, "f"] & D \\
\Delta_{V,W} C'. \arrow[u, "\Delta_{V,W} h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

A ***Henkin natural transformation*** between functors $F, G : \mathbf{B} \to \mathcal{CCC}$ is a natural transformation $H : UF \to UG$ such that for each $V \in \mathbf{B}$, $H_V$ is a Henkin representation of ccc's.

Analogous to Henkin representations of cartesian-closed categories, we can now define Henkin representations of *PL*-categories:

**Definition.** Let $\mathcal{B} = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta\rangle$ be a *PL*-category and $\mathcal{P} = \langle \mathbf{P}, M\rangle$ be a pre-structure. A ***Henkin-PL-representation***$\langle B, H\rangle : \mathcal{B} \to \mathcal{P}$ is a finite product preserving functor $B : \mathbf{B} \to \mathbf{P}$ together with a Henkin natural transformation $H : UF \to UM \circ B$, such that for all $V \in \mathbf{B}$ and $D \in F_{V\times\Omega}$,

$$\Delta_{BV,B\Omega}(H_V \forall_V D) = H_{V\times\Omega}\Delta_V \forall_V D \xrightarrow{H_{V\times\Omega}\theta_{V,D}} H_{V\times\Omega} D$$

is a partial $\forall$-diagram. Notice that, by naturality of $H$,

$$\begin{tikzcd}
F_V \arrow[r, "H_V"] \arrow[d, "\Delta_V = F_{\pi_V}"'] & M_{BV} \arrow[d, "\Delta_{BV,B\Omega} = M_{B\pi_V}"] \\
F_{V\times\Omega} \arrow[r, "H_{V\times\Omega}"'] & M_{BV\times B\Omega}.
\end{tikzcd}$$

**Lemma 5.32.** *The kernel of a Henkin-PL-representation $\langle B, H\rangle : \mathcal{B} \to \mathcal{P}$, defined for all $f, g \in (C,D)_V$ by $f \sim_{V,C,D} g$ iff $H_V f = H_V g$, is a PL-congruence on $\mathcal{B}$.*

*Proof.* For each $V \in \mathbf{B}$, $H_V : F_V \to M_{BV}$ is a Henkin representation of ccc's, and hence $\sim_V$ is a ccc-congruence on $F_V$ by Lemma 5.2. It remains to be seen that for $C \in FV$ and $D \in FV \times \Omega$, $f \sim_{V\times\Omega,\Delta_V C,D} f'$ implies $\eta_V f \sim_{V,C,\forall_V D} \eta_V f'$. Suppose $f, f' \in (\Delta_V C, D)_{V\times\Omega}$ with $H_{V\times\Omega} f = H_{V\times\Omega} f'$. One has

$$\begin{tikzcd}
\Delta_V \forall_V D \arrow[r, "\theta_{V,D}"] & D & & \Delta_{BV,B\Omega} H_V \forall_V D \arrow[r, equal] & H_{V\times\Omega}\Delta_V\forall_V D \arrow[r, "H_{V\times\Omega}\theta_{V,D}"] & H_{V\times\Omega} D \\
\Delta_V C \arrow[u, "\Delta_V \eta_V f"] \arrow[ur, "f"'] & & \Rightarrow & \Delta_{BV,B\Omega} H_V C \arrow[u, "\Delta_{BV,B\Omega} H_V \eta_V f"] \arrow[r, equal] & H_{V\times\Omega}\Delta_V C. \arrow[u, "H_{V\times\Omega}\Delta_V \eta_V f"'] \arrow[ur, "H_{V\times\Omega} f"'] &
\end{tikzcd}$$

The top row is a partial $\forall$-diagram, hence the arrow $H_V \eta_V f$ is uniquely determined by $H_{V\times\Omega} f$. Since there is an identical diagram for $f'$, and since $H_{V\times\Omega} f = H_{V\times\Omega} f'$ by assumption, one has $H_V \eta_V f = H_V \eta_V f'$.□

**Definition.** We say that a Henkin-*PL*-representation $\langle B, H\rangle$ is a ***Henkin-PL-embedding*** if its kernel is the trivial congruence, *i.e.* if $H_V$ is a Henkin embedding for each $V$. Notice that we do not require $B$ to be an embedding of the base; it seems unnecessary to do so since we are only concerned with equality in the fibers.

### 5.4.3 Standard structures

Consider a cartesian-closed category $\mathbf{D}$. Let $\widetilde{\mathbf{D}}$ be the pre-structure $\langle \mathscr{S}, M\rangle$, where the base category is the category of sets, and the functor $M : \mathscr{S} \to \mathcal{CCC}$ maps a set $X$ to $\mathbf{D}^X$, the $X$-fold power of $\mathbf{D}$. We call this pre-structure the ***standard structure over*** $\mathbf{D}$.

For a ccc, we considered the point functor $\Gamma : \mathbf{C} \to \mathscr{S}$. We now consider an analogue to this functor for *PL*-categories. Consider a *PL*-category $\mathcal{B} = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta\rangle$, together with a functor $H^0 : F_1 \to \mathbf{D}$. We define $\Gamma_{H^0} = \langle B, H\rangle$, where $B : \mathbf{B} \to \mathscr{S}$ is the point functor of the base category, mapping $V$ to the hom-set $\mathbf{B}(1, V)$, and $H : F_V \to_V M_{BV} = \mathbf{D}^{BV}$ is the natural transformation defined on objects by $H_V C(x) = H^0(F_x C)$, where $C \in F_V$ and $x \in BV = (1, V)$. The following proposition gives a sufficient condition for $\langle B, H\rangle$ to be Henkin-*PL*-embedding.

**Proposition 5.33.** *The pair $\langle B, H\rangle$ is a Henkin-PL-embedding $\mathcal{B} \to \widetilde{\mathbf{D}}$ if the following hold:*

1. *$H^0$ is a Henkin embedding,*
2. *$H^0$ preserves monic cones, and*
3. *the functors $F_x : F_V \to F_1$, where $x : 1 \to V$, form a collective embedding for each $V \in \mathbf{B}$.*

*Proof.* Clearly $B$ preserves products and $H$ is natural. What remains to be shown is that each $H_V$ is a Henkin embedding, and that the condition on $\forall$-diagrams is satisfied. First, notice that $H_V : \mathcal{F}_V \to \mathbf{C}^{BV}$ factors as

$$\begin{tikzcd}
F_V \arrow[d, "(F_x)_{x\in BV}"'] \arrow[dr, "H_V"] & \\
F_1^{BV} \arrow[r, "(H^0)^{BV}"'] & \mathbf{C}^{BV}.
\end{tikzcd}$$

Clearly, $(F_x)_{x\in BV}$ is a ccc representation; by assumption 3, it is also an embedding. Assumption 1 implies that $(H^0)^{BV}$ is a Henkin embedding, hence $H_V$ is a Henkin embedding for every $V$. Now suppose $\Delta_V \forall_V D \xrightarrow{\theta_{V,D}} D$ is a $\forall$-diagram in $\mathcal{B}$. We need to show that

$$\Delta_{BV,B\Omega}(H_V \forall_V D) = H_{V\times\Omega}\Delta_V \forall_V D \xrightarrow{H_{V\times\Omega}\theta_{V,D}} H_{V\times\Omega} D$$

is a partial $\forall$-diagram. Unraveling the definitions, this amounts to showing that for each $y \in BV$, the collection of morphisms

$$(H^0 \forall_1 C \xrightarrow{H^0 F_z \theta_{1,C}} H^0 F_z C)_{z:1\to\Omega}$$

is collectively monic, where $C = F_{y\times\Omega}D$. Since $H^0$ preserves monic cones by assumption 2, it suffices to show that for every $C \in F_\Omega$, the family

$$(\forall_1 C \xrightarrow{F_z\theta_{1,C}} F_z C)_{z:1\to\Omega}$$

is collectively monic. Let $f \neq g : A \to \forall_1 C$. Then $\eta^{-1}f \neq \eta^{-1}g : \Delta_1 A \to C$. By assumption 3, there is $z : 1 \in \Omega$ with $F_z\eta^{-1}f \neq F_z\eta^{-1}g$, *i.e.* $F_z(\theta_{1,C} \circ \Delta_1 f) \neq F_z(\theta_{1,C} \circ \Delta_1 g)$. But $F_z\Delta_1 = \mathsf{id}_{F_1}$, hence $(F_z\theta_{1,C}) \circ f \neq (F_z\theta_{1,C}) \circ g$, which proves the claim. □

### 5.4.4 Freely adjoining arrows to the base of a *PL*-category

Given a *PL*-category $\mathcal{B} = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta\rangle$, we may freely adjoin an arrow $x : 1 \to U$ to the base as follows: Let $\mathbf{B}'$ be the Kleisli category of the comonad $T(V) = U \times V$, *i.e.* $\mathbf{B}'$ has the same objects as $\mathbf{B}$ and $\mathbf{B}'(V, W) = \mathbf{B}(U \times V, W)$ (compare Section 5.1.3). Define $F' : \mathbf{B}'^{op} \to \mathcal{CCC}$ by $F'_V = F_{U\times V}$; this is natural in $V$. Define $\Omega' = \Omega$, $\gamma'_V = \gamma_{U\times V}$, $\forall'_V = \forall_{U\times V}$, and $\eta'_{V,C,D} = \eta_{U\times V,C,D}$. It is trivial to check that $\mathcal{B}' = \langle \mathbf{B}', \Omega', F', \gamma', \forall', \eta'\rangle$ is indeed a *PL*-category; for instance $\forall'_V = \forall_{U\times V} : F_{U\times V\times\Omega} \to F_{U\times V}$ is indeed right adjoint to $\Delta'_V = \Delta_{U\times V} : F_{U\times V} \to F_{U\times V\times\Omega}$. Let $\jmath = \langle B_0, G_0\rangle : \mathcal{B} \to \mathcal{B}'$ be the natural *PL*-representation, defined by $B_0\varphi = \varphi \circ \pi'$ and $(G_0)_V = \Delta_V : F_V \to F'_V = F_{U\times V}$. Let $x \in \mathbf{B}'(1, U)$ be $\mathsf{id} \in \mathbf{B}(U, U)$. We write $\mathcal{B}'$ as $\mathcal{B}[1 \xrightarrow{x} U]$, which is justified by its universal property:

**Proposition 5.34.** *$\mathcal{B}[1 \xrightarrow{x} U]$ has the following universal property: for any PL-representation $\langle B, G\rangle : \mathcal{B} \to \mathcal{D}$ and any arrow $\psi : 1 \to BU$ in $\mathcal{D}$, there is a unique PL-representation $\langle \hat{B}, \hat{G}\rangle : \mathcal{B}[1 \xrightarrow{x} U] \to \mathcal{D}$ such that*

$$\begin{tikzcd} \mathcal{B} \arrow[r, "\jmath"] \arrow[dr, "\langle B,G\rangle"'] & \mathcal{B}[1 \xrightarrow{x} U] \arrow[d, "\langle \hat{B},\hat{G}\rangle"] \\ & \mathcal{D} \end{tikzcd}$$

*and $\hat{B}x = \psi$.*

*Proof.* Let $B'$ map an object $V$ to $BV$ and a morphism $\varphi \in \mathbf{B}'(V, W) = \mathbf{B}(U \times V, W)$ to $BV \xrightarrow{\psi\times\mathsf{id}} BU \times BV \xrightarrow{B\varphi} BW$. Define $G'_V$ by

$$\begin{tikzcd} F'_V \arrow[r, "G'_V"'] \arrow[d, equal] & D_{BV}. \\ F_{U\times V} \arrow[r, "G_{U\times V}"'] & D_{BU\times BV} \arrow[u, "D_{\psi\times\mathsf{id}}"'] \end{tikzcd}$$

It is readily checked that this is the unique *PL*-representation with the desired properties. □

**Lemma 5.35.** *The canonical PL-representation $\jmath : \mathcal{B} \to \mathcal{B}[1 \xrightarrow{x} \Omega]$ is a PL-embedding.*

*Proof.* In $\mathcal{B}$, there is always a point $\psi : 1 \in \Omega$, for instance $\gamma^{-1}1$. The unique extension $\mathcal{B}[1 \xrightarrow{x} \Omega] \to \mathcal{B}$ of the identity that sends $x$ to $\psi$ is a left inverse to $\jmath$. □

**Proposition 5.36.** *Any PL-category $\mathcal{B}$ can be PL-embedded in a PL-category $\mathcal{B}'$ such that the functors $F'_z : F'_{\Omega^n} \to F'_1$, where $z : 1 \to \Omega^n$, form a collective embedding for each $n$.*

*Proof.* To $\mathcal{B}$, adjoin countably many arrows $1 \to \Omega$ by constructing a sequence $\mathcal{B} = \mathcal{B}_0 \xrightarrow{\jmath_0} \mathcal{B}_1 \xrightarrow{\jmath_1} \cdots$ of *PL*-categories, where $\mathcal{B}_{i+1} = \mathcal{B}_i[1 \xrightarrow{x_i} \Omega]$ and $\jmath_i$ is the canonical embedding. Notice that the $n$-fiber of $\mathcal{B}_i$ is the $n+i$-fiber of $\mathcal{B}$. Let $\mathcal{B}'$ be the colimit of this sequence, *i.e.* the objects of the base are the same as for each $\mathcal{B}_i$, and the hom-sets of the base and objects and hom-sets of the fibers are constructed as the directed unions of the respective parts of the $\mathcal{B}_i$. It is easily checked that $\mathcal{B}'$ is a *PL*-category with a canonical *PL*-embedding

$\jmath : \mathcal{B} \to \mathcal{B}'$. To show that the functors $F'_z : F'_{\Omega^n} \to F'_1$ form a collective embedding, it suffices to show that for every $V$, the functors $F'_{V\times x} : F'_{V\times\Omega} \to F'_V$ form a collective embedding, where $x : 1 \to \Omega$. Consider two morphisms $f \neq g : C \to D$ in $F'_{V\times\Omega}$. Then $C, D, f$, and $g$ already exist in some $\mathcal{B}_i$. Consider $x_i : 1 \to \Omega$ in $\mathcal{B}_{i+1}$. Writing $F^i$ for the fiber functor of $\mathcal{B}_i$, one has

$$\begin{tikzcd} F^i_{V\times\Omega} \arrow[d, hook, "\jmath_i"'] \arrow[dr, "\mathrm{id}"] & \\ F^{i+1}_{V\times\Omega} \arrow[r, "F^{i+1}_{V\times x_i}"'] & F^{i+1}_V, \end{tikzcd}$$

hence $F^{i+1}_{V\times x_i}\jmath_i f \neq F^{i+1}_{V\times x_i}\jmath_i g$, which implies $F'_{V\times x_i} f \neq F'V \times x_i g$. $\square$

### 5.4.5 Henkin-*PL*-representation theorems

#### Henkin-*PL*-representations in $\widetilde{\mathscr{S}^+}$

**Lemma 5.37.** *If the 0-fiber of a PL-category has the property that $A \twoheadrightarrow 1$ for every object $A$, then the same is true for any $n$-fiber.*

*Proof.* Let $\forall_n : F_{\Omega^n} \to F_1$ be the right adjoint of the canonical functor $\Delta_n = F_{\bigcirc} : F_1 \to F_{\Omega^n}$. Let $C, D$ be objects of $F_{\Omega^n}$ and consider $f, g : 1 \to D$ such that

$$\begin{array}{rr} & C \longrightarrow 1 \overset{f}{\underset{g}{\underset{\div}{\rightrightarrows}}} D \\ \Rightarrow & \forall_n C \longrightarrow\!\!\!\!\!\rightarrow 1 = \forall_n 1 \overset{\forall_n f}{\underset{\forall_n g}{\underset{\div}{\rightrightarrows}}} \forall_n D \\ \Rightarrow & \Delta_n\forall_n 1 \longrightarrow 1 \overset{f}{\underset{g}{\underset{\div}{\rightrightarrows}}} D. \end{array}$$

But both $\Delta_n$ and $\forall_n$ preserve terminators, hence $\Delta_n\forall_n 1 = 1$ and $f = g$. $\square$

**Theorem 5.38.** *Let $\mathcal{B}$ be a PL-category whose base is generated by $\Omega$. Then $\mathcal{B}$ can be Henkin-PL-embedded in $\widetilde{\mathscr{S}^+}$, the standard structure over $\mathscr{S}^+$, if and only if for every object $A$ of the 0-fiber, the morphism $A \twoheadrightarrow 1$ is epic.*

*Proof.* $\Rightarrow$: Trivial, because a Henkin-*PL*-embedding $\mathcal{B} \to \widetilde{\mathscr{S}^+}$ gives rise to a Henkin embedding $F_1 \to \mathscr{S}^+$, and embeddings reflect epics.

$\Leftarrow$: By Lemma 5.37, $C \twoheadrightarrow 1$ holds for all objects of all fibers. By Proposition 5.36, $\mathcal{B}$ can be embedded in a *PL*-category $\mathcal{B}'$ such that the functors $F'_z : F'_V \to F'_1$ form a collective embedding, for every $V \in \mathbf{B}$. Note that in the sequence of *PL*-categories $(\mathcal{B}_i)_i$ constructed in the proof of Proposition 5.36, the 0-fiber of $\mathcal{B}_i$ is the $n$-fiber of $\mathcal{B}$. Hence, $C \twoheadrightarrow 1$ holds for all objects of $F^i_1$, and therefore for all objects of $F'_1$ as well. By Theorem 5.9, there is a Henkin embedding $H^0 : F'_1 \xrightarrow{F_{\mathcal{A}}} \mathbf{D} \xrightarrow{\Gamma} \mathscr{S}^+$. By Lemma 5.6, $F_{\mathcal{A}}$ preserves monic cones, and so does the point functor $\Gamma$. Therefore, Proposition 5.33 is applicable and we obtain a Henkin-*PL*-embedding $\mathcal{B}' \to \widetilde{\mathscr{S}^+}$. $\square$

#### Henkin-*PL*-representations in $\widetilde{\mathscr{S}^P}$

**Theorem 5.39.** *Any PL-category $\mathcal{B}$ can be Henkin-PL-embedded in $\widetilde{\mathscr{S}^P}$, for some poset $P$.*

Table 5.4: Typing rules for the polymorphic lambda calculus

$$(\textit{typeapp}) \quad \frac{\Gamma \rhd M : \forall\alpha.\tau}{\Gamma \rhd M\sigma : \tau[\sigma/\alpha]}$$

$$(\textit{typeabs}) \quad \frac{\Gamma \rhd M : \tau \qquad \alpha \notin \mathrm{FTV}(\Gamma)}{\Gamma \rhd \Lambda\alpha.M : \forall\alpha.\tau}$$

Table 5.5: Equational rules for the polymorphic lambda calculus

$$(\textit{cong}_6) \quad \frac{\Gamma \rhd M = M' : \forall\alpha.\tau}{\Gamma \rhd M\sigma = M'\sigma : \tau} \qquad (B) \quad \frac{}{\Gamma \rhd (\Lambda\alpha.M)\sigma = M[\sigma/\alpha] : \tau}$$

$$(\textit{cong}_7) \quad \frac{\Gamma \rhd M = M' : \tau \qquad \alpha \notin \mathrm{FTV}(\Gamma)}{\Gamma \rhd \Lambda\alpha.M = \Lambda\alpha.M' : \forall\alpha.\tau} \qquad (H) \quad \frac{\alpha \notin \mathrm{FTV}(M)}{\Gamma \rhd \Lambda\alpha.(M\alpha) = M : \forall\alpha.\tau}$$

*Proof.* By Proposition 5.36, $\mathcal{B}$ can be embedded in a *PL*-category $\mathcal{B}'$ such that the functors $F'_z : F'_V \to F'_1$ form a collective embedding, for every $V \in \mathbf{B}$. By Theorem 5.14, there is a Henkin embedding $H^0 : F'_1 \to \mathscr{S}^P$, for some poset $P$, such that $H^0$ preserves monic cones. With Proposition 5.33, one obtains a Henkin-*PL*-embedding of $\mathcal{B}'$ in $\widetilde{\mathscr{S}^P}$. □

# 5.5 The interpretation of the polymorphic lambda calculus

## 5.5.1 The polymorphic lambda calculus

The polymorphic lambda calculus was independently introduced by Girard [22] and Reynolds [51]. Here, we describe a version of the second order lambda calculus with surjective pairing and a unit type.

Let $\mathcal{TV}$ be an infinite set of ***type variables*** $\alpha, \beta, \ldots$, and let $TC$ be a set of ***type constants*** $t, u, \ldots$. ***Polymorphic types*** $\sigma, \tau, \ldots$ are given by the grammar:

$$\sigma ::= \alpha \mid t \mid 1 \mid \sigma \times \tau \mid \sigma \to \tau \mid \forall\alpha.\sigma.$$

Let $\mathcal{V}$ be an infinite set of ***individual variables*** $x, y, \ldots$. For each closed type $\sigma$, let $C_\sigma$ be a set of ***individual constants*** $c^\sigma, d^\sigma, \ldots$. The collection $\langle TC, (C_\sigma)_\sigma \rangle$ is also called a ***polymorphic signature***. ***Raw polymorphic lambda terms*** $M, N, \ldots$ are given by the grammar:

$$M ::= x \mid c^\sigma \mid * \mid \langle M, N \rangle \mid \pi_1 M \mid \pi_2 M \mid MN \mid \lambda x{:}\sigma.M \mid M\sigma \mid \Lambda\alpha.M.$$

As usual, the individual variable $x$ is bound in the term $\lambda x{:}\sigma.M$. Moreover, the type variable $\alpha$ is bound in the term $\Lambda\alpha.M$ and in the type $\forall\alpha.\sigma$. All other occurrences of variables are free, and we write FV$(M)$ for the free individual variables and FTV$(M)$ for the free type variables of a term $M$, as well as FTV$(\sigma)$ for the free type variables of a type $\sigma$. We identify types, as well as raw terms, up to renaming of bound variables. There are three kinds of substitution: substitution of types in types $\tau[\sigma/\alpha]$, substitution of types in terms $M[\sigma/\alpha]$, and substitution of terms in terms $M[N/x]$.

A ***type assignment*** $\Gamma = x_1{:}\sigma_1, x_2{:}\sigma_2, \ldots, x_m{:}\sigma_m$ is defined as for the simply-typed lambda calculus. We write FTV$(\Gamma)$ = FTV$(\sigma_1) \cup \ldots \cup$ FTV$(\sigma_m)$. The ***valid typing judgments*** $\Gamma \rhd M : \sigma$ of the polymorphic lambda calculus are derived by the rules in Tables 5.1 and 5.4. An ***equation*** is again an expression of the form $\Gamma \rhd M = N : \sigma$, where $\Gamma \rhd M : \sigma$ and $\Gamma \rhd N : \sigma$ are valid typing judgments. If $E$ is an equation and $\mathcal{E}$ is a set of equations, we write $\mathcal{E} \vdash_p E$ if $E$ can be derived from $\mathcal{E}$ by the rules for the simply-typed lambda calculus in Table 5.2, together with the one for the polymorphic lambda calculus in Table 5.5. $\mathcal{E}$ is called a ***theory*** if it is closed under derivability. The smallest theory of the polymorphic lambda calculus (for a fixed polymorphic signature) is denoted by $\underline{PL}$.

### 5.5.2 Strict interpretation in a *PL*-category

Fix a polymorphic signature. A ***(strict) interpretation*** $I$ of the polymorphic lambda calculus in a *PL*-category $\mathcal{B}$, which we schematically write as $I : \underline{PL} \to \mathcal{B}$, consists of an interpretation of types and an interpretation of typing judgments, both relative to a sequence $\bar{\alpha} = \alpha_1, \ldots$
$alpha_n$ of type variables. A type $\sigma$ with FTV$(\sigma) \subseteq \{\bar{\alpha}\}$ is interpreted as an object $[\![\sigma]\!]^I_{\bar{\alpha}}$ of $F_{\Omega^n}$. A valid typing judgment $\Gamma \rhd M : \tau$ with FTV$(\Gamma, M, \tau) \subseteq \{\bar{\alpha}\}$ is interpreted as a morphism $[\![\Gamma \rhd M : \tau]\!]^I_{\bar{\alpha}}$ of $F_{\Omega^n}$. Like for the simply-typed lambda calculus, an interpretation $I$ is uniquely determined by its values on type constants and individual constants.

Let $I : TC \to |F_1|$ be an interpretation of type constants as objects of the 0-fiber. This extends uniquely to an interpretation $[\![\sigma]\!]^I_{\alpha_1,\ldots,\alpha_n}$ of every type. Recall that $\gamma$ is the natural isomorphism $(V, \Omega) \xrightarrow{\sim}_V |F_V|$, and that $F_{\bigcirc} : F_1 \to F_{\Omega^n}$ is the ccc-representation induced by the unique morphism $\bigcirc : \Omega^n \to 1$. We assume that bound variables are renamed as necessary.

$$\begin{array}{rcl}
[\![\alpha_i]\!]^I_{\bar{\alpha}} & = & \gamma_{\Omega^n}\pi_i, \text{ where } \pi_i \in (\Omega^n, \Omega) \text{ is the } i\text{th projection} \\
[\![t]\!]^I_{\bar{\alpha}} & = & F_{\bigcirc} I(t) \\
[\![1]\!]^I_{\bar{\alpha}} & = & 1 \\
[\![\sigma \times \tau]\!]^I_{\bar{\alpha}} & = & [\![\sigma]\!]^I_{\bar{\alpha}} \times [\![\tau]\!]^I_{\bar{\alpha}} \\
[\![\sigma \to \tau]\!]^I_{\bar{\alpha}} & = & ([\![\tau]\!]^I_{\bar{\alpha}})^{[\![\sigma]\!]^I_{\bar{\alpha}}} \\
[\![\forall\alpha'.\sigma]\!]^I_{\bar{\alpha}} & = & \forall_{\Omega^n}[\![\sigma]\!]^I_{\bar{\alpha},\alpha'}
\end{array}$$

If $C$ is any object of $F_V$, then it corresponds, via $\gamma$, to a morphism of the base $\varphi : V \to \Omega$. The morphism $\langle \mathsf{id}_V, \varphi\rangle : V \to V \times \Omega$ gives rise to a functor $F_{\langle \mathsf{id}_V, \varphi\rangle} : F_{V\times\Omega} \to F_V$, which we denote by $[C]_V$. We call this functor the ***substitution functor***.

**Lemma 5.40.** *The following are properties of the interpretation of polymorphic types:*

1. **Permutation of Type Variables.** The interpretation is independent of the ordering of the free type variables, or of the addition of dummy variables, in the following sense: If $s : \{1, \ldots, k\} \to \{1, \ldots, n\}$ is injective and FTV$(\tau) \subseteq \{\alpha_{s1}, \ldots, \alpha_{sk}\}$, then

$$[\![\tau]\!]^I_{\alpha_1,\ldots,\alpha_n} = F_{\langle \pi_{s1},\ldots,\pi_{sk}\rangle}[\![\tau]\!]^I_{\alpha_{s1},\ldots,\alpha_{sk}}.$$

   In particular, if $\alpha' \notin$ FTV$(\sigma)$, then $[\![\sigma]\!]^I_{\bar{\alpha},\alpha'} = \Delta_{\Omega^n}[\![\sigma]\!]^I_{\bar{\alpha}}$.

2. **Type Substitution.** For all types $\sigma$ and $\tau$ with FTV$(\sigma) \subseteq \{\bar{\alpha}\}$ and FTV$(\tau) \subseteq \{\bar{\alpha}, \alpha'\}$,

$$[\![\tau[\sigma/\alpha']]\!]^I_{\bar{\alpha}} = [[\![\sigma]\!]^I_{\bar{\alpha}}]_{\Omega^n}[\![\tau]\!]^I_{\bar{\alpha},\alpha'}.$$ □

Notice that $[C]_V \circ \Delta_V = F_{\langle \mathsf{id}_V, \varphi\rangle} \circ F_{\pi_V} = \mathsf{id}_{F_V}$. Therefore, applying $[C]_V$ to the co-unit $\theta_{V,D} : \Delta_V \forall_V D \to D$ yields a natural transformation inst$_{V,C,D} : \forall_V D \to_D [C]_V D$, which will be useful for the interpretation of type application.

If $\Gamma = x_1{:}\sigma_1, \ldots, x_m{:}\sigma_m$ is a type assignment, we write $[\![\Gamma]\!]^I_{\bar{\alpha}} = [\![\sigma_1]\!]^I_{\bar{\alpha}} \times \cdots \times [\![\sigma_m]\!]^I_{\bar{\alpha}}$. Let $I_\sigma : C_\sigma \to (1, [\![\sigma]\!]^I)_1$ be an interpretation of term constants as morphisms of the 0-fiber, for each closed type $\sigma$. This extends uniquely to an interpretation $[\![\Gamma \rhd M : \tau]\!]^I_{\bar\alpha}$ of valid typing judgments:

$$
\begin{array}{rcl}
[\![\Gamma \rhd x_j : \sigma_j]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\pi_j} [\![\sigma_j]\!]^I_{\bar\alpha}, \text{ the } j\text{th projection} \\
[\![\Gamma \rhd c^\sigma : \sigma]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\bigcirc} 1 \xrightarrow{F_\bigcirc I_\sigma(c^\sigma)} [\![\sigma]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd * : 1]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\bigcirc} 1 = [\![1]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd \langle M, N\rangle : \sigma \times \tau]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\langle [\![\Gamma \rhd M:\sigma]\!]^I_{\bar\alpha}, [\![\Gamma \rhd N:\tau]\!]^I_{\bar\alpha}\rangle} [\![\sigma]\!]^I_{\bar\alpha} \times [\![\tau]\!]^I_{\bar\alpha} = [\![\sigma \times \tau]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd \pi_1 M : \sigma]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{[\![\Gamma \rhd M:\sigma\times\tau]\!]^I_{\bar\alpha}} [\![\sigma]\!]^I_{\bar\alpha} \times [\![\tau]\!]^I_{\bar\alpha} \xrightarrow{\pi} [\![\sigma]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd \pi_2 M : \tau]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{[\![\Gamma \rhd M:\sigma\times\tau]\!]^I_{\bar\alpha}} [\![\sigma]\!]^I_{\bar\alpha} \times [\![\tau]\!]^I_{\bar\alpha} \xrightarrow{\pi'} [\![\sigma]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd MN : \tau]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\langle [\![\Gamma \rhd M:\sigma\to\tau]\!]^I_{\bar\alpha}, [\![\Gamma \rhd N:\sigma]\!]^I_{\bar\alpha}\rangle} ([\![\tau]\!]^I_{\bar\alpha})^{[\![\sigma]\!]^I_{\bar\alpha}} \times [\![\sigma]\!]^I_{\bar\alpha} \xrightarrow{\varepsilon} [\![\tau]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd \lambda x{:}\sigma.M : \sigma \to \tau]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{([\![\Gamma, x:\sigma \rhd M:\tau]\!]^I_{\bar\alpha})^\star} ([\![\tau]\!]^I_{\bar\alpha})^{[\![\sigma]\!]^I_{\bar\alpha}} = [\![\sigma \to \tau]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd M\sigma : \tau[\sigma/\alpha']]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{[\![\Gamma \rhd M:\forall\alpha'.\tau]\!]^I_{\bar\alpha}} \forall_{\Omega^n} [\![\tau]\!]^I_{\bar\alpha,\alpha'} \xrightarrow{\text{inst}} [[\![\sigma]\!]^I_{\bar\alpha}]_{\Omega^n} [\![\tau]\!]^I_{\bar\alpha,\alpha'} = [\![\tau[\sigma/\alpha']]\!]^I_{\bar\alpha} \\
[\![\Gamma \rhd \Lambda\alpha'.M : \forall\alpha'.\tau]\!]^I_{\bar\alpha} & = & [\![\Gamma]\!]^I_{\bar\alpha} \xrightarrow{\eta_{\Omega^n}([\![M]\!]^I_{\bar\alpha,\alpha'})} \forall_{\Omega^n} [\![\tau]\!]^I_{\bar\alpha,\alpha'} = [\![\forall\alpha'.\tau]\!]^I_{\bar\alpha}
\end{array}
$$

**Lemma 5.41.** *The interpretation of the polymorphic lambda calculus in a PL-category, defined inductively as above, has the expected properties:*

1. **Permutation of Type Variables.** If $s : \{1, \ldots, k\} \to \{1, \ldots, n\}$ is injective and the free type variables $\text{FTV}(\Gamma, M, \tau) \subseteq \{\alpha_{s1}, \ldots, \alpha_{sk}\}$, then
$$[\![\Gamma \rhd M : \tau]\!]^I_{\alpha_1,\ldots,\alpha_n} = F_{\langle \pi_{s1},\ldots,\pi_{sk}\rangle} [\![\Gamma \rhd M : \tau]\!]^I_{\alpha_{s1},\ldots,\alpha_{sk}}.$$

2. **Permutation of Individual Variables.** If $s : \{1, \ldots, l\} \to \{1, \ldots, m\}$ is injective and $\text{FV}(M) \subseteq \{x_{s1}, \ldots, x_{sl}\}$, and if $\Gamma' = x_1{:}\sigma_1, \ldots x_m{:}\sigma_m$ and $\Gamma = x_{s1}{:}\sigma_{s1}, \ldots x_{sl}{:}\sigma_{sl}$ then
$$
\begin{tikzcd}
{[\![\Gamma']\!]^I_{\bar\alpha}} \arrow[rr, "{[\![\Gamma' \rhd M:\tau]\!]^I_{\bar\alpha}}"] \arrow[dr, "{\langle \pi_{s1},\ldots,\pi_{sl}\rangle}"'] & & {[\![\tau]\!]^I_{\bar\alpha}.} \\
& {[\![\Gamma]\!]^I_{\bar\alpha}} \arrow[ur, "{[\![\Gamma \rhd M:\tau]\!]^I_{\bar\alpha}}"'] &
\end{tikzcd}
$$

3. **Type Substitution.** Whenever $\text{FTV}(\Gamma, M, \tau) \subseteq \{\bar\alpha, \alpha'\}$ and $\text{FTV}(\sigma) \subseteq \{\bar\alpha\}$, then
$$[\![(\Gamma \rhd M : \tau)[\sigma/\alpha']]\!]^I_{\bar\alpha} = [[\![\sigma]\!]^I_{\bar\alpha}]_{\Omega^n} [\![\Gamma \rhd M : \tau]\!]^I_{\bar\alpha,\alpha'}.$$

4. **Term Substitution.** Let $\Gamma = x_1{:}\sigma_1, \ldots x_m{:}\sigma_m$ and $\Gamma' = y_1{:}\rho_1, \ldots y_l{:}\rho_l$, and suppose $\Gamma \rhd M : \tau$ and $\Gamma' \rhd N_j : \sigma_j$ for $j = 1, \ldots, m$. Then
$$
\begin{tikzcd}
{[\![\Gamma']\!]^I_{\bar\alpha}} \arrow[rr, "{[\![\Gamma' \rhd M[\bar N/\bar x]:\tau]\!]^I_{\bar\alpha}}"] \arrow[dr, "{\langle [\![\Gamma' \rhd N_1:\sigma_1]\!]^I_{\bar\alpha},\ldots,[\![\Gamma' \rhd N_n:\sigma_n]\!]^I_{\bar\alpha}\rangle}"'] & & {[\![\tau]\!]^I_{\bar\alpha}.} \\
& {[\![\Gamma]\!]^I_{\bar\alpha}} \arrow[ur, "{[\![\Gamma \rhd M:\tau]\!]^I_{\bar\alpha}}"'] &
\end{tikzcd}
$$
□

As usual, we say that an interpretation $I$ ***satisfies*** an equation $\Gamma \rhd M = N : \tau$, in symbols $I \models \Gamma \rhd M = N : \tau$, if $[\![\Gamma \rhd M : \tau]\!]^I_{\bar\alpha} = [\![\Gamma \rhd N : \tau]\!]^I_{\bar\alpha}$. This notion is independent of $\bar\alpha$, as long as $\text{FTV}(\Gamma, M, N, \tau) \subseteq \bar\alpha$. We write $\models_{\mathcal{M}}$ for semantic consequence in a class $\mathcal{M}$ of *PL*-categories, meaning $\mathcal{E} \models_{\mathcal{M}} E$ if all interpretations in a *PL*-category in $\mathcal{M}$ that satisfy $\mathcal{E}$ also satisfy $E$.

**Proposition 5.42 (Seely [56]). Soundness.**

$$\mathcal{E} \vdash_p E \quad \textit{implies} \quad \mathcal{E} \models_{\mathcal{PL}} E.$$

□

If $\mathcal{T}$ is a theory and $I : \underline{PL} \to \mathcal{B}$ is an interpretation such that $I \models \mathcal{T}$, then we also write $I : \mathcal{T} \to \mathcal{B}$. An interpretation can be post-composed with a *PL*-representation in an evident way: $\mathcal{T} \xrightarrow{I} \mathcal{B} \xrightarrow{G} \mathcal{B}'$ is the interpretation $J$ defined by $[\![\sigma]\!]^J_{\bar{\alpha}} = G[\![\sigma]\!]^I_{\bar{\alpha}}$ and $[\![\Gamma \rhd M : \tau]\!]^J_{\bar{\alpha}} = G[\![\Gamma \rhd M : \tau]\!]^I_{\bar{\alpha}}$.

### 5.5.3 The *PL*-category associated to a theory

From a theory $\mathcal{T}$ over a polymorphic signature, one constructs a *PL*-category $\mathfrak{F}_{PL}(\mathcal{T}) = \langle \mathbf{B}, \Omega, F, \gamma, \forall, \eta \rangle$ as follows: Fix an enumeration $\alpha_1, \alpha_2, \ldots$ of type variables, and fix an individual variable $x$. The base $\mathbf{B}$ has countably many objects, which we denote $1, \Omega, \Omega^2, \ldots$; the hom-set $(\Omega^n, \Omega^k)$ is given by all $k$-tuples $\langle \sigma_1, \ldots, \sigma_k \rangle$ of polymorphic types with $\mathrm{FTV}(\sigma_1, \ldots, \sigma_k) \subseteq \{\alpha_1, \ldots, \alpha_n\}$. Composition is given by substitution:

$$\Omega^n \xrightarrow{\langle \sigma_1, \ldots, \sigma_k \rangle} \Omega^k \xrightarrow{\langle \tau_1, \ldots, \tau_l \rangle} \Omega^l \quad = \quad \langle \tau_1[\sigma_i/\alpha_i], \ldots, \tau_l[\sigma_i/\alpha_i] \rangle,$$

where $\tau[\sigma_i/\alpha_i]$ denotes the simultaneous substitution of $\sigma_1, \ldots, \sigma_k$ for $\alpha_1, \ldots, \alpha_k$. The identity at $\Omega^n$ is $\langle \alpha_1, \ldots, \alpha_n \rangle$. One checks that the base has finite products.

The objects of the $n$-fiber are defined via $|F_{\Omega^n}| = (\Omega^n, \Omega)$, *i.e.* they are polymorphic types $\sigma$ with $\mathrm{FTV}(\sigma) \subseteq \{\alpha_1, \ldots, \alpha_n\}$. The morphisms $f_M \in (\sigma, \tau)_{\Omega^n}$ of the $n$-fiber are named by terms $M$ such that $x{:}\sigma \rhd M : \tau$ is a valid typing judgment. Two terms $M$ and $N$ name the same morphism if $\mathcal{T} \vdash_p x{:}\sigma \rhd M = N : \tau$. Just as in the construction of the ccc associated to a simply-typed theory (see Section 5.2.3), one checks that $F$ is indeed cartesian-closed. The action of $F$ on morphisms $\varphi : \Omega^n \to \Omega^k$ of the base is as follows: $F_{\langle \sigma_1, \ldots, \sigma_k \rangle}$ maps objects $\tau$ to $\tau[\sigma_i/\alpha_i]$ and morphisms $f_M$ to $f_{M[\sigma_i/\alpha_i]}$. This defines a ccc-representation $F_{\Omega^k} \to F_{\Omega^n}$. Notice that $\Delta_{\Omega^n}\sigma = \sigma$. The right adjoint $\forall_{\Omega^n}$ acts on objects as $\forall_{\Omega^n}\sigma = \forall\alpha_{n+1}.\sigma$. The adjunction $\eta_{\Omega^n,\sigma,\tau} : (\sigma, \tau)_{\Omega^{n+1}} \to (\sigma, \forall\alpha_{n+1}.\tau)_{\Omega^n}$ is given by $\eta_{\Omega^n,\sigma,\tau} f_M = f_{\Lambda\alpha_{n+1}.M}$.

**Proposition 5.43 (Seely [56]).** *The above construction yields a well-defined PL-category $\mathfrak{F}_{PL}(\mathcal{T})$. There is a canonical strict interpretation $I_0 : \mathcal{T} \to \mathfrak{F}_{PL}(\mathcal{T})$ with $[\![\sigma]\!]^{I_0}_{\bar{\alpha}} = \sigma$ and $[\![x{:}\sigma \rhd M : \tau]\!]^{I_0}_{\bar{\alpha}} = f_M : \sigma \to \tau$. Moreover, $I_0$ has the following universal property: For any strict interpretation $J : \mathcal{T} \to \mathcal{B}$, there is a unique PL-representation $G : \mathfrak{F}_{PL}(\mathcal{T}) \to \mathcal{B}$ such that*

$$\begin{tikzcd} \mathcal{T} \arrow[d, "I_0"'] \arrow[dr, "J"] & \\ \mathfrak{F}_{PL}(\mathcal{T}) \arrow[r, "G"'] & \mathcal{B}. \end{tikzcd}$$

□

**Corollary 5.44. Completeness of *PL*-categories for the polymorphic lambda calculus.** *Each theory $\mathcal{T}$ of the polymorphic lambda calculus arises as the theory of some strict interpretation in a PL-category. Hence,*

$$\mathcal{E} \models_{\mathcal{PL}} E \quad \textit{implies} \quad \mathcal{E} \vdash_p E.$$

□

### 5.5.4 The non-strict interpretation of the polymorphic lambda calculus

A ***non-strict interpretation*** of the polymorphic lambda calculus in a pre-structure $\mathcal{P}$, denoted $I : \underline{PL} \to \mathcal{P}$, is a Henkin-*PL*-representation $H : \mathfrak{F}_{PL}(\underline{PL}) \to \mathcal{P}$. One defines $[\![\sigma]\!]^I_{\bar{\alpha}} = H[\![\sigma]\!]^{I_0}_{\bar{\alpha}}$ and $[\![\Gamma \rhd M : \tau]\!]^I_{\bar{\alpha}} = H[\![\Gamma \rhd M : \tau]\!]^{I_0}_{\bar{\alpha}}$. The notations $I \models E$, as well as $\mathcal{E} \models^{\textit{non-strict}}_{\mathcal{M}} E$, have their usual meanings. The following Soundness Theorem is a consequence of Lemma 5.32. Again, completeness is evident, since the class of non-strict interpretations includes the class of strict ones.

**Proposition 5.45. Soundness of the Non-Strict Interpretation.**

$$\mathcal{E} \vdash_p E \quad \textit{implies} \quad \mathcal{E} \models^{\textit{non-strict}}_{\mathcal{PL}} E.$$

□

A non-strict interpretation of the polymorphic lambda calculus in a standard structure $\widetilde{\mathbf{D}}$ is called a ***standard model***. The case $\mathbf{D} = \mathscr{S}^+$ gives rise to set-theoretic models with non-empty types which are closely related to the environment-style models that were described by Bruce and Meyer [10]. The case $\mathbf{D} = \mathscr{S}^P$ gives rise to polymorphic Kripke models. Finally, the case $\mathbf{D} = \mathscr{S}$ gives rise to set-theoretic models of polymorphism with possibly empty types. We will leave the discussion of the latter class of models for elsewhere.

## 5.6 From Henkin-*PL*-representation theorems to polymorphic completeness theorems

### 5.6.1 Set-theoretic models with non-empty types

A ***set-theoretic model of polymorphism with non-empty types*** is a non-strict interpretation in the standard structure $\widetilde{\mathscr{S}^+}$. Write $\models^{\textit{non-strict}}_{\widetilde{\mathscr{S}^+}}$ for semantic consequence with respect to this class of models.

**Theorem 5.46. Soundness and Completeness for set-theoretic models of polymorphism with non-empty types.** *The rule (non-empty) is sound for set-theoretic models of polymorphism with non-empty types. Moreover, any theory that is closed under (non-empty) arises from such an interpretation. Consequently,*

$$\mathcal{E} \models^{\textit{non-strict}}_{\widetilde{\mathscr{S}^+}} E \quad \textit{if and only if} \quad \mathcal{E} \vdash^{\textit{non-empty}}_p E.$$

*Proof.* Soundness follows from Lemma 5.37 and the remarks in Section 5.3.2. For completeness, let $\mathcal{T}$ be a theory that is closed under (*non-empty*). Let $I_0 : \mathcal{T} \to \mathfrak{F}_{PL}(\mathcal{T})$ be the canonical interpretation. Because of the rule (*non-empty*), one has $C \twoheadrightarrow 1$ for all objects of the base, hence, by Theorem 5.38, there is a Henkin-*PL*-embedding $H : \mathfrak{F}_{PL}(\mathcal{T}) \to \widetilde{\mathscr{S}^+}$. Then the interpretation $I = H \circ I_0$ satisfies exactly $\mathcal{T}$. □

### 5.6.2 Polymorphic Kripke models

A ***polymorphic Kripke model*** is a non-strict interpretation in a standard structure $\widetilde{\mathscr{S}^P}$ where $P$ is a poset. Semantic consequence for this class of models is denoted by $\models^{\textit{non-strict}}_{\textit{Kripke}}$.

**Theorem 5.47. Soundness and Completeness for polymorphic Kripke models.** *Each polymorphic lambda theory is the theory of some polymorphic Kripke model. Therefore,*

$$\mathcal{E} \models^{\textit{non-strict}}_{\textit{Kripke}} E \quad \textit{if and only if} \quad \mathcal{E} \vdash_p E.$$

*Proof.* This is a direct consequence of Theorem 5.39. □

# Chapter 6

# First-Order Axioms for Asynchrony

The distinction between *synchronous* and *asynchronous* communication is a relevant issue in the design and analysis of distributed and concurrent networks. Intuitively, communication is said to be synchronous if messages are sent and received simultaneously, via a 'handshake' or 'rendez-vous' of sender and receiver. It is asynchronous if messages travel through a communication medium with possible delay, such that the sender cannot be certain if or when a message has been received.

Asynchronous communication is often studied in the framework of concurrent process paradigms such as the asynchronous $\pi$-calculus, which was originally introduced by Honda and Tokoro [26], and which was independently discovered by Boudol [9] as a result of his work with Berry on chemical abstract machines [8]. Another such asynchronous paradigm is the join calculus, which was recently proposed by Fournet and Gonthier as a calculus of mobile agents in distributed networks with locality and failure [17, 18].

In this chapter, we study properties of asynchronous communication in general, not with regard to any particular process calculus. We give a general-purpose, mathematically rigorous definition of asynchrony, and then we show that this notion can be equivalently characterized by a small number of first-order axioms. We model processes by labeled transition systems with input and output, a framework that is sufficiently general to fit concurrent process paradigms such as the $\pi$-calculus or the join calculus, as well as data flow models and other such formalisms. These transition systems are similar to Lynch and Stark's input/output automata [35], but our treatment is more category-theoretic and close in spirit to Abramsky's interaction categories [1, 2].

Various properties of asynchrony have been exploited in different contexts by many authors. For instance, Lynch and Stark [35] postulate a form of *input receptivity* for their automata. Palamidessi [45] makes use of a certain *confluence* property to prove that the expressive power of the asynchronous $\pi$-calculus is strictly less than that of the synchronous $\pi$-calculus. Axioms similar to the ones that are presented here have been postulated by Shields [59] and Bednarczyk [6] for a notion of asynchronous labeled transition systems, but without the input/output distinction which is central to the present approach.

The main novelty of our approach is that the axioms are not postulated *a priori*, but derived from more primitive notions. We define asynchrony in elementary terms: an agent is asynchronous if its input and/or output is filtered through a communication medium, such as a buffer or a queue, possibly with feedback. We then show that our first- and second-order axioms precisely capture each of these notions. This characterization justifies the axioms *a posteriori*. As a testbed and for illustration, we apply these axioms to an asynchronous version of Milner's CCS, and to the core join calculus.

## 6.1 An elementary definition of asynchrony

If $R$ is a binary relation, we write $R^{-1}$ for the inverse relation and $R^*$ for the reflexive, transitive closure of $R$. We also write $\leftarrow$ for $\rightarrow^{-1}$, etc. The binary identity relation on a set is denoted $\Delta$. The composition of two binary relations $R$ and $Q$ is written $R \circ Q$ or simply $RQ$, *i.e.* $xRQz$ if there exists $y$ such that $xRyQz$.

We write $xR$ for the unary relation $\{y|xRy\}$, and similarly $Ry$ for $\{x|xRy\}$. The disjoint union of sets $X$ and $Y$ is denoted by $X + Y$.

### 6.1.1 Labeled transition systems and bisimulation

To keep this chapter self-contained, we summarize the standard definitions for labeled transition systems and weak and strong bisimulation.

**Definition.** A ***labeled transition system (LTS)*** is a tuple $\mathbf{S} = \langle S, A, \rightarrow_{\mathbf{S}}, s_0\rangle$, where $S$ is a set of ***states***, $A$ is a set of ***actions***, $\rightarrow_{\mathbf{S}} \subseteq S \times A \times S$ is a ***transition relation*** and $s_0 \in S$ is an ***initial state***. We call $A$ the ***type*** of $\mathbf{S}$, and we write $\mathbf{S}\colon A$.

We often omit the subscript on $\rightarrow_{\mathbf{S}}$, and we write $|\mathbf{S}|$ for the set of states $S$. For $\alpha \in A$, we regard $\xrightarrow{\alpha}$ as a binary relation on $|\mathbf{S}|$ via $s \xrightarrow{\alpha} s'$ iff $\langle s, \alpha, s'\rangle \in \rightarrow$. The definitions of strong and weak bisimulation rely on the following principle of co-inductive definition:

**Principle 6.1.** *Let $X$ be a set and $P$ a property of subsets of $X$. If $P(R)$ is defined by clauses of the form $\mathcal{F}_i(R) \subseteq \mathcal{G}_i(R)$, where $\mathcal{F}_i$ and $\mathcal{G}_i$ are set-valued, monotone operators, and if $\mathcal{F}_i$ preserves unions, then $P$ is closed under unions. In particular, there is a maximal $R_{max} \subseteq X$ with $P(R_{max})$.*

*Proof.* Since $\mathcal{F}_i$ preserves unions, it has a right adjoint $\mathcal{F}'_i$. Then $P(R) \iff \forall i.\mathcal{F}_i(R) \subseteq \mathcal{G}_i(R) \iff R \subseteq \bigcap_i \mathcal{F}'_i\mathcal{G}_i(R)$. Hence $P$ is the set of pre-fixpoints of a monotone operator and therefore closed under least upper bounds. Let $R_{max} = \bigcup\{R \mid P(R)\}$. ☐

**Definition.** Let $\mathbf{S}$ and $\mathbf{T}$ be LTSs of type $A$. A binary relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a ***strong bisimulation*** if for all $\alpha \in A$, $R\xrightarrow{\alpha} \subseteq \xrightarrow{\alpha}R$ and $R^{-1}\xrightarrow{\alpha} \subseteq \xrightarrow{\alpha}R^{-1}$. In diagrams:

$$\begin{tikzcd} s \arrow[r, phantom, "R"] & t \arrow[d, "\alpha"] \\ & t' \end{tikzcd} \Rightarrow \exists s'.\ \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & t \arrow[d, "\alpha"] \\ s' \arrow[r, phantom, "R"] & t' \end{tikzcd} \qquad\text{and}\qquad \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & t \\ s' & \end{tikzcd} \Rightarrow \exists t'.\ \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & t \arrow[d, "\alpha"] \\ s' \arrow[r, phantom, "R"] & t' \end{tikzcd}$$

Next, we consider LTSs with a distinguished action $\tau \in A$, called the ***silent*** or the ***unobservable*** action. Let $\overset{\tau}{\Rightarrow}$ be the relation $\xrightarrow{\tau}^*$. For $a \in A \setminus \tau$, let $\overset{a}{\Rightarrow}$ be the relation $\xrightarrow{\tau}^*\xrightarrow{a}\xrightarrow{\tau}^*$. A binary relation $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a ***weak bisimulation*** if for all $\alpha \in A$, $R\xrightarrow{\alpha} \subseteq \overset{\alpha}{\Rightarrow}R$ and $R^{-1}\xrightarrow{\alpha} \subseteq \overset{\alpha}{\Rightarrow}R^{-1}$. In diagrams:

$$\begin{tikzcd} s \arrow[r, phantom, "R"] & t \arrow[d, "\alpha"] \\ & t' \end{tikzcd} \Rightarrow \exists s'.\ \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, Rightarrow, "\alpha"'] & t \arrow[d, "\alpha"] \\ s' \arrow[r, phantom, "R"] & t' \end{tikzcd} \qquad\text{and}\qquad \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & t \\ s' & \end{tikzcd} \Rightarrow \exists t'.\ \begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & t \arrow[d, Rightarrow, "\alpha"] \\ s' \arrow[r, phantom, "R"] & t' \end{tikzcd}$$

By Principle 6.1, it follows that there is a maximal strong bisimulation, which we denote by $\sim$, and a maximal weak bisimulation, which we denote by $\approx$. We say that $s \in |\mathbf{S}|$ and $t \in |\mathbf{T}|$ are ***strongly (weakly) bisimilar*** if $s \sim t$ ($s \approx t$). Finally, $\mathbf{S}$ and $\mathbf{T}$ are said to be strongly (weakly) bisimilar if $s_0 \sim t_0$ ($s_0 \approx t_0$).

*Remark.* Note that $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ is a weak bisimulation if and only if for all $\alpha \in A$, $R\overset{\alpha}{\Rightarrow} \subseteq \overset{\alpha}{\Rightarrow}R$ and $R^{-1}\overset{\alpha}{\Rightarrow} \subseteq \overset{\alpha}{\Rightarrow}R^{-1}$.

If $\mathbf{S}, \mathbf{T}, \mathbf{U}$ are labeled transition systems and if $R \subseteq |\mathbf{S}| \times |\mathbf{T}|$ and $Q \subseteq |\mathbf{T}| \times |\mathbf{U}|$ are weak (respectively, strong) bisimulations, then so are the identity relation $\Delta \subseteq |\mathbf{S}| \times |\mathbf{S}|$, the inverse $R^{-1} \subseteq |\mathbf{T}| \times |\mathbf{S}|$, and the composition $R \circ Q \subseteq |\mathbf{S}| \times |\mathbf{U}|$. Hence weak and strong bisimilarity each define a *global* equivalence relation on the class of all states of all possible labeled transition systems.

In particular, $\sim$ and $\approx$, as binary relations on an LTS $\mathbf{S}$, are equivalence relations. We denote the respective equivalence classes of a state $s$ by $[s]_\sim$ and $[s]_\approx$. On the quotient $\mathbf{S}/\!\sim$, we define transitions $[s]_\sim \xrightarrow{a} [t]_\sim$ iff $s \xrightarrow{a}\sim t$, making it into a well-defined transition system. Similarly, on $\mathbf{S}/\!\approx$, we define $[s]_\approx \xrightarrow{a} [t]_\approx$ iff $s \xrightarrow{a}\approx t$. For all $s \in \mathbf{S}$, one has $s \sim [s]_\sim$ and $s \approx [s]_\approx$, and hence $\mathbf{S} \sim (\mathbf{S}/\!\sim)$ and $\mathbf{S} \approx (\mathbf{S}/\!\approx)$. We say that $\mathbf{S}$ is ***$\sim$-reduced*** if $\mathbf{S} = \mathbf{S}/\!\sim$, and ***$\approx$-reduced*** if $\mathbf{S} = \mathbf{S}/\!\approx$.

### 6.1.2 Input, output and sequential composition

So far we have distinguished only one action: the silent action $\tau$. We will now add further structure to the set of actions by distinguishing input and output actions. Let *in* and *out* be constants. For any sets $X$ and $Y$, define a set of ***input actions*** $\mathit{In}\,X := \{\mathit{in}\} \times X$, and a set of ***output actions*** $\mathit{Out}\,Y := \{\mathit{out}\} \times Y$. Note that $\mathit{In}\,X$ and $\mathit{Out}\,Y$ are disjoint. We will write input and output actions as $\mathit{in}\,x$ and $\mathit{out}\,x$ instead of $\langle \mathit{in}, x\rangle$ and $\langle \mathit{out}, x\rangle$, respectively. Let $B$ be a set whose elements are not of the form $\mathit{in}\,x$, $\mathit{out}\,y$ or $\tau$. The elements of $\mathbf{B} + \{\tau\}$ are called ***internal actions***.

**Definition.** We define $X \rightarrow_B Y$ to be the set $\mathit{In}\,X + \mathit{Out}\,Y + B + \{\tau\}$. A labeled transition system $\mathbf{S}$ of type $X \rightarrow_B Y$ is called an ***LTS***, or simply an ***agent***. If $B$ is empty, we will omit the subscript in $X \rightarrow_B Y$.

The traditional CCS notation is "$x$" for input actions and "$\bar{x}$" for output actions. We use $\mathit{in}\,x$ and $\mathit{out}\,x$ instead to emphasize the distinction between a message $\mathit{in}\,x$ and its content $x$.

Our labeled transition systems with input and output are similar to the input/output automata of Lynch and Stark [35]. However, we consider a notion of sequential composition that is more in the spirit of Abramsky's interaction categories [1, 2]. Given two agents $\mathbf{S}\colon X \rightarrow_B Y$ and $\mathbf{T}\colon Y \rightarrow_B Z$, we define $\mathbf{S};\mathbf{T}\colon X \rightarrow_B Z$ by feeding the output of $\mathbf{S}$ into the input of $\mathbf{T}$. This is a special case of parallel composition and hiding. Notice that this notion of sequential composition is different from the one of CSP or ACP, where $\mathbf{T}$ cannot start execution until $\mathbf{S}$ is finished.

Sequential composition, together with certain other agent constructors that we will investigate in Section 6.3.1, can be used to build arbitrary networks of agents.

**Definition 6.2.** Let $\mathbf{S}\colon X \rightarrow_B Y$ and $\mathbf{T}\colon Y \rightarrow_B Z$ be agents with respective initial states $s_0$ and $t_0$. The ***sequential composition*** $\mathbf{S};\mathbf{T}$ is of type $X \rightarrow_B Z$. It has states $|\mathbf{S}| \times |\mathbf{T}|$ and initial state $\langle s_0, t_0\rangle$. The transitions are given by the following rules:

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s' \qquad \alpha \text{ not output}}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t' \qquad \alpha \text{ not input}}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S};\mathbf{T}} \langle s,t'\rangle} \qquad \frac{s \xrightarrow{\mathit{out}\,y}_{\mathbf{S}} s' \qquad t \xrightarrow{\mathit{in}\,y}_{\mathbf{T}} t'}{\langle s,t\rangle \xrightarrow{\tau}_{\mathbf{S};\mathbf{T}} \langle s',t'\rangle}$$

*Example 6.3.* For any set $X$, define an agent $\mathcal{I}_X$ of type $X \rightarrow X$ with states $X + \{\bot\}$, initial state $\bot$ and transitions $\bot \xrightarrow{\mathit{in}\,x} x$ and $x \xrightarrow{\mathit{out}\,x} \bot$, for all $x \in X$. $\mathcal{I}_X$ acts as a buffer of capacity one: A possible sequence of transitions is

$$\bot \xrightarrow{\mathit{in}\,x} x \xrightarrow{\mathit{out}\,x} \bot \xrightarrow{\mathit{in}\,y} y \xrightarrow{\mathit{out}\,y} \bot \xrightarrow{\mathit{in}\,z} z \xrightarrow{\mathit{out}\,z} \bot \ldots$$

Let $X = \{x\}$. Then $\mathcal{I}_X$ and $\mathcal{I}_X;\mathcal{I}_X$ are the following agents:

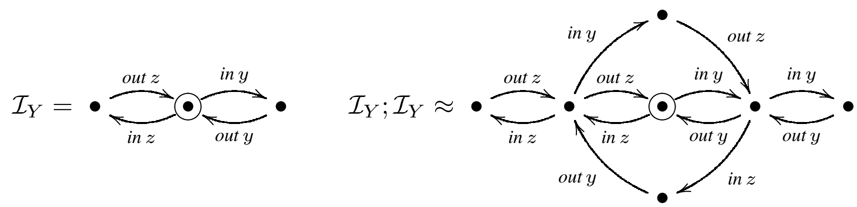
$$\mathcal{I}_X = \begin{tikzcd} \bot \arrow[r, bend left, "\mathit{in}\,x"] & x \arrow[l, bend left, "\mathit{out}\,x"] \end{tikzcd} \qquad \mathcal{I}_X;\mathcal{I}_X = \begin{tikzcd} \langle\bot,\bot\rangle \arrow[d, "\mathit{in}\,x"'] & \langle\bot,x\rangle \arrow[l, "\mathit{out}\,x"'] \arrow[d, "\mathit{in}\,x"] \\ \langle x,\bot\rangle \arrow[ur, "\tau"] & \langle x,x\rangle \arrow[l, "\mathit{out}\,x"] \end{tikzcd}$$

Here the initial state of each agent is circled. When representing agents in diagrams like these, it is often convenient to omit the names of the states, and to identify weakly bisimilar states. With that convention, we write:

$$\mathcal{I}_X = \begin{tikzcd} \odot \arrow[r, bend left, "\mathit{in}\,x"] & \bullet \arrow[l, bend left, "\mathit{out}\,x"] \end{tikzcd} \qquad \mathcal{I}_X;\mathcal{I}_X \approx \begin{tikzcd} \odot \arrow[r, bend left, "\mathit{in}\,x"] & \bullet \arrow[l, bend left, "\mathit{out}\,x"] \arrow[r, bend left, "\mathit{in}\,x"] & \bullet \arrow[l, bend left, "\mathit{out}\,x"] \end{tikzcd}$$

Note that $\mathcal{I}_X;\mathcal{I}_X$ is a queue of capacity 2. Let $Y = \{y, z\}$. The following diagrams represent $\mathcal{I}_Y$ and

$\mathcal{I}_Y;\mathcal{I}_Y$:




Again, $\mathcal{I}_Y;\mathcal{I}_Y$ is a queue of capacity 2. Notice that it is first-in, first-out.

Two LTSs $\mathbf{S}$ and $\mathbf{T}$ of type $A$ are ***isomorphic*** if there is a bijection between $|\mathbf{S}|$ and $|\mathbf{T}|$ preserving $\to$ and initial states.

**Lemma 6.4.** *1. Sequential Composition of labeled transition systems is associative up to isomorphism.*

*2. The following hold for the composition* $\mathbf{S};\mathbf{T}$*:*

$$\frac{s \stackrel{\alpha}{\Rightarrow}_{\mathbf{S}} s' \qquad \alpha \textit{ not output}}{\langle s,t\rangle \stackrel{\alpha}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \stackrel{\alpha}{\Rightarrow}_{\mathbf{T}} t' \qquad \alpha \textit{ not input}}{\langle s,t\rangle \stackrel{\alpha}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s,t'\rangle} \qquad \frac{s \stackrel{out\,y}{\Longrightarrow}_{\mathbf{S}} s' \qquad t \stackrel{in\,y}{\Longrightarrow}_{\mathbf{T}} t'}{\langle s,t\rangle \stackrel{\tau}{\Rightarrow}_{\mathbf{S};\mathbf{T}} \langle s',t'\rangle}$$

*3. Sequential Composition of agents respects both weak and strong bisimulation, i.e.*

$$\frac{\mathbf{S}_1 \approx \mathbf{S}_2 \qquad \mathbf{T}_1 \approx \mathbf{T}_2}{\mathbf{S}_1;\mathbf{T}_1 \approx \mathbf{S}_2;\mathbf{T}_2} \qquad \text{and} \qquad \frac{\mathbf{S}_1 \sim \mathbf{S}_2 \qquad \mathbf{T}_1 \sim \mathbf{T}_2}{\mathbf{S}_1;\mathbf{T}_1 \sim \mathbf{S}_2;\mathbf{T}_2}$$

*Proof.* 1. It is easy to check that $\langle\langle s,t\rangle,u\rangle \xrightarrow{\alpha} \langle\langle s',t'\rangle,u'\rangle$ if and only if $\langle s,\langle t,u\rangle\rangle \xrightarrow{\alpha} \langle s',\langle t',u'\rangle\rangle$.

2. The first two statements are trivial from Definition 6.2. For the third one, assume $s \xrightarrow{\tau}{}^{*} s_1 \xrightarrow{out\,y} s_2 \xrightarrow{\tau}{}^{*} s'$ and $t \xrightarrow{\tau}{}^{*} t_1 \xrightarrow{in\,y} t_2 \xrightarrow{\tau}{}^{*} t'$. Then $\langle s,t\rangle \xrightarrow{\tau}{}^{*} \langle s_1,t\rangle \xrightarrow{\tau}{}^{*} \langle s_1,t_1\rangle \xrightarrow{\tau} \langle s_2,t_2\rangle \xrightarrow{\tau}{}^{*} \langle s',t_2\rangle \xrightarrow{\tau}{}^{*} \langle s',t'\rangle$.

3. Let $\mathbf{S}_1,\mathbf{S}_2\colon X \to_B Y$ and $\mathbf{T}_1,\mathbf{T}_2\colon Y \to_B Z$. Suppose $Q \subseteq |\mathbf{S}_1| \times |\mathbf{S}_2|$ and $R \subseteq |\mathbf{T}_1| \times |\mathbf{T}_2|$ are weak bisimulations. We show that $Q \times R = \{\langle\langle s_1,t_1\rangle,\langle s_2,t_2\rangle\rangle \mid s_1 Q s_2 \text{ and } t_1 R t_2\} \subseteq |\mathbf{S}_1;\mathbf{T}_1| \times |\mathbf{S}_2;\mathbf{T}_2|$ is a weak bisimulation. It suffices without loss of generality to show one of the two directions. Suppose

$$\begin{array}{l} \langle s_1,t_1\rangle \; Q \times R \; \langle s_2,t_2\rangle \\ {\scriptstyle\alpha}\big\downarrow \\ \langle s_1',t_1'\rangle \end{array}$$

for some $\alpha \in X \to_B Z$. There are three cases, depending on which of the three rules in Definition 6.2 was used to derive $\langle s_1,t_1\rangle \xrightarrow{\alpha} \langle s_1',t_1'\rangle$:

**Case 1:** $s_1 \xrightarrow{\alpha} s_1'$, $t_1 = t_1'$ and $\alpha$ is not output: By $Q$ there is $s_2'$ such that $s_2 \stackrel{\alpha}{\Rightarrow} s_2'$ and $s_1' Q s_2'$. Let $t_2' = t_2$.
**Case 2:** $t_1 \xrightarrow{\alpha} t_1'$, $s_1 = s_1'$ and $\alpha$ is not input: By $R$ there is $t_2'$ such that $t_2 \stackrel{\alpha}{\Rightarrow} t_2'$ and $t_1' R t_2'$. Let $s_2' = s_2$.
**Case 3:** $s_1 \xrightarrow{out\,y} s_1'$, $t_1 \xrightarrow{in\,y} t_1'$ and $\alpha = \tau$: By $Q$ and $R$, there are $s_2'$ and $t_2'$ such that $s_2 \stackrel{out\,y}{\Longrightarrow} s_2'$, $s_1' Q s_2'$, $t_2 \stackrel{in\,y}{\Longrightarrow} t_2'$ and $t_1' R t_2'$.

In each case, by 2.,

$$\begin{array}{ccc} \langle s_1,t_1\rangle & Q \times R & \langle s_2,t_2\rangle \\ {\scriptstyle\tau}\big\downarrow & & \big\Downarrow{\scriptstyle\alpha} \\ \langle s_1',t_1'\rangle & Q \times R & \langle s_2',t_2'\rangle. \end{array}$$

For strong bisimulation, the proof is similar. $\square$

Unfortunately, agents do not form a category under sequential composition: there are no identity morphisms. In Section 6.1.4, we will introduce two categories of agents, one of which has unbounded buffers as its identity morphisms, and the other one queues.

### 6.1.3 Buffers and queues

For any set $X$, let $X^*$ be the free monoid and $X^{**}$ the free commutative monoid generated by $X$. The elements of $X^*$ are finite sequences. The empty sequence is denoted by $\epsilon$. The elements of $X^{**}$ are finite multisets. The empty multiset is denoted by $\emptyset$. We define the following agents of type $X \to_B X$:

1. The ***buffer*** $\mathcal{B}_X$ has states $X^{**}$, initial state $\emptyset$, and transitions $w \xrightarrow{\mathit{in}\,x} wx$ and $xw \xrightarrow{\mathit{out}\,x} w$, for all $w \in X^{**}$ and $x \in X$.

2. The ***queue*** $\mathcal{Q}_X$ has states $X^*$, initial state $\epsilon$, and transitions $w \xrightarrow{\mathit{in}\,x} wx$ and $xw \xrightarrow{\mathit{out}\,x} w$, for all $w \in X^*$ and $x \in X$.

The only difference between the definitions of $\mathcal{B}_X$ and $\mathcal{Q}_X$ is whether the states are considered as sequences or multisets. We will write $\mathcal{B}$ and $\mathcal{Q}$ without subscript if $X$ is clear from the context. $\mathcal{B}$ acts as an infinite capacity buffer which does not preserve the order of messages. For example, one possible sequence of transitions is

$$\emptyset \xrightarrow{\mathit{in}\,x} x \xrightarrow{\mathit{in}\,y} xy \xrightarrow{\mathit{in}\,z} xyz \xrightarrow{\mathit{out}\,y} xz \xrightarrow{\mathit{out}\,x} z \xrightarrow{\mathit{in}\,w} wz \ldots$$

$\mathcal{Q}$ acts as an infinite capacity first-in, first-out queue. A possible sequence of transitions is

$$\epsilon \xrightarrow{\mathit{in}\,x} x \xrightarrow{\mathit{in}\,y} xy \xrightarrow{\mathit{out}\,x} y \xrightarrow{\mathit{in}\,z} yz \xrightarrow{\mathit{in}\,w} yzw \xrightarrow{\mathit{out}\,y} zw \ldots$$

**Lemma 6.5.** *1. $\mathcal{B};\mathcal{B} \approx \mathcal{B}$ and $\mathcal{B};\mathcal{B} \not\sim \mathcal{B}$.*

*2. $\mathcal{Q};\mathcal{Q} \approx \mathcal{Q}$ and $\mathcal{Q};\mathcal{Q} \not\sim \mathcal{Q}$.*

*3. $\mathcal{Q};\mathcal{B} \approx \mathcal{B}$ and $\mathcal{Q};\mathcal{B} \not\sim \mathcal{B}$.*

*4. If $|X| \geqslant 2$, then $\mathcal{B};\mathcal{Q} \not\approx \mathcal{B}$ and $\mathcal{B};\mathcal{Q} \not\approx \mathcal{Q}$.*

*Proof.* 1.-3.: Define $\langle u, v\rangle R w$ iff $vu = w$, where $u$, $v$ and $w$ are multisets or sequences, as appropriate. In each case, $R$ is a weak bisimulation. To see that strong bisimilarity does not hold, observe that in each case, the composite agent has silent actions, while $\mathcal{B}$ and $\mathcal{Q}$ do not.

4.: Observe that $\mathcal{B};\mathcal{Q}$ has a transition $s_0 \overset{\mathit{in}\,x}{\Longrightarrow}\overset{\mathit{in}\,y}{\Longrightarrow} s_1$ from its initial state such that $s_1 \overset{\mathit{out}\,y}{\Longrightarrow}\overset{\mathit{out}\,x}{\Longrightarrow}$ is possible, but $s_1 \overset{\mathit{out}\,x}{\Longrightarrow}\overset{\mathit{out}\,y}{\Longrightarrow}$ is not. This is not the case for either $\mathcal{B}$ or $\mathcal{Q}$. Such properties are preserved under weak bisimulation. $\square$

The remainder of this chapter is devoted to examining the effect of composing arbitrary agents with buffers and queues.

### 6.1.4 Notions of asynchrony

In the asynchronous model of communication, messages are assumed to travel through a communication medium or *ether*. Sometimes, the medium is assumed to be first-in, first-out (a queue); sometimes, as in the asynchronous $\pi$-calculus, messages might be received in any order (a buffer).

Our approach is simple: we model the medium explicitly. An asynchronous agent is one whose output and/or input behaves as if filtered through either a buffer $\mathcal{B}$ or a queue $\mathcal{Q}$.

**Definition 6.6.** An agent $\mathbf{S}\colon X \to_B Y$ is

| | | | | | |
|---|---|---|---|---|---|
| ***out-buffered*** | if | $\mathbf{S} \approx \mathbf{S};\mathcal{B}$ | ***out-queued*** | if | $\mathbf{S} \approx \mathbf{S};\mathcal{Q}$ |
| ***in-buffered*** | if | $\mathbf{S} \approx \mathcal{B};\mathbf{S}$ | ***in-queued*** | if | $\mathbf{S} \approx \mathcal{Q};\mathbf{S}$ |
| ***buffered*** | if | $\mathbf{S} \approx \mathcal{B};\mathbf{S};\mathcal{B}$ | ***queued*** | if | $\mathbf{S} \approx \mathcal{Q};\mathbf{S};\mathcal{Q}$ |

We use the word ***asynchrony*** as a generic term to stand for any such property. The reason we distinguish six different notions is that, although it is probably most common to think of asynchrony as part of the *output* behavior of an agent, it is equally sensible to regard it as part of the *input* behavior, or both. Since input and output behave somewhat differently, we will study them separately. Yet another notion of asynchrony, incorporating feedback, will be defined in Section 6.3.2.

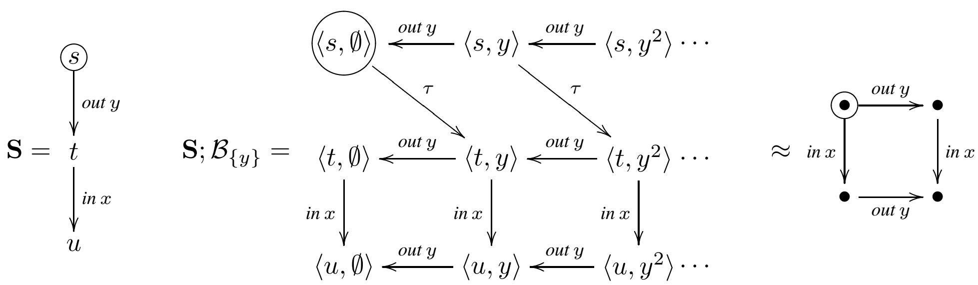
*Remark.* Because of Lemma 6.5, the operation of pre- or post-composing an agent with $\mathcal{B}$ or $\mathcal{Q}$ is idempotent up to $\approx$. Consequently, any agent of the form $\mathbf{S};\mathcal{B}$ is out-buffered, any agent of the form $\mathcal{B};\mathbf{S}$ is in-buffered, an agent is buffered iff it is in- and out-buffered, and so on. Also, each of the six properties is invariant under weak bisimulation.

Notice that it is almost never the case that an agent $\mathbf{S}$ is strongly bisimilar to $\mathbf{S};\mathcal{B}$ or to $\mathcal{B};\mathbf{S}$. This will be clear from the examples in Section 6.1.5. Weak bisimulation appears to be the finest equivalence relation that is sensible for studying asynchrony. It is also possible to consider coarser equivalences; the results of this chapter generalize in a straightforward way to any equivalence on processes that contains weak bisimulation; see Remark 6.12.

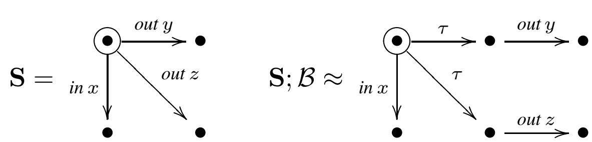
Let $B$ be a set. Buffered agents $\mathbf{S}\colon X \to_B Y$ form the morphisms of a category $\mathbf{Buf}_B$, whose objects are sets $X$, $Y$, etc.; the identity morphism on $X$ is given by the buffer $\mathcal{B}_X$. Similarly, queued agents form a category $\mathbf{Que}_B$. These categories have a symmetric monoidal structure, which will be described, along with other constructions on agents, in Section 6.3.1.

### 6.1.5 Examples

*Example 6.7.* The first example shows the effect of post-composing different agents with the buffer $\mathcal{B}$. Notice that although $\mathcal{B}$ has infinitely many states, $\mathbf{S};\mathcal{B}$ may have only finitely many states up to weak bisimulation.




*Example 6.8.*




*Example 6.9.* Here is an example on in-bufferedness. Notice that an input action is possible at every state of $\mathcal{B};\mathbf{S}$.

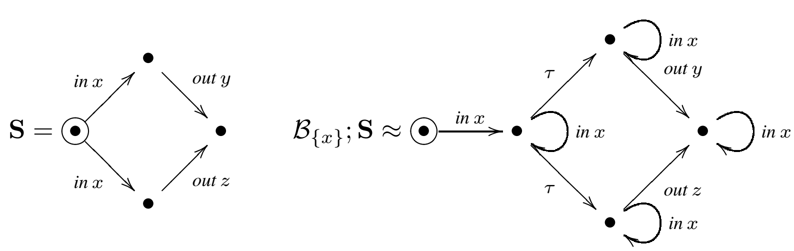


Table 6.1: First-order axioms for out-buffered agents

$$\begin{tikzcd} s \arrow[r, "out\,y"] & s' \arrow[d, "\alpha"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,y"] & t \end{tikzcd}$$

*output-commutativity (*OB1*)*

$$\begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,y"] & t \end{tikzcd}$$

where $\alpha \neq out\,y$

*output-confluence (*OB2*)*

$$\begin{tikzcd} s \arrow[r, "out\,y"] \arrow[d, "out\,y"'] & s' \\ s'' & \end{tikzcd} \Rightarrow s' = s''$$

*output-determinacy (*OB3*)*

## 6.2 First-order axioms for asynchrony

In this section, we will give necessary and sufficient conditions for each of the notions of asynchrony from Definition 6.6. These conditions are in the form of *first-order axioms*, by which we mean axioms that use quantification only over states and actions, but not over subsets of states or actions. The axioms, which are shown in Tables 6.1 through 6.4, characterize each of our notions of asynchrony *up to weak bisimulation*; this means, an LTS is asynchronous iff it is weakly bisimilar to one satisfying the axioms. It is possible to lift the condition "up to weak bisimulation" at the cost of introducing second-order axioms; this is the subject of Section 6.6.

### 6.2.1 Out-buffered agents

Table 6.1 lists three axioms for out-buffered agents. We use the convention that variables are implicitly existentially quantified if they occur only on the right-hand-side of an implication, and all other variables are implicitly universally quantified. Thus the axioms are:

**(**OB1**)** *Output-commutativity*: output actions can always be delayed.

**(**OB2**)** *Output-confluence*: when an output action and some other action are possible, then they can be performed in either order with the same result. In particular, neither action precludes the other.

**(**OB3**)** *Output-determinacy*: from any state $s$, there is at most one transition $out\,y$ for each $y \in Y$.

Each of these axioms is plausible for the behavior of a buffer. Output-determinacy is maybe the least intuitive of the three properties; the idea is that once an output action is stored in a buffer, there is only one way of retrieving it. Together, these axioms characterize out-bufferedness up to weak bisimulation:

**Theorem 6.10 (Characterization of out-buffered agents).** *An agent $\mathbf{S}$ is out-buffered if and only if $\mathbf{S} \approx \mathbf{T}$ for some $\mathbf{T}$ satisfying (*OB1*)–(*OB3*).*

This is a direct consequence of the following proposition:

**Proposition 6.11.**

1. *Every agent of the form $\mathbf{S};\mathcal{B}$ satisfies (*OB1*)–(*OB3*).*
2. *If $\mathbf{S}$ satisfies (*OB1*)–(*OB3*), then $\mathbf{S} \approx \mathbf{S};\mathcal{B}$.*

Table 6.2: First-order axioms for in-buffered agents

$$\begin{tikzcd} s \arrow[r, "\alpha"] & s' \arrow[d, "in\ x"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "\alpha"] \arrow[d, "in\ x"'] & s' \arrow[d, "in\ x"] \\ s'' \arrow[r, "\alpha"] & t \end{tikzcd}$$

*input-commutativity (IB1)*

$$\begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "\alpha"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "in\ x"] & t \end{tikzcd}$$

*input-confluence (IB2)*

$$\begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "in\ x"'] & s' \\ s'' & \end{tikzcd} \Rightarrow s' = s''$$

*input-determinacy (IB3)*

$$s \quad \Rightarrow \quad s \xrightarrow{in\ x} t$$

*input-receptivity (IB4)*

Table 6.3: First-order axioms for out-queued agents

$$\begin{tikzcd} s \arrow[r, "out\ y"] & s' \arrow[d, "\alpha"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\ y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\ y"] & t \end{tikzcd}$$

where $\alpha$ not output

*output-commutativity' (OQ1)*

$$\begin{tikzcd} s \arrow[r, "out\ y"] \arrow[d, "\alpha"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\ y"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\ y"] & t \end{tikzcd}$$

where $\alpha$ not output

*output-confluence' (OQ2)*

$$\begin{tikzcd} s \arrow[r, "out\ y"] \arrow[d, "out\ z"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{array}{c} y = z \\ \text{and} \\ s' = s'' \end{array}$$

*output-determinacy' (OQ3)*

Table 6.4: First-order axioms for in-queued agents

$$\begin{tikzcd} s \arrow[r, "\alpha"] & s' \arrow[d, "in\ x"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "\alpha"] \arrow[d, "in\ x"'] & s' \arrow[d, "in\ x"] \\ s'' \arrow[r, "\alpha"] & t \end{tikzcd}$$

where $\alpha$ not input

*input-commutativity' (IQ1)*

$$\begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "\alpha"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "in\ x"] & t \end{tikzcd}$$

where $\alpha$ not input

*input-confluence' (IQ2)*

$$\begin{tikzcd} s \arrow[r, "in\ x"] \arrow[d, "in\ x"'] & s' \\ s'' & \end{tikzcd} \Rightarrow s' = s''$$

*input-determinacy (IQ3)*

$$s \quad \Rightarrow \quad s \xrightarrow{in\ x} t$$

*input-receptivity (IQ4)*

*Proof.* 1. Clearly, the buffer $\mathcal{B}$ satisfies (OB1)–(OB3). Moreover, these conditions are preserved by arbitrary sequential composition from the left. We show this for (OB1); the other cases are similar. Suppose $\mathbf{B}$ satisfies (OB1). To show that $\mathbf{S};\mathbf{B}$ satisfies (OB1), consider transitions

$$\begin{tikzcd} \langle u,s\rangle \arrow[r, "out\,y"] & \langle u,s'\rangle \arrow[d, "\alpha"] \\ & \langle u',t\rangle. \end{tikzcd}$$

Then $s \xrightarrow{out\,y} s'$ in $\mathbf{B}$. By Definition 6.2, there are three cases for $\langle u,s'\rangle \xrightarrow{\alpha} \langle u',t\rangle$:

**Case 1:** $s' = t$, $u \xrightarrow{\alpha} u'$, $\alpha$ not output.
**Case 2:** $u = u'$, $s' \xrightarrow{\alpha} t$, $\alpha$ not input. Hence, by hypothesis there is $s''$ such that $s \xrightarrow{\alpha} s'' \xrightarrow{out\,y} t$.
**Case 3:** $\alpha = \tau$, $u \xrightarrow{out\,x} u'$, $s' \xrightarrow{in\,x} t$. Hence, by hypothesis there is $s''$ such that $s \xrightarrow{in\,x} s'' \xrightarrow{out\,y} t$.

In each of the three cases, the diagram can be completed:

Case 1:

$$\begin{tikzcd} \langle u,s\rangle \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & \langle u,t\rangle \arrow[d, "\alpha"] \\ \langle u',s\rangle \arrow[r, "out\,y"] & \langle u',t\rangle \end{tikzcd}$$

Case 2:

$$\begin{tikzcd} \langle u,s\rangle \arrow[r, "out\,y"] \arrow[d, "\alpha"'] & \langle u,s'\rangle \arrow[d, "\alpha"] \\ \langle u,s''\rangle \arrow[r, "out\,y"] & \langle u,t\rangle \end{tikzcd}$$

Case 3:

$$\begin{tikzcd} \langle u,s\rangle \arrow[r, "out\,y"] \arrow[d, "\tau"'] & \langle u,s'\rangle \arrow[d, "\tau"] \\ \langle u',s''\rangle \arrow[r, "out\,y"] & \langle u',t\rangle \end{tikzcd}$$

2. Suppose $\mathbf{S}\colon X \to_B Y$ satisfies (OB1)–(OB3). For any sequence $w = y_1 y_2 \cdots y_n \in Y^*$, we write $s \xrightarrow{out\,w} t$ if $s \xrightarrow{out\,y_1}\xrightarrow{out\,y_2} \cdots \xrightarrow{out\,y_n} t$ ($n \geqslant 0$). Note that if $w' \in Y^*$ is a permutation of $w$, then $s \xrightarrow{out\,w'} t$ iff $s \xrightarrow{out\,w} t$ by (OB1). Consider the relation $R \subseteq |\mathbf{S}| \times |\mathbf{S};\mathcal{B}|$ given by $sR\langle t,w\rangle$ iff $s \xrightarrow{out\,w} t$. Clearly, $R$ relates initial states. We show that $R$ is a weak bisimulation. In one direction, suppose

$$\begin{tikzcd} s \arrow[d, "\alpha"'] & R & \langle t,w\rangle \\ s'. \end{tikzcd}$$

Two cases arise:

**Case 1:** $\alpha = out\,y$ for some $y \in w$. By the definition of $R$, $s \xrightarrow{out\,y} s'' \xrightarrow{out\,w'} t$, where $w = yw'$. By (OB3), we have $s' = s''$. Therefore $s'R\langle t,w'\rangle$, and also $\langle t,w\rangle \xrightarrow{\alpha} \langle t,w'\rangle$.✓
**Case 2:** $\alpha \neq out\,y$ for all $y \in w$. From $s \xrightarrow{out\,w} t$ and $s \xrightarrow{\alpha} s'$, we get $s' \xrightarrow{out\,w} t'$ and $t \xrightarrow{\alpha} t'$ by repeated application of (OB2). Therefore $s'R\langle t',w\rangle$ and $\langle t,w\rangle \stackrel{\alpha}{\Rightarrow} \langle t',w\rangle$ (notice the use of $\Rightarrow$ here, which is necessary in case $\alpha$ is an output action).✓

In the other direction, suppose

$$\begin{tikzcd} s & R & \langle t,w\rangle \arrow[d, "\alpha"] \\ & & \langle t',w'\rangle. \end{tikzcd}$$

We distinguish three cases for $\langle t,w\rangle \xrightarrow{\alpha} \langle t',w'\rangle$, depending on which rule in Definition 6.2 was used.

**Case 1:** $t \xrightarrow{\alpha} t'$, $w = w'$ and $\alpha$ not output. Then $s \xrightarrow{out\,w} t \xrightarrow{\alpha} t'$, which implies $s \xrightarrow{\alpha} s' \xrightarrow{out\,w} t'$ by repeated application of (OB1), *i.e.* $s \xrightarrow{\alpha} s'R\langle t',w\rangle$.✓
**Case 2:** $t = t'$, $w \xrightarrow{\alpha} w'$ and $\alpha$ not input. Since $\mathcal{B}$ has only input and output transitions, $\alpha$ must be $out\,y$ for some $y \in Y$ with $w = yw'$. Then $s \xrightarrow{out\,y} s' \xrightarrow{out\,w'} t$, *i.e.* $s \xrightarrow{\alpha} s'R\langle t,w'\rangle$.✓

**Case 3:** $t \xrightarrow{\text{out}\,y} t'$, $w \xrightarrow{\text{in}\,y} w'$ and $\alpha = \tau$. In this case, $w' = wy$ and $s \xrightarrow{\text{out}\,w} t \xrightarrow{\text{out}\,y} t'$, hence $sR\langle t', w'\rangle$.✓ □

*Remark 6.12.* Theorem 6.10 generalizes to other notions of equivalence of processes, as long as they are coarser than weak bisimulation. Indeed, if $\cong$ is an equivalence of processes such that $\approx\ \subseteq\ \cong$, then for any agent $\mathbf{S}$, there exists some out-buffered $\mathbf{T}$ with $\mathbf{S} \cong \mathbf{T}$ iff there exists $\mathbf{T}'$ satisfying (OB1)–(OB3) and $\mathbf{S} \cong \mathbf{T}'$. This is a trivial consequence of Theorem 6.10. Similar remarks apply to the other results in this section and in Section 6.3.

### 6.2.2 In-buffered agents

The axioms for in-buffered agents are listed in Table 6.2. The main difference to the out-buffered case is the property *input-receptivity*: an in-buffered agent can perform any input action at any time. This was illustrated in Example 6.9. The input/output automata of Lynch and Stark [35] have this property, and so does Honda and Tokoro's original version of the asynchronous $\pi$-calculus [26].

*Remark.* Somewhat surprisingly, the axioms in Table 6.2 are not independent. In fact, (IB1) and (IB2) are equivalent in the presence of (IB3) and (IB4). We present all four axioms in order to highlight the analogy to the output case.

**Theorem 6.13 (Characterization of in-buffered agents).** *An agent* $\mathbf{S}$ *is in-buffered if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying (*IB1*)–(*IB4*).*

This is a consequence of the following proposition:

**Proposition 6.14.**

1. *Every agent of the form* $\mathcal{B};\mathbf{S}$ *satisfies (*IB1*)–(*IB4*).*
2. *If* $\mathbf{S}$ *satisfies (*IB1*)–(*IB4*), then* $\mathbf{S} \approx \mathcal{B};\mathbf{S}$.

*Proof.* The proof is much like the proof of Theorem 6.11. We give the details of 2. to demonstrate how each of the properties (IB1)–(IB4) is used.

2. Suppose $\mathbf{S}\colon X \rightarrow_B Y$ satisfies (IB1)–(IB4). For any sequence $w = x_1x_2\cdots x_n \in X^*$ we write $s \xrightarrow{\text{in}\,w} t$ if $s \xrightarrow{\text{in}\,x_1}\xrightarrow{\text{in}\,x_2} \cdots \xrightarrow{\text{in}\,x_n} t$ ($n \geqslant 0$). Again, notice that if $w' \in X^*$ is a permutation of $w$, then $s \xrightarrow{\text{in}\,w'} t$ iff $s \xrightarrow{\text{in}\,w} t$ by (IB1). Consider the relation $R \subseteq |\mathcal{B};\mathbf{S}| \times |\mathbf{S}|$ given by $\langle w, s\rangle Rt$ iff $s \xrightarrow{\text{in}\,w} t$. $R$ relates initial states, and we show that it is a weak bisimulation. In one direction, suppose

$$\begin{tikzcd} \langle w,s\rangle \ R & t \arrow[d, "\alpha"] \\ & t'. \end{tikzcd}$$

Then $s \xrightarrow{\text{in}\,w} t$, hence $\langle w, s\rangle \overset{\tau}{\Rightarrow} \langle \emptyset, t\rangle \overset{\alpha}{\Rightarrow} \langle \emptyset, t'\rangle$. But clearly $\langle \emptyset, t'\rangle Rt'$.

In the other direction, suppose

$$\begin{tikzcd} \langle w,s\rangle \arrow[d, "\alpha"'] & R \ t \\ \langle w',s'\rangle. & \end{tikzcd}$$

We distinguish the usual three cases by Definition 6.2.

**Case 1:** $s = s'$, $w \xrightarrow{\alpha} w'$ and $\alpha$ not output. In this case, $\alpha = \text{in}\,x$ for some $x \in X$ with $w' = wx$. By definition of $R$, $s \xrightarrow{\text{in}\,w} t \xrightarrow{\text{in}\,x} t'$, hence $\langle w', s\rangle Rt'$.✓

**Case 2:** $s \xrightarrow{\alpha} s'$, $w = w'$ and $\alpha$ not input. To $s \xrightarrow{\alpha} s'$ and $s \xrightarrow{\text{in}\,w} t$ repeatedly apply (IB2) to get $t \xrightarrow{\alpha} t'$ and $s' \xrightarrow{\text{in}\,w} t'$, hence $\langle w, s'\rangle Rt'$.✓

**Case 3:** $w \xrightarrow{\text{out}\,x} w'$, $s \xrightarrow{\text{in}\,x} s'$ and $\alpha = \tau$. Then $w = xw'$ and $s \xrightarrow{\text{in}\,x} s'' \xrightarrow{\text{in}\,w'} t$. But by (IB3), $s' = s''$, hence $s' \xrightarrow{\text{in}\,w'} t$, therefore $\langle w', s'\rangle Rt$.✓ □

### 6.2.3 Out-queued and in-queued agents

The results for buffers are easily adapted to queues. The relevant properties are given in Tables 6.3 and 6.4. Notice that the conditions for *commutativity* and *confluence* differ from the respective rules in the buffered case only in their side conditions. Different outputs (respectively, different inputs) no longer commute or conflow. *Output-determinacy* is strengthened: from each state, there is at most one possible output transition.

Note that (IB1)–(IB4) imply (IQ1)–(IQ4). This is due to the fact that every in-buffered agent is also in-queued as a consequence of Lemma 6.5(3). On the other hand, no implication holds between (OQ1)–(OQ3) and (OB1)–(OB3), since out-bufferedness and out-queuedness are incomparable notions due to Lemma 6.5(4).

Just like in the buffered case, the axioms for input are not independent: we have (IQ1) $\Longleftrightarrow$ (IQ2) in the presence of the other axioms.

**Theorem 6.15 (Characterization of in- and out-queued agents).** *An agent* $\mathbf{S}$ *is out-queued if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying (*OQ1*)–(*OQ3*). Moreover,* $\mathbf{S}$ *is in-queued if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some* $\mathbf{T}$ *satisfying (*IQ1*)–(*IQ4*).* □

## 6.3 More agent constructors and asynchrony with feedback

### 6.3.1 Some operations on agents

In this section, we will introduce some operations on agents, such as renaming and hiding of actions, parallel composition and feedback.

1. *Domain extension.* If $\mathbf{S}$ is an LTS of type $A$, and if $A \subseteq A'$, then $\mathbf{S}$ can also be regarded as an LTS of type $A'$.

2. *Domain restriction (hiding).* If $\mathbf{S}$ is an LTS of type $A$, and if $\tau \in A' \subseteq A$, then $\mathbf{S}|_{A'}$ is defined to be the LTS of type $A'$ which has the same states as $\mathbf{S}$, and whose transitions are those of $\mathbf{S}$ restricted to $|\mathbf{S}| \times A' \times |\mathbf{S}|$.

3. *Composition with functions.* Let $\mathbf{S}\colon X \to_B Y$, and let $f\colon X' \to X$ and $g\colon Y \to Y'$ be functions. By $f;\mathbf{S};g$ we denote the agent of type $X' \to_B Y'$ with the same states as $\mathbf{S}$, and with input transitions $s \xrightarrow{in\,x}_{f;\mathbf{S};g} t$ if $s \xrightarrow{in\,fx}_{\mathbf{S}} t$, output transitions $s \xrightarrow{out\,gy}_{f;\mathbf{S};g} t$ if $s \xrightarrow{out\,y}_{\mathbf{S}} t$, and with $s \xrightarrow{\alpha}_{f;\mathbf{S};g} t$ iff $s \xrightarrow{\alpha}_{\mathbf{S}} t$ when $\alpha$ is an internal action.

Domain extension, domain restriction and composition with functions are special cases of the following, general renaming construct:

4. *General renaming and hiding.* Let $\mathbf{S}$ be an LTS of type $A$ and let $r \subseteq A \times A'$ be a relation such that $\tau r \alpha'$ iff $\tau = \alpha'$. Define $\mathbf{S}_r$ to be the LTS of type $A'$ that has the same states and initial state as $\mathbf{S}$ and transitions $s \xrightarrow{\alpha}_{\mathbf{S}_r} t$ iff $s \xrightarrow{\alpha'}_{\mathbf{S}} t$ for some $\alpha r \alpha'$.

Let us now turn to various forms of parallel composition.

5. *Parallel composition without interaction.* Let $\mathbf{S}$ and $\mathbf{T}$ be LTSs of type $A$. Then $\mathbf{S}\|\mathbf{T}$ is the LTS of type $A$ with states $|\mathbf{S}| \times |\mathbf{T}|$ and initial state $\langle s_0, t_0\rangle$, and whose transitions are given by the rules

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} s'}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S}\|\mathbf{T}} \langle s',t\rangle} \qquad \frac{t \xrightarrow{\alpha}_{\mathbf{T}} t'}{\langle s,t\rangle \xrightarrow{\alpha}_{\mathbf{S}\|\mathbf{T}} \langle s,t'\rangle}.$$

6. *Symmetric monoidal structure.* Let $X \oplus X'$ be the disjoint union of sets. For $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon X' \to_B Y'$, define $\mathbf{S} \oplus \mathbf{T}\colon X \oplus X' \to_B Y \oplus Y'$ to be the agent $\mathbf{S}_r\|\mathbf{T}_q$, where $r$ and $q$ are the inclusions of $X \to_B Y$, respectively $X' \to_B Y'$ into $X \oplus X' \to_B Y \oplus Y'$. Then $\oplus$ defines a symmetric monoidal structure on the categories **Buf** and **Que**. The tensor unit is given by the agent $\mathbf{I}$ of type $\emptyset \to \emptyset$ with one state and no transitions.

The constructors we have considered so far, including sequential composition, are not sufficient to build arbitrary networks. What is missing is the ability to construct loops. The next constructor allows the output of an agent to be connected to its own input:

7. *Self-composition (feedback).* Let $\mathbf{S}\colon X \to_B Y$. Let $O \subseteq Y \times X$ be a set of pairs. Define $\mathbf{S} \circlearrowleft O$, the self-composition of $\mathbf{S}$ along $O$, to be the LTS of type $X \to_B Y$ whose states are identical with those of $\mathbf{S}$, and whose transitions are given by the rules

$$\frac{s \xrightarrow{\alpha}_{\mathbf{S}} t}{s \xrightarrow{\alpha}_{\mathbf{S}\circlearrowleft O} t} \qquad \frac{s \xrightarrow{out\,y}\overset{\tau}{\Rightarrow}\xrightarrow{in\,x}_{\mathbf{S}} t \qquad \langle y,x\rangle \in O}{s \xrightarrow{\tau}_{\mathbf{S}\circlearrowleft O} t}.$$

   In the common case where $\mathbf{S}\colon X \to_B X$ and $O = \{\langle x,x\rangle \mid x \in X\}$, we will write $\mathbf{S}^\circ$ instead of $\mathbf{S} \circlearrowleft O$.

We can use self-composition to define both sequential and parallel composition.

8. *Sequential composition.* The sequential composition of agents was defined in Definition 6.2. Alternatively, one can define it from the more primitive notions of direct sum, feedback and hiding: Let $\mathbf{S}\colon X \to_B Y$ and $\mathbf{T}\colon Y \to_B Z$. Then $\mathbf{S} \oplus \mathbf{T}\colon X \oplus Y \to_B Y \oplus Z$, and with $\Delta Y = \{\langle y,y\rangle \mid y \in Y\}$, one gets $\mathbf{S};\mathbf{T} \approx ((\mathbf{S} \oplus \mathbf{T}) \circlearrowleft \Delta Y)|_{X \to_B Z}$.

9. *Parallel composition (with interaction).* Let $\mathbf{S},\mathbf{T}\colon X \to_B X$. The parallel composition $\mathbf{S}|\mathbf{T}$ is defined to be the agent $(\mathbf{S}\|\mathbf{T})^\circ$.

**Proposition 6.16.** *All of the agent constructors in this section respect weak bisimulation. For instance, if* $\mathbf{S} \approx \mathbf{S}'$ *and* $\mathbf{T} \approx \mathbf{T}'$*, then* $\mathbf{S}_r \approx \mathbf{S}'_r$ *and* $\mathbf{S}\|\mathbf{T} \approx \mathbf{S}'\|\mathbf{T}'$*, etc.* □

### 6.3.2 Asynchrony with feedback

In concurrent process calculi such as CCS or the $\pi$-calculus, we do not think of channels as edges in a data flow graph, but rather we think of a single global ether through which all messages travel. This idea is most visible in the *chemical semantics* of these calculi [8]. There the ether is modeled as a "chemical solution", which is a multiset of processes, some of which are transient messages. As a consequence, messages that are emitted from a process are immediately available as input to all processes, including the sending process itself. In our setting, this is best modeled by requiring that all processes are of type $X \to X$ for one fixed set $X$, and by using self-composition to feed the output back to the input.

In the presence of feedback, out-bufferedness takes a slightly different form, which is expressed in the following definition.

**Definition.** An agent $\mathbf{S}\colon X \to_B X$ is ***out-buffered with feedback*** if $\mathbf{S} \approx \mathbf{R}^\circ$ for some out-buffered agent $\mathbf{R}$.

*Example 6.17.* The following agent $\mathbf{S}$ is out-buffered with feedback, but not out-buffered:

$$\mathbf{S} = \begin{tikzcd} \odot \arrow[r, "out\,x"] \arrow[d, "in\,x"'] \arrow[dr, "\tau"] & \bullet \arrow[r, "out\,x"] \arrow[d, "in\,x"'] \arrow[dr, "\tau"] & \bullet \arrow[d, "in\,x"] \\ \bullet \arrow[r, "out\,x"'] & \bullet \arrow[r, "out\,x"'] & \bullet \end{tikzcd}.$$

*Remark.* Recently, Amadio, Castellani and Sangiorgi [4] have given a definition of asynchronous bisimulation, which accounts for the fact that an agent of type $X \to X$ might receive a message, and then immediately send it again, without this interaction being observable on the outside. Feedback is concerned with the dual phenomenon, namely a process that sends a message and then immediately receives it again.

Out-bufferedness with feedback is characterized up to weak bisimulation by the first-order axioms that are listed in Table 6.5.

Table 6.5: First-order axioms for out-buffered agents with feedback

$$\begin{tikzcd} s \arrow[r, "out\,x"] & s' \arrow[d, "\alpha"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,x"] & t \end{tikzcd}$$

*output-commutativity* (FB1)

$$\begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\alpha"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\alpha"'] & s' \arrow[d, "\alpha"] \\ s'' \arrow[r, "out\,x"] & t \end{tikzcd}$$

where $\alpha \neq out\,x$ and $\alpha \neq \tau$

*output-confluence* (FB2)

$$\begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "out\,x"'] & s' \\ s'' & \end{tikzcd} \Rightarrow s' = s''$$

*output-determinacy* (FB3)

$$\begin{tikzcd} s \arrow[r, "out\,x"] & s' \arrow[d, "in\,x"] \\ & t \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,x"] \arrow[dr, "\tau"'] & s' \arrow[d, "in\,x"] \\ & t \end{tikzcd}$$

*feedback* (FB4)

$$\begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\tau"'] & s' \\ s'' & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\tau"'] & s' \arrow[d, "\tau"] \\ s'' \arrow[r, "out\,x"] & t \end{tikzcd} \text{ or } \begin{tikzcd} s \arrow[r, "out\,x"] \arrow[d, "\tau"'] & s' \arrow[dl, "in\,x"] \\ s'' & \end{tikzcd}$$

*output-tau* (FB5)

**Theorem 6.18 (Characterization of out-buffered agents with feedback).** *An agent* $\mathbf{S}\colon X \to_B X$ *is out-buffered with feedback if and only if* $\mathbf{S} \approx \mathbf{T}$ *for some agent* $\mathbf{T}$ *satisfying* (FB1)–(FB5)*.*

Before we prove this theorem, we need two lemmas. The first one gives a useful consequence of the axioms for out-bufferedness with or without feedback.

**Lemma 6.19.** *Suppose an agent* $\mathbf{S}$ *satisfies either* (OB1)–(OB3) *or* (FB1)–(FB5)*. Then it satisfies the following property, which we call* ***backwards output-determinacy****:*

$$\begin{tikzcd} s \arrow[d, "out\,x"'] & s' \arrow[d, "out\,x"] \\ t \arrow[r, phantom, "\approx"] & t' \end{tikzcd} \Rightarrow s \approx s'.$$

*Proof.* The proof is straightforward. The relation $R := \{\langle s, s'\rangle \mid s \approx s' \text{ or } (\exists t, t') s \xrightarrow{out\,x} t \approx t' \xleftarrow{aoutx} s'\}$ is weak bisimulation that relates $s$ and $s'$. $\square$

The next lemma establishes a technical property needed in the proof of Theorem 6.18. Recall that an agent $\mathbf{T}$ is ***≈-reduced*** if $\mathbf{T} = \mathbf{T}/{\approx}$.

**Lemma 6.20.** *Assume* $\mathbf{T}$ *is ≈-reduced and satisfies* (FB1)–(FB5)*. Define a subset* $A \subseteq \{\langle s, t\rangle \mid s \xrightarrow{\tau} t\}$ *as follows:* $\langle s, t\rangle \in A$ *iff for all sequences* $w \in X^*$*,*

$$\begin{tikzcd} s \arrow[r, "out\,w"] \arrow[d, "\tau"'] & u \\ t & \end{tikzcd} \Rightarrow \begin{tikzcd} s \arrow[r, "out\,w"] \arrow[d, "\tau"'] & u \arrow[d, "\tau"] \\ t \arrow[r, "out\,w"] & v. \end{tikzcd}$$

*Then the following hold:*

1. *Whenever* $s \xrightarrow{\tau} t \xrightarrow{out\,x} t'$ *and* $s \xrightarrow{out\,x} s' \xrightarrow{\tau} t'$*, then* $\langle s, t\rangle \in A$ *iff* $\langle s', t'\rangle \in A$*.*

2. *If* $s \xrightarrow{\tau} t$ *and* $\langle s,t\rangle \notin A$, *then* $s \xrightarrow{out\,x}\xrightarrow{in\,x} t$ *for some* $x \in X$.

*Proof.* 1. $\Rightarrow$: Assume $\langle s,t\rangle \in A$ and $s' \xrightarrow{out\,w} u$. Then there are $v$ and $t''$ with $u \xrightarrow{\tau} v$ and $t \xrightarrow{out\,x} s'' \xrightarrow{out\,w} v$. By (FB3), $s' = s''$, hence $s' \xrightarrow{out\,w} v$ and $u \xrightarrow{\tau} v$. This shows $\langle s',t'\rangle \in A$.

$\Leftarrow$: Conversely, assume $\langle s',t'\rangle \in A$ and $s \xrightarrow{out\,w} u$. We show that there exists $v$ with $u \xrightarrow{\tau} v$ and $t \xrightarrow{out\,w} v$.

**Case 1:** $x \notin w$. We get $s' \xrightarrow{out\,w} u'$ and $u \xrightarrow{out\,x} u'$ by (FB2), and $t' \xrightarrow{out\,w} v'$ and $u' \xrightarrow{\tau} v'$ by the assumption that $\langle s',t'\rangle \in A$, then $u \xrightarrow{\tau} v_1 \xrightarrow{out\,x} v'$ and also $t \xrightarrow{out\,w} v_2 \xrightarrow{out\,x} v'$ by (FB1). By Lemma 6.19, $v_1 \approx v_2$, hence, since $\mathbf{T}$ is $\approx$-reduced, $v_1 = v_2$. We can take $v = v_1$.

**Case 2:** $x \in w$. Let $xw'$ be a permutation of $w$ that begins with $x$. By (FB1), $s \xrightarrow{out\,x} s'' \xrightarrow{out\,w'} u$, and by (FB3), $s' = s''$. Since $\langle s',t'\rangle \in A$, one has $u \xrightarrow{\tau} v$ and $t' \xrightarrow{out\,w'} v$ for some $v$, hence $t \xrightarrow{out\,xw'} v$ and again by (FB3), $t \xrightarrow{out\,w} v$.

2. Assume $s \xrightarrow{\tau} t$ and $\langle s,t\rangle \notin A$. By definition of $A$, there exists $w \in X^*$ with $s \xrightarrow{out\,w} u$ such that there exists no $v$ with $t \xrightarrow{out\,w} v$ and $u \xrightarrow{\tau} v$. Choose such a $w$ of minimal length, and let $w = w'x$ (note $w$ cannot be the empty sequence). Then $s \xrightarrow{out\,w'} s' \xrightarrow{out\,x} u$, $t \xrightarrow{out\,w'} t'$, and $s' \xrightarrow{\tau} t'$, and there is no $v$ with $t' \xrightarrow{out\,x} v$ and $u \xrightarrow{\tau} v$. By (FB5), there is a transition $u \xrightarrow{in\,x} t'$. From $s \xrightarrow{out\,w'} s' \xrightarrow{out\,x} u \xrightarrow{in\,x} t'$ and (FB1), one gets $s \xrightarrow{out\,x}\xrightarrow{in\,x} t'' \xrightarrow{out\,w'} t'$. By Lemma 6.19, $t'' \approx t$, hence $t'' = t$ since $\mathbf{T}$ is $\approx$-reduced. This shows $s \xrightarrow{out\,x}\xrightarrow{in\,x} t$. ☐

*Proof of Theorem 6.18:* Consider the following auxiliary operation on agents: For $\mathbf{R}\colon X {\to_B} X$, define $\mathbf{R}^\bullet$ by

$$\frac{s \xrightarrow{\alpha}_{\mathbf{R}} t}{s \xrightarrow{\alpha}_{\mathbf{R}^\bullet} t} \qquad \frac{s \xrightarrow{out\,x}_{\mathbf{R}}\xrightarrow{in\,x}_{\mathbf{R}} t}{s \xrightarrow{\tau}_{\mathbf{R}^\bullet} t}.$$

In general, $(-)^\bullet$ does not respect weak bisimulation. Notice that if $\mathbf{R}$ satisfies (OB1) or (IB1), then $\mathbf{R}^\circ \approx \mathbf{R}^\bullet$.

$\Rightarrow$: Suppose $\mathbf{S}\colon X {\to_B} X$ is out-buffered with feedback. Then there is some $\mathbf{R}$ satisfying (OB1)–(OB3), such that $\mathbf{S} \approx \mathbf{R}^\circ$. It is straightforward to verify that $\mathbf{R}^\bullet$ satisfies (FB1)–(FB5), and we can take $\mathbf{T} = \mathbf{R}^\bullet \approx \mathbf{R}^\circ \approx \mathbf{S}$.

$\Leftarrow$: Suppose $\mathbf{T}\colon X {\to_B} X$ satisfies (FB1)–(FB5). We will show $\mathbf{T}$ is out-buffered with feedback. Notice that $\mathbf{T}/{\approx}$ also satisfies (FB1)–(FB5), hence we can without loss of generality assume that $\mathbf{T}$ is $\approx$-reduced. Define a subset $A \subseteq \{\langle s,t\rangle \mid s \xrightarrow{\tau} t\}$ as in Lemma 6.20. Let $\mathbf{R}\colon X {\to_B} X$ be the agent obtained from $\mathbf{T}$ by removing all transitions of the form $s \xrightarrow{\tau} t$ where $\langle s,t\rangle \notin A$. More precisely, $|\mathbf{R}| = |\mathbf{T}|$ and $s \xrightarrow{\alpha}_{\mathbf{R}} t$ iff $\alpha \neq \tau$ and $s \xrightarrow{\alpha}_{\mathbf{T}} t$, or $\alpha = \tau$ and $\langle s,t\rangle \in A$. We claim that $\mathbf{R}$ satisfies (OB1)–(OB3). Indeed, (OB1) and (OB2) follow from the respective properties of $\mathbf{T}$ in the case where $\alpha \neq \tau$. In the case where $\alpha = \tau$, (OB1) for $\mathbf{R}$ follows from (FB1) for $\mathbf{T}$ and Lemma 6.20(1,$\Leftarrow$); whereas (OB2) follows from the definition of $A$ and Lemma 6.20(1,$\Rightarrow$). Finally, (OB3) for $\mathbf{R}$ follows directly from (FB3) for $\mathbf{T}$.

We now show that $\mathbf{T} = \mathbf{R}^\bullet$. The two agents have the same states. For transitions, first note that $\to_{\mathbf{R}} \subseteq \to_{\mathbf{T}}$, and hence $\to_{\mathbf{R}^\bullet} \subseteq \to_{\mathbf{T}^\bullet} = \to_{\mathbf{T}}$, with the latter equality holding because of (FB4). For the converse, assume $s \xrightarrow{\alpha}_{\mathbf{T}} t$. If $\alpha \neq \tau$ or $\langle s,t\rangle \in A$, then $s \xrightarrow{\alpha}_{\mathbf{R}} t$ and we are done. Else $\alpha = \tau$ and $\langle s,t\rangle \notin A$, and by Lemma 6.20(2), $s \xrightarrow{out\,x}\xrightarrow{in\,x} t$ holds in $\mathbf{T}$, hence in $\mathbf{R}$. This shows $s \xrightarrow{\tau}_{\mathbf{R}^\bullet} t$.

We have shown that $\mathbf{T} = \mathbf{R}^\bullet = \mathbf{R}^\circ$ for some $\mathbf{R}$ satisfying (OB1)–(OB3). Hence, $\mathbf{T}$ is out-buffered with feedback, which finishes the proof of Theorem 6.18. ☐

## 6.4 Example: Asynchronous CCS

In this section, we will show that an asynchronous version of Milner's calculus of communicating systems (CCS) [40, 41] fits into the framework of out-buffered labeled transition systems with feedback.

Table 6.6: Transitions rules for asynchronous CCS

$$
\begin{array}{llll}
(\mathit{act}) & \dfrac{}{\alpha.P \xrightarrow{\alpha} P} & (\mathit{synch}) & \dfrac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\bar{\alpha}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \\[2ex]
(\mathit{sum}) & \dfrac{G \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P} & (\mathit{res}) & \dfrac{P \xrightarrow{\alpha} P' \qquad \alpha \notin L \cup \bar{L}}{P \setminus L \xrightarrow{\alpha} P' \setminus L} \\[2ex]
(\mathit{sum}') & \dfrac{G' \xrightarrow{\alpha} P}{G + G' \xrightarrow{\alpha} P} & (\mathit{rel}) & \dfrac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f\alpha} P'[f]} \\[2ex]
(\mathit{comp}) & \dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} & (\mathit{rec}) & \dfrac{P \xrightarrow{\alpha} P' \qquad A \overset{\mathrm{def}}{=} P}{A \xrightarrow{\alpha} P'} \\[2ex]
(\mathit{comp}') & \dfrac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'} & &
\end{array}
$$

Let $X = \{a, b, c, \dots\}$ be an infinite set of ***names***, and let $\bar{X} = \{\bar{a}, \bar{b}, \bar{c}, \dots\}$ be a corresponding set of ***co-names***, such that $X$ and $\bar{X}$ are disjoint and in one-to-one correspondence via $(\bar{\ })$. We also write $\bar{\bar{a}} = a$. Names correspond to input-actions, and co-names to output-actions. Let $\tau \notin X + \bar{X}$, and let $\mathit{Act} = X + \bar{X} + \{\tau\}$ be the set of ***actions***. We use the letters $\alpha, \beta, \dots$ for actions. We use the letter $L$ for sets of names, and we write $\bar{L}$ for $\{\bar{a} \mid a \in L\}$. We use the letter $f$ for ***relabeling functions***, which are functions $f : X \to X$. Any relabeling function extends to $f : \mathit{Act} \to \mathit{Act}$ by letting $f\bar{a} = \overline{fa}$ and $f\tau = \tau$.

Let $A, B, C, \dots$ range over a fixed set of ***process constants***. Asynchronous CCS ***processes*** $P, Q, \dots$ and ***guards*** $G, H, \dots$ are given by the following grammars:

$$P ::= \bar{a}.0 \;\big|\; P|P \;\big|\; Q \setminus L \;\big|\; P[f] \;\big|\; A \;\big|\; G$$

$$G ::= a.P \;\big|\; \tau.P \;\big|\; G + H \;\big|\; \mathbf{0}$$

Notice that the choice operator $+$ is restricted to input- and $\tau$-guarded processes. Output-guarded choice is traditionally disallowed in asynchronous process calculi. This is in accordance with the results of this chapter, since output-guarded choice violates the two asynchronous principles of output-determinacy and output-confluence. For the $\pi$-calculus, Nestmann and Pierce [44] have recently shown that input-guarded choice can be encoded from the other constructs; hence they include it in their version of the asynchronous $\pi$-calculus, and we include it here for asynchronous CCS as well.

Assume a set of ***defining equations*** $A \overset{\mathrm{def}}{=} P$, one for each process constant $A$. The operational semantics of asynchronous CCS is given in terms of a labeled transition system $\mathbf{S}_{\mathrm{CCS}} = \langle S, \mathit{Act}, \to \rangle$, which is defined in Table 6.6. The states are CCS processes. Notice that we have not specified a distinguished initial state; this is more convenient in this context, and no harm is done. Also notice that there is no rule for $\mathbf{0}$. This is because the process $\mathbf{0}$ is inert, *i.e.* there are no transitions $\mathbf{0} \xrightarrow{\alpha} P$.

**Lemma 6.21.** *If $G \xrightarrow{\alpha} P$ for a guard $G$, then $\alpha \notin \bar{X}$, i.e. $\alpha$ is not an output action.*

*Proof.* By induction on the derivation of $G \xrightarrow{\alpha} P$. □

To fit the labeled transition system $\mathbf{S}_{\mathrm{CCS}}$ into our framework of labeled transition systems with input and output, we simply identify the set $X$ of names with $\mathit{In}\, X$, and the set $\bar{X}$ of co-names with $\mathit{Out}\, X$. Then $\mathbf{S}_{\mathrm{CCS}}$ is a labeled transition system of type $X \to X$. Before we prove that this system is out-buffered with feedback, observe that output-determinacy fails for $\mathbf{S}_{\text{CCS}}$:

$$\begin{tikzcd}
\bar{a}.\mathbf{0}|\bar{a}.\mathbf{0} \arrow[r, "\bar{a}"] \arrow[d, "\bar{a}"'] & \mathbf{0}|\bar{a} \\
\bar{a}|\mathbf{0},
\end{tikzcd}$$

and $\mathbf{0}|\bar{a} \neq \bar{a}|\mathbf{0}$. The following lemma helps to remedy the situation:

**Lemma 6.22.** *An agent* $\mathbf{S}$ *is out-buffered with feedback if it satisfies (*FB1*), (*FB2*), (*FB5*), (*FB4*) and the following property (*WEAK-FB3*), which we call* ***weak output-determinacy****:*

$$\begin{tikzcd}
s \arrow[r, "out\ y"] \arrow[d, "out\ y"'] & s' \\
s''
\end{tikzcd}
\quad\Rightarrow\quad
\begin{tikzcd}
s \arrow[r, "out\ y"] \arrow[d, "out\ y"'] & s' \arrow[d, "out\ y"] \\
s'' \arrow[r, "out\ y"] & t
\end{tikzcd}
\quad \text{or} \quad s' = s''$$

*Proof.* First notice that if $\mathbf{S}$ satisfies the hypothesis, then so does $\mathbf{S}/\approx$, hence one can without loss of generality assume that $\mathbf{S}$ is $\approx$-reduced. Next, one shows backwards output-determinacy as in Lemma 6.19. For a $\approx$-reduced process, backwards output-determinacy and (WEAK-FB3) already implies (FB3), and therefore $\mathbf{S}$ is out-buffered with feedback by Theorem 6.18. □

**Theorem 6.23.** *The labeled transition system* $\mathbf{S}_{\text{CCS}}$ *is out-buffered with feedback.*

*Proof.* By Lemma 6.22, it suffices to show that $\mathbf{S}_{\text{CCS}}$ satisfies the axioms (FB1), (FB2), (WEAK-FB3), (FB5), and (FB4). Each of these is proved in a similar fashion. (FB1), (FB2), (WEAK-FB3) and (FB4) can be proved independently, while (FB5) relies on (FB2) and (WEAK-FB3) as hypotheses. Since this is the most interesting case, we show only the proof of (FB5). Suppose therefore that (FB2) and (WEAK-FB3) have already been proved. We want to show

$$\begin{tikzcd}
P \arrow[r, "\bar{b}"] \arrow[d, "\tau"'] & Q \\
R
\end{tikzcd}
\quad\Rightarrow\quad
\begin{tikzcd}
P \arrow[r, "\bar{b}"] \arrow[d, "\tau"'] & Q \arrow[d, "\tau"] \\
R \arrow[r, "\bar{b}"] & S
\end{tikzcd}
\quad \text{or} \quad
\begin{tikzcd}
P \arrow[r, "\bar{b}"] \arrow[d, "\tau"'] & Q \arrow[dl, "b"] \\
R
\end{tikzcd}
\quad .$$

We show this by induction on the derivation of $P \xrightarrow{\bar{b}} Q$. We distinguish six cases based on the last rule in that derivation. Remember that this last rule cannot have been (*sum*) or (*sum*$'$) by Lemma 6.21.

(*act*): $P = \bar{b}.\mathbf{0}$ and $Q = \mathbf{0}$. This is impossible, since $\bar{b}.\mathbf{0} \stackrel{\tau}{\not\to} R$.

(*comp*): $P = P'|P''$ and $Q = Q'|P''$, where $P' \xrightarrow{\bar{b}} Q'$. Then $P \xrightarrow{\tau} R$ must have been inferred by one of the rules (*comp*), (*comp*$'$) or (*synch*). Therefore, $R = R'|R''$, and one of the following holds:

**Case 1:** $P' \xrightarrow{\tau} R'$ and $P'' = R''$. By induction hypothesis on $P' \xrightarrow{\tau} R'$ and $P' \xrightarrow{\bar{b}} Q'$, either there is $S'$ with $R' \xrightarrow{\bar{b}} S'$ and $Q' \xrightarrow{\tau} S'$, in which case we can choose $S = S'|P''$; or else $Q' \xrightarrow{b} R'$, and hence $Q = Q'|P'' \xrightarrow{b} R'|P'' = R$.
**Case 2:** $P' = R'$ and $P'' \xrightarrow{\tau} R''$. Then one can choose $S = Q'|R''$.
**Case 3:** $P' \xrightarrow{\alpha} R'$ and $P'' \xrightarrow{\bar{\alpha}} R''$. In case $\alpha \neq \bar{b}$, we can use (FB2) to get $R' \xrightarrow{\bar{b}} S'$ and $Q' \xrightarrow{\alpha} S'$, and we let $S = S'|R''$. In case $\alpha = \bar{b}$, we can use (WEAK-FB3) to get either $R' \xrightarrow{\bar{b}} S'$ and $Q' \xrightarrow{\bar{b}} S'$, and we let again $S = S'|R''$; or else $R' = Q'$, and hence $Q = Q'|P'' \xrightarrow{b=\bar{\alpha}} Q'|R'' = R$.

(*comp*$'$): This case is symmetric to the previous one.

Table 6.7: Transitions rules for the core join calculus

| | | | |
|---|---|---|---|
| $(str1)$ | $\Delta \vdash_N \Pi, P\vert Q$ | $\rightharpoonup$ | $\Delta \vdash_N \Pi, P, Q$ |
| $(str2)$ | $\Delta \vdash_N \Pi, \mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P$ | $\rightharpoonup$ | $\Delta, R_1, \ldots, R_m \vdash_{N'} \Pi, P$ |
| | | | where $N' = N + dn(R_1, \ldots, R_m)$ |
| $(join)$ | $\Delta \vdash_N \Pi, x_1\langle\tilde{y_1}\rangle, \ldots, x_n\langle\tilde{y_n}\rangle$ | $\mapsto$ | $\Delta \vdash_N \Pi, [\tilde{y_1}/\tilde{v}_1, \ldots, \tilde{y_n}/\tilde{v}_n]P$ |
| | | | where $(x_1(\tilde{v}_1)\vert \ldots \vert x_n(\tilde{v}_n) \rhd P) \in \Delta$ |

(*res*): $P = P' \setminus L$ and $Q = Q' \setminus L$, where $P' \xrightarrow{\bar{b}} Q'$ and $b \notin L$. Then $R = R' \setminus L$ and $P' \xrightarrow{\tau} R'$. By induction hypothesis, we get either $Q' \xrightarrow{\tau} S'$ and $R' \xrightarrow{\bar{b}} S'$ for some $S'$, and we can let $S = S' \setminus L$. Or else we get $Q' \xrightarrow{b} R'$, hence $Q \xrightarrow{b} R$.

(*rel*): $P = P'[f]$ and $Q = Q'[f]$, where $P' \xrightarrow{\bar{c}} Q'$ and $\bar{b} = f\bar{c}$. Then $R = R'[f]$ and $P' \xrightarrow{\tau} R'$. By induction hypothesis, we get either $Q' \xrightarrow{\tau} S'$ and $R' \xrightarrow{\bar{c}} S'$ for some $S'$, and we can let $S = S'[f]$. Or else we get $Q' \xrightarrow{c} R'$, hence $Q \xrightarrow{b} R$.

(*rec*): $P = A$ where $A \stackrel{\text{def}}{=} P'$ and $P' \xrightarrow{\bar{b}} Q$. Since $A \xrightarrow{\tau} R$, we must also have $P' \xrightarrow{\tau} R$, and the claim follows by induction hypothesis. ☐

## 6.5 Example: The core join calculus

The join calculus was introduced by Fournet and Gonthier in [17] and further developed in [18]. It is a concurrent, message passing calculus like the $\pi$-calculus. However, the reaction rule is simpler and closer to the semantics of a chemical abstract machine. Moreover, the scoping rules of the join calculus are such that locality can be easily modeled. The full join calculus deals with a distributed system of locations, and it contains features that deal with such issues as migration and failure. Here, we will only be concerned with the *core* join calculus, which is the fragment of the join calculus that pertains to a single location.

Let $\mathcal{N}$ be a countable set of ***names***. We use $x, y, \ldots$ to denote names, and $\tilde{x}, \tilde{y}, \ldots$ to denote sequences of names. Core join calculus ***processes*** $P, Q, \ldots$ and ***rule*** $R, S, \ldots$ are given by the following grammars:

$$P ::= x\langle\tilde{y}\rangle \;\Big|\; P|Q \;\Big|\; \mathbf{def}\ R_1 \wedge \ldots \wedge R_m\ \mathbf{in}\ P \qquad R ::= x_1(\tilde{v}_1)|\ldots|x_n(\tilde{v}_n) \rhd P$$

A process of the form $x\langle\tilde{v}\rangle$ is called a ***message***. In the rule $R = x_1(\tilde{v}_1)|\ldots|x_n(\tilde{v}_n) \rhd P$, the names $\tilde{v}_1 \ldots \tilde{v}_n$ are bound, and they are assumed to be distinct. The names $x_1 \ldots x_n$ are called the ***defined names*** of $R$, denoted $dn(R)$. Finally, all of the defined names of $R_1, \ldots, R_m$ are bound in the process **def** $R_1 \wedge \ldots \wedge R_m$ **in** $P$. For a more comprehensive treatment, see [17, 18].

The semantics of the core join calculus is given in the style of a chemical abstract machine. A ***state*** $\Delta \vdash_N \Pi$ is a multiset $\Delta$ of rules together with a multiset $\Pi$ of processes. $N$ is a set of names, such that $fn(\Delta, \Pi) \subseteq N$. We identify states up to $\alpha$-equivalence, *i.e.* up to renaming of bound variables. The transitions of this machine follow a simple idea: the processes on the right hand side evolve according to the rules on the left-hand side. There are two kinds of transitions: ***structural*** transitions, denoted $\rightharpoonup$, and ***reactions***, denoted $\mapsto$. The transition rules are shown in Table 6.7. The rule $(join)$ is of course only applicable is the length of $\tilde{y}_i$ and $\tilde{v}_i$ are the same, for all $i$. Note that in the rule $(str2)$, the sets $N$ and $dn(R_1, \ldots, R_m)$ must be disjoint; this may necessitate renaming some bound variables in **def** $R_1 \wedge \ldots \wedge R_m$ **in** $P$.

*Remark.* In the original formulation of the join-calculus [17, 18], the structural rules are assumed to be reversible. We adopt a different convention here. Especially the inverse of rule $str2$ causes problems in our setting, as it allows a state under certain conditions to rename its free names.

To make make the join calculus into a labeled transition system with input and output, let $X = \{x\langle\tilde{y}\rangle \mid x \in \mathcal{N}, \tilde{y} \in \mathcal{N}^*\}$ be the set of messages. We add input and output transitions:

$$\begin{array}{lrcl} (in) & \Delta \vdash_N \Pi & \xrightarrow{\mathit{in}\,x\langle\tilde{y}\rangle} & \Delta \vdash_{N\cup\{x,\tilde{y}\}} \Pi, x\langle\tilde{y}\rangle \\ (out) & \Delta \vdash_N \Pi, x\langle\tilde{y}\rangle & \xrightarrow{\mathit{out}\,x\langle\tilde{y}\rangle} & \Delta \vdash_N \Pi \end{array}$$

Further, we let $\xrightarrow{\tau} \;=\; \rightharpoonup \cup \mapsto$. With these definitions, the join calculus defines a labeled transition system $\mathbf{S}_{\text{join}} \colon X \to X$.

**Theorem 6.24.** *The labeled transition system* $\mathbf{S}_{\text{join}}$ *defined by the core join calculus is out-buffered with feedback.*

## 6.6 Other characterizations of asynchrony

In Sections 6.2 and 6.3, we have characterized notions of asynchrony by first-order axioms *up to weak bisimulation*. It is possible to remove the words "up to weak bisimulation", *i.e.* to characterize asynchrony directly. This happens at the cost of introducing second-order axioms. The shift to second-order seems to be inevitable, since weak bisimulation itself is a second-order notion.

### 6.6.1 Out-buffered agents

Consider the two different output transitions in

$$\mathbf{S} = \begin{tikzcd} s \arrow[r, "\tau"'] \arrow[rr, bend left, "\mathit{out}\,y"] \arrow[d, "\mathit{in}\,x"'] & t \arrow[r, "\mathit{out}\,y"'] & u. \\ v \end{tikzcd}$$

The transition $s \xrightarrow{\mathit{out}\,y} u$ has the implicit effect of disabling the action $\mathit{in}\,x$. The transition $t \xrightarrow{\mathit{out}\,y} u$ has no such side effect. Roughly, out-bufferedness is characterized by the fact that every output transition $\xrightarrow{\mathit{out}\,y}$ factors into a silent part $\overset{\tau}{\Rightarrow}$ and a part $\overset{\mathit{out}\,y}{\approx\!\succ}$ without side effects.

The second-order axioms for out-buffered agents are given in Table 6.8. A state $s$ in an LTS $\mathbf{S}$ is ***reachable*** if there exist transitions $s_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} s$ from the initial state $s_0$. If $\mathbf{S} \approx \mathbf{T}$, then for every reachable $s \in \mathbf{S}$, there is reachable $t \in \mathbf{T}$ with $s \approx t$.

**Theorem 6.25.** *An agent* $\mathbf{S} \colon X \to_B Y$ *is out-buffered if and only if there exists a binary relation* $\overset{\mathit{out}\,y}{\approx\!\succ} \subseteq |\mathbf{S}| \times |\mathbf{S}|$ *for each* $y \in Y$*, satisfying* (OB1*)*–*(OB5*)*.*

*Proof.* $\Rightarrow$: Suppose $\mathbf{S}$ is out-buffered. By Theorem 6.10, $\mathbf{S} \approx \mathbf{T}$ for some $\mathbf{T}$ satisfying (OB1)–(OB3). For $s, t \in |\mathbf{S}|$, define $s \overset{\mathit{out}\,y}{\approx\!\succ} t$ iff there exist $s', t' \in |\mathbf{T}|$ with $s \approx s' \xrightarrow{\mathit{out}\,y} t' \approx t$. It is easy to verify that $\overset{\mathit{out}\,y}{\approx\!\succ}$ satisfies (OB1*)–(OB5*).

$\Leftarrow$: Suppose $\mathbf{S}$ satisfies (OB1*)–(OB5*). Notice that if a relation $\overset{\mathit{out}\,y}{\approx\!\succ}$ satisfies (OB1*)–(OB5*), then so does $\approx \circ \overset{\mathit{out}\,y}{\approx\!\succ} \circ \approx$. Hence assume without loss of generality that $\overset{\mathit{out}\,y}{\approx\!\succ}$ is invariant under weak bisimulation. For any sequence $w = y_1 y_2 \cdots y_n \in Y^*$, write $s \overset{\mathit{out}\,w}{\approx\!\succ} t$ if $s \overset{\mathit{out}\,y_1}{\approx\!\succ} \overset{\mathit{out}\,y_2}{\approx\!\succ} \cdots \overset{\mathit{out}\,y_n}{\approx\!\succ} \approx t$. Note that in the case $n = 0$ this

Table 6.8: Second-order axioms for out-buffered agents

$$\begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] \arrow[d, Rightarrow, "\alpha"'] & t \\ s' & \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] \arrow[d, Rightarrow, "\alpha"'] & t \arrow[d, Rightarrow, "\alpha"] \\ s' \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] & t' \end{tikzcd}$$
where $\alpha \neq \mathit{out}\,y$

(OB1*)

$$\begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] & t \arrow[d, Rightarrow, "\alpha"] \\ & t' \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] \arrow[d, Rightarrow, "\alpha"'] & t \arrow[d, Rightarrow, "\alpha"] \\ s' \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] & t' \end{tikzcd}$$
where $\alpha \neq \mathit{out}\,y$

(OB2*)

$$\begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] \arrow[d, Rightarrow, "\mathit{out}\,y"'] & t \\ s' & \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} s \arrow[r, phantom, "\overset{\mathit{out}\,y}{\approx\!\succ}"] \arrow[d, Rightarrow, "\mathit{out}\,y"'] & t \arrow[d, Rightarrow, "\tau"] \\ s' \arrow[r, phantom, "\approx"] & t' \end{tikzcd}$$

(OB3*)

$$s \overset{\mathit{out}\,y}{\approx\!\succ} t \quad \Rightarrow \quad s \overset{\mathit{out}\,y}{\Longrightarrow} \approx t$$

(OB4*)

$$s \xrightarrow{\mathit{out}\,y} t \quad \Rightarrow \quad s \overset{\tau}{\Rightarrow} \overset{\mathit{out}\,y}{\approx\!\succ} t$$
where $s$ reachable

(OB5*)

means $s \approx t$. Consider the relation $R \subseteq |\mathbf{S}| \times |\mathbf{S}; \mathcal{B}|$ defined by $R = \{\langle s, \langle t, w\rangle\rangle \mid s \overset{\mathit{out}\,w}{\approx\!\succ} t \text{ and } t \text{ reachable}\}$. Clearly, $R$ relates initial states: $s_0 R \langle s_0, \emptyset\rangle$. We show that $R$ is a weak bisimulation. Suppose

$$\begin{tikzcd} s \arrow[r, phantom, "R"] \arrow[d, "\alpha"'] & \langle t, w\rangle \\ s', & \end{tikzcd}$$

where $w = y_1 \cdots y_n$.

**Case 1:** $\alpha$ is $\mathit{out}\,y_i$ for some $1 \leqslant i \leqslant n$. Take the minimal such $i$. Then

$$\begin{tikzcd}[column sep=small] s \arrow[r, phantom, "\overset{\mathit{out}\,y_1}{\approx\!\succ}"] \arrow[d, "\mathit{out}\,y_i"'] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_{i-1}}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\overset{\mathit{out}\,y_i}{\approx\!\succ}"] \arrow[d, Rightarrow, "\mathit{out}\,y_i"'] & \bullet \arrow[r, phantom, "\overset{\mathit{out}\,y_{i+1}}{\approx\!\succ}"] \arrow[d, Rightarrow, "\tau"'] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_n}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\approx"] \arrow[d, Rightarrow, "\tau"'] & t \arrow[d, Rightarrow, "\tau"'] \\ s' \arrow[r, phantom, "\overset{\mathit{out}\,y_1}{\approx\!\succ}"] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_{i-1}}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\approx"] & \bullet \arrow[r, phantom, "\overset{\mathit{out}\,y_{i+1}}{\approx\!\succ}"] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_n}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\approx"] & t' \end{tikzcd}$$

by (OB1*) and (OB3*). With $w' = y_1 \cdots y_{i-1} y_{i+1} \cdots y_n$ we hence have $s' R \langle t', w'\rangle$, and also $\langle t, w\rangle \overset{\mathit{out}\,y_i}{\Longrightarrow} \langle t', w'\rangle$.✓

**Case 2:** $\alpha \neq \mathit{out}\,y_i$ for all $i$. From $s \xrightarrow{\alpha} s'$ and $s \overset{\mathit{out}\,w}{\approx\!\succ} t$, by repeated application of (OB3*), we get $s' \overset{\mathit{out}\,w}{\approx\!\succ} t'$ and $t \overset{\alpha}{\Rightarrow} t'$ for some $t'$, hence $s' R \langle t', w\rangle$ and $\langle t, w\rangle \overset{\alpha}{\Rightarrow} \langle t', w\rangle$.✓

Now suppose

$$\begin{tikzcd} s \arrow[r, phantom, "R"] & \langle t, w\rangle \arrow[d, "\alpha"] \\ & \langle t', w'\rangle. \end{tikzcd}$$

We distinguish three cases for $\langle t, w\rangle \xrightarrow{\alpha} \langle t', w'\rangle$ by Definition 6.2:

**Case 1:** $t \xrightarrow{\alpha} t'$, $w = w'$ and $\alpha$ not output. Then $s \overset{\mathit{out}\,w}{\approx\!\succ} t \xrightarrow{\alpha} t'$ implies $s \overset{\alpha}{\Rightarrow} s' \overset{\mathit{out}\,w}{\approx\!\succ} t'$ by repeated application of (OB2*), *i.e.* $s \overset{\alpha}{\Rightarrow} s' R \langle t', w\rangle$.✓

**Case 2:** $t = t'$, $w \xrightarrow{\alpha} w'$ and $\alpha$ not input. If $w = y_1 \cdots y_n$, then $\alpha = \mathit{out}\,y_i$ for some $1 \leqslant i \leqslant n$. Let $i$ be the minimal such index. Then

$$\begin{tikzcd}[column sep=small] s \arrow[r, phantom, "\overset{\mathit{out}\,y_1}{\approx\!\succ}"] \arrow[d, Rightarrow, "\mathit{out}\,y"'] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_{i-1}}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\overset{\mathit{out}\,y_i}{\approx\!\succ}"] \arrow[d, Rightarrow, "\mathit{out}\,y"'] & \bullet \arrow[r, phantom, "\overset{\mathit{out}\,y_{i+1}}{\approx\!\succ}"] \arrow[d, equal] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_n}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\approx"] & t \\ s' \arrow[r, phantom, "\overset{\mathit{out}\,y_1}{\approx\!\succ}"] & \cdots \arrow[r, phantom, "\overset{\mathit{out}\,y_{i-1}}{\approx\!\succ}"] & \bullet \arrow[r, phantom, "\approx"] & \bullet & & & \end{tikzcd}$$

by (OB4*) and (OB2*), hence $s \overset{\mathit{out}\,y}{\Longrightarrow} s' R \langle t, w'\rangle$.✓

**Case 3:** $t \xrightarrow{\mathit{out}\,y} t'$, $w \xrightarrow{\mathit{in}\,y} w'$ and $\alpha = \tau$. Then $w' = wy$. By (OB5*), since $t$ is reachable, there is $t''$ with $t \overset{\tau}{\Rightarrow} t'' \overset{\mathit{out}\,y}{\approx\!\succ} t'$. Then $s \overset{\mathit{out}\,w}{\approx\!\succ} t$ and repeated application of (OB2*) give $s \overset{\tau}{\Rightarrow} s' \overset{\mathit{out}\,w}{\approx\!\succ} t'' \overset{\mathit{out}\,y}{\approx\!\succ} t'$, hence $sR\langle t', w'\rangle$.✓ □

*Remark.* Notice that Principle 6.1 can be applied to obtain a unique maximal relation $\overset{\mathit{out}\,y}{\approx\!\succ}$, for every $y$, satisfying (OB1*)–(OB4*). Thus, $\mathbf{S}$ is out-buffered if this unique relation also satisfies (OB5*). Notice in particular how (OB1*) and (OB2*) resemble the definition of weak bisimulation; one may think of the relation $\overset{\mathit{out}\,y}{\approx\!\succ}$ as a weak bisimulation up to a suspended output.

### 6.6.2 In-buffered agents

The second-order axioms for in-buffered agents are given in Table 6.9. This is similar to the axioms for out-buffered agents, but notice that there is no analogue to (OB2*). This reflects the fact that unlike output transitions, input transitions can *enable*, but not *disable* other transitions.

**Theorem 6.26.** *An agent $\mathbf{S}\colon X \to_B Y$ is in-buffered if and only if there exists a binary relation $\overset{\mathit{in}\,x}{\approx\!\succ}$ for each $x \in X$, satisfying (IB1*)–(IB4*).*

*Proof.* $\Rightarrow$: As in the proof of Theorem 6.25.

$\Leftarrow$: Suppose $\mathbf{S}$ satisfies (IB1*)–(IB4*). Again, we can without loss of generality assume that $\overset{\mathit{in}\,x}{\approx\!\succ}$ is invariant under weak bisimulation. For any sequence $w = x_1x_2\cdots x_n \in X^*$, write $s \overset{\mathit{in}\,w}{\approx\!\succ} t$ if $s \overset{\mathit{in}\,x_1}{\approx\!\succ}\overset{\mathit{in}\,x_2}{\approx\!\succ} \cdots \overset{\mathit{in}\,x_n}{\approx\!\succ}\approx t$ $(n \geqslant 0)$. Consider the relation $R \subseteq |\mathcal{B};\mathbf{S}| \times |\mathbf{S}|$ defined by $R = \{\langle\langle w, s\rangle, t\rangle \mid s \overset{\mathit{in}\,w}{\approx\!\succ} t \text{ and } t \text{ reachable}\}$. Notice that $R$ relates initial states: $\langle\emptyset, s_0\rangle R s_0$. To see that $R$ is a weak bisimulation, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ & & \downarrow{\scriptstyle\alpha} \\ & & t', \end{array}$$

where $w = x_1\cdots x_n$. From $s \overset{\mathit{in}\,w}{\approx\!\succ} t$, with (IB3*) and weak bisimulation we get $s \overset{\mathit{in}\,w}{\Longrightarrow} s' \approx t$, hence $s' \overset{\alpha}{\Rightarrow} s''$ for some $s'' \approx t'$. Consequently $\langle w, s\rangle \overset{\tau}{\Rightarrow} \langle\emptyset, s'\rangle \overset{\alpha}{\Rightarrow} \langle\emptyset, s''\rangle R t'$. Conversely, suppose

$$\begin{array}{ccc} \langle w, s\rangle & R & t \\ {\scriptstyle\alpha}\downarrow & & \\ \langle w', s'\rangle. & & \end{array}$$

Again, we distinguish three cases:

**Case 1:** $s = s'$, $w \xrightarrow{\alpha} w'$ and $\alpha$ not output. Then $\alpha = \mathit{in}\,x$ and $w' = wx$ for some $x \in X$. By (IB4*), $t \overset{\mathit{in}\,x}{\approx\!\succ} t''$ for some $t''$, and by (IB3*), $t \overset{\mathit{in}\,x}{\Longrightarrow} t' \approx t''$, hence also $t \overset{\mathit{in}\,x}{\approx\!\succ} t'$, and we get $s \overset{\mathit{in}\,w}{\approx\!\succ} t \overset{\mathit{in}\,x}{\approx\!\succ} t'$, *i.e.* $\langle w', s\rangle R t'$ and $t \overset{\mathit{in}\,x}{\Longrightarrow} t'$.✓

**Case 2:** $s \xrightarrow{\alpha} s'$, $w = w'$ and $\alpha$ not input. From $s \overset{\mathit{in}\,w}{\approx\!\succ} t$ by repeated application of (IB1*), we get $t \overset{\alpha}{\Rightarrow} t'$ and $s' \overset{\mathit{in}\,w}{\approx\!\succ} t'$, *i.e.* $\langle w, s'\rangle R t'$.✓

**Case 3:** $w \xrightarrow{\mathit{out}\,x} w'$, $s \xrightarrow{\mathit{in}\,x} s'$ and $\alpha = \tau$. If $w = x_1x_2\cdots x_n$, then $x$ must be $x_i$ for some $1 \leqslant i \leqslant n$. Let such $i$ be minimal and construct

$$\begin{array}{ccccccccccccccc} s & \overset{\mathit{in}\,x_1}{\approx\!\succ} & \cdots & \overset{\mathit{in}\,x_{i-1}}{\approx\!\succ} & \bullet & \overset{\mathit{in}\,x_i}{\approx\!\succ} & \bullet & \overset{\mathit{in}\,x_{i+1}}{\approx\!\succ} & \cdots & \overset{\mathit{in}\,x_n}{\approx\!\succ} & \bullet & \approx & t \\ {\scriptstyle\mathit{in}\,x}\downarrow & & & & {\scriptstyle\mathit{in}\,x}\Downarrow & & {\scriptstyle\tau}\Downarrow & & & & {\scriptstyle\tau}\Downarrow & & {\scriptstyle\tau}\Downarrow \\ s' & \overset{\mathit{in}\,x_1}{\approx\!\succ} & \cdots & \overset{\mathit{in}\,x_{i-1}}{\approx\!\succ} & \bullet & \approx & \bullet & \overset{\mathit{in}\,x_{i+1}}{\approx\!\succ} & \cdots & \overset{\mathit{in}\,x_n}{\approx\!\succ} & \bullet & \approx & t' \end{array}$$

by (IB1*) and (IB2*). This shows $\langle s', w'\rangle R t'$.✓ □

Table 6.9: Second-order axioms for in-buffered agents

$$\begin{array}{ccc} s & \overset{in\,x}{\approx\!\succ} & t \\ {\scriptstyle\alpha}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{in\,x}{\approx\!\succ} & t \\ {\scriptstyle\alpha}\Downarrow & & \Downarrow{\scriptstyle\alpha} \\ s' & \overset{in\,x}{\approx\!\succ} & t' \end{array}$$

where $\alpha \neq in\,x$

(IB1*)

$$\begin{array}{ccc} s & \overset{in\,x}{\approx\!\succ} & t \\ {\scriptstyle in\,x}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{in\,x}{\approx\!\succ} & t \\ {\scriptstyle in\,x}\Downarrow & & \Downarrow{\scriptstyle\tau} \\ s' & \approx & t' \end{array}$$

(IB2*)

$$s \overset{in\,x}{\approx\!\succ} t \quad\Rightarrow\quad s \overset{in\,x}{\Longrightarrow} s' \approx t$$

(IB3*)

$$s \quad\Rightarrow\quad s \overset{in\,x}{\approx\!\succ} s'$$

where $s$ reachable

(IB4*)

Table 6.10: Second-order axioms for out-queued agents

$$\begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ {\scriptstyle\alpha}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ {\scriptstyle\alpha}\Downarrow & & \Downarrow{\scriptstyle\alpha} \\ s' & \overset{out\,y}{\approx\!\ggcurly} & t' \end{array}$$

where $\alpha$ not output

(OQ1*)

$$\begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ & & \Downarrow{\scriptstyle\alpha} \\ & & t' \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ {\scriptstyle\alpha}\Downarrow & & \Downarrow{\scriptstyle\alpha} \\ s' & \overset{out\,y}{\approx\!\ggcurly} & t' \end{array}$$

where $\alpha$ not output

(OQ2*)

$$\begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ {\scriptstyle out\,z}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow y = z \text{ and } \quad \begin{array}{ccc} s & \overset{out\,y}{\approx\!\ggcurly} & t \\ {\scriptstyle out\,z}\Downarrow & & \Downarrow{\scriptstyle\tau} \\ s' & \approx & t' \end{array}$$

(OQ3*)

$$s \overset{out\,y}{\approx\!\ggcurly} t \quad\Rightarrow\quad s \overset{out\,y}{\Longrightarrow} \approx t$$

(OQ4*)

$$s \xrightarrow{out\,y} t \quad\Rightarrow\quad s \overset{\tau}{\Rightarrow} \overset{out\,y}{\approx\!\ggcurly} t$$

where $s$ reachable

(OQ5*)

Table 6.11: Second-order axioms for in-queued agents

$$\begin{array}{ccc} s & \overset{in\,x}{\approx\!\ggcurly} & t \\ {\scriptstyle\alpha}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{in\,x}{\approx\!\ggcurly} & t \\ {\scriptstyle\alpha}\Downarrow & & \Downarrow{\scriptstyle\alpha} \\ s' & \overset{in\,x}{\approx\!\ggcurly} & t' \end{array}$$

where $\alpha$ not input

(IQ1*)

$$\begin{array}{ccc} s & \overset{in\,x}{\approx\!\ggcurly} & t \\ {\scriptstyle in\,x}\Downarrow & & \\ s' & & \end{array} \quad\Rightarrow\quad \begin{array}{ccc} s & \overset{in\,x}{\approx\!\ggcurly} & t \\ {\scriptstyle in\,x}\Downarrow & & \Downarrow{\scriptstyle\tau} \\ s' & \approx & t' \end{array}$$

(IQ2*)

$$s \overset{in\,x}{\approx\!\ggcurly} t \quad\Rightarrow\quad s \overset{in\,x}{\Longrightarrow} \approx t$$

(IQ3*)

$$s \quad\Rightarrow\quad s \overset{in\,x}{\approx\!\ggcurly} s'$$

where $s$ reachable

(IQ4*)

### 6.6.3 Out-queued and in-queued agents

The second-order axioms for out- and in-queued agents are given in Tables 6.10 and 6.11, respectively. Notice that the only difference to the buffered case are the side conditions.

**Theorem 6.27.** *An agent* $\mathbf{S}\colon X \to_B Y$ *is out-queued if and only if there are relations* $\overset{out\, y}{\approx\!\!\twoheadrightarrow}$ *satisfying* ($\mathrm{OQ}1^*$)–($\mathrm{OQ}5^*$). $\mathbf{S}$ *is in-queued if and only if there are relations* $\overset{in\, x}{\approx\!\!\twoheadrightarrow}$ *satisfying* ($\mathrm{IQ}1^*$)–($\mathrm{IQ}4^*$). ☐

# Bibliography


[1] S. Abramsky. Interaction categories and communicating sequential processes. In A. W. Roscoe, editor, *A Classical Mind: Essays in honour of C. A. R. Hoare*, pages 1–16. Prentice Hall International, 1994.

[2] S. Abramsky, S. Gay, and R. Nagarajan. Interaction categories and typed concurrent programming. In *Proceedings of the 1994 Marktoberdorf Summer School*. Springer, 1994.

[3] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994.

[4] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous $\pi$-calculus. In *Proc. CONCUR '96*, Springer LNCS 1119, pages 147–162, 1996.

[5] H. P. Barendregt. *The Lambda Calculus, its Syntax and Semantics*. North-Holland, 2nd edition, 1984.

[6] M. A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, 1988.

[7] G. Berry. Stable models of typed $\lambda$-calculi. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming*, Springer LNCS 62, pages 72–89, 1978.

[8] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[9] G. Boudol. Asynchrony and the $\pi$-calculus. Technical Report 1702, INRIA, Sophia-Antipolis, 1992.

[10] K. B. Bruce and A. R. Meyer. The semantics of second-order polymorphic lambda calculus. In G. Kahn, D. B. MacQueen, and G. Plotkin, editors, *Proc. Conf. on Semantics of Data Types, Sophia-Antipolis, 1984*, Springer LNCS 173, pages 131–144, 1984.

[11] S. Bulman-Fleming and W. Taylor. Union-indecomposable varieties. *Colloquium Mathematicum*, 35:189–199, 1976.

[12] A. Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39:472–482, 1936.

[13] P. M. Cohn. *Universal Algebra,* Revised Edition. D. Reidel Publishing, Holland, 1981.

[14] D. Čubrić. Embedding of a free cartesian closed category into the category of sets. *Journal of Pure and Applied Algebra*, 1995.

[15] D. Čubrić. On the semantics of the universal quantifier. *Annals of Pure and Applied Logic*, 1997. To appear.

[16] P. Di Gianantonio, F. Honsell, S. Liani, and G. D. Plotkin. Countable non-determinism and uncountable limits. In *Proceedings of CONCUR '94*, Springer LNCS 836, 1994. See also: Uncountable limits and the Lambda Calculus, *Nordic Journal of Computing* 2, 1995.

[17] C. Fournet and G. Gonthier. The reflexive cham and the join-calculus. In *POPL '96: Proceedings of the 23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1996.

[18] C. Fournet, G. Gonthier, J.-J. Levy, L. Maranget, and D. Remy. A calculus of mobile agents. In *Proceedings of CONCUR '96*, Springer LNCS 1119, pages 406–421, 1996.

[19] P. J. Freyd. Combinators. In *Proc. Mathematical Applications to Computer Science*. American Mathematical Society, 1989.

[20] P. J. Freyd and A. Scedrov. *Categories, Allegories*. North-Holland, 1989.

[21] H. Friedman. Equality between functionals. In R. Parikh, editor, *Proceedings of the Logic Colloquium '73*, Springer Lecture Notes in Mathematics 453, pages 22–37, 1975.

[22] J.-Y. Girard. Une extension de l'interpretation de Gödel à l'analyse, et son application a l'elimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 63–92. North-Holland, 1971.

[23] J.-Y. Girard. The system $F$ of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[24] G. Grätzer. *Universal Algebra*. D. Van Nostrand, 1968.

[25] J. Hagemann and A. Mitschke. On $n$-permutable congruences. *Algebra Universalis*, 3:8–12, 1973.

[26] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proc. ECOOP 91, Geneve*, 1991.

[27] F. Honsell and S. Ronchi Della Rocca. An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus. *Journal of Computer and System Sciences*, 45(1), 1992.

[28] G. Huet, editor. *Logical Foundations of Functional Programming*. Addison-Wesley, 1990.

[29] M. Hyland. A syntactic characterization of the equality in some models for the lambda calculus. *J. London Math. Soc.*, 12:361–370, 1976.

[30] B. Jacobs, I. Margaria, and M. Zacchi. Filter models with polymorphic types. *Theoretical Computer Science*, 95:143–158, 1992.

[31] C. P. J. Koymans. Models of the lambda calculus. *Information and Control*, 52:306–332, 1982.

[32] J.-L. Krivine. *Lambda-calculus, types and models*. Masson, 1993.

[33] J. Lambek. From $\lambda$-calculus to cartesian closed categories. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 375–402. Academic Press, London, New York, 1980.

[34] J. Lambek and P. J. Scott. *An Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics 7. Cambridge University Press, New York, 1986.

[35] N. A. Lynch and E. W. Stark. A proof of the Kahn principle for input/output automata. *Information and Computation*, 82:81–92, 1989.

[36] S. MacLane. *Categories for the Working Mathematician*. Springer GTM 5. 1971.

[37] A. I. Mal'cev. K obščeĭ teorii algebraičeskih sistem. *Mat. Sb. N. S. 35 (77)*, pages 3–20, 1954.

[38] A. R. Meyer. What is a model of the lambda calculus? *Information and Control*, 52:87–122, 1982.

[39] A. R. Meyer, J. C. Mitchell, E. Moggi, and R. Statman. Empty types in polymorphic lambda calculus. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, pages 253–262, 1987. Reprinted in [28].

[40] R. Milner. *A Calculus of Communicationg Systems*. Springer LNCS 92. 1980.

[41] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. Elsevier, 1990.

[42] J. C. Mitchell and E. Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51:99–124, 1991.

[43] J. Myhill and R. Flagg. A type-free system extending (ZFC). *Annals of Pure and Applied Logic 43*, pages 79–97, 1989.

[44] U. Nestmann and B. C. Pierce. Decoding choice encodings. In *Proceedings of CONCUR '96*, Springer LNCS 1119, pages 179–194, 1996.

[45] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous $\pi$-calculus. In *POPL '97: Proceedings of the 24th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (Paris)*, 1997.

[46] G. D. Plotkin. The $\lambda$-calculus is $\omega$-incomplete. *The Journal of Symbolic Logic*, 39:313–317, 1974.

[47] G. D. Plotkin. *Domains*. Department of Computer Science, University of Edinburgh, 1983.

[48] G. D. Plotkin. Set-theoretical and other models of the lambda-calculus. *Theoretical Computer Science*, 121:351–409, 1993.

[49] G. D. Plotkin. A semantics for static type inference. *Information and Computation*, 109:256–299, 1994.

[50] G. D. Plotkin. On a question of H. Friedman. *Information and Computation*, 126(1):74–77, 1996.

[51] J. C. Reynolds. Towards a theory of type structure. In *Proceedings, Colloque sur la Programmation*, Springer LNCS 19, pages 408–425, 1974.

[52] J. C. Reynolds. Polymorphism is not set-theoretic. In *International Symposium on Semantics of Data Types*, Springer LNCS 173, pages 145–156, 1984.

[53] D. S. Scott. Continuous lattices, toposes, algebraic geometry and logic. In F. W. Lawvere, editor, *Proc. 1971 Dalhousie Conference*, Springer Lecture Notes in Mathematics 274, pages 97–136, 1972.

[54] D. S. Scott. Relating theories of the $\lambda$-calculus. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 403–450. Academic Press, London, New York, 1980.

[55] R. A. G. Seely. Hyperdoctrines, natural deduction and the Beck condition. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 29:505–542, 1983.

[56] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *The Journal of Symbolic Logic*, 52(4), Dec. 1987.

[57] P. Selinger. Order-incompleteness and finite lambda models. Extended abstract. In *Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science*, pages 432–439, 1996.

[58] P. Selinger. First-order axioms for asynchrony. In *Proceedings of CONCUR '97*, Springer LNCS, 1997. To appear.

[59] M. W. Shields. Concurrent machines. *Theoretical Computer Science*, 28:449–465, 1985.

[60] A. K. Simpson. Categorical completeness results for the simply-typed lambda-calculus. In *Proc. TLCA '95*, Springer LNCS 902, pages 414–427, 1995.

[61] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, 11(4):761–783, 1982.

[62] C. Strachey. Fundamental concepts in programming languages. Unpublished lecture notes, International Summer School in Computer Programming, Copenhagen, Aug. 1967.

[63] W. Taylor. Structures incompatible with varieties, Abstract 74T-A224. *Notices of the American Mathematical Society*, 21:A-529, 1974.

[64] C. P. Wadsworth. The relation between computational and denotational properties for Scott's $D_\infty$-models of the lambda-calculus. *SIAM Journal on Computing*, 5:488–521, 1976.

# Index


absolute interpretation
  in an algebra, 27, 39
  of the lambda calculus, 26
absolutely unorderable algebra, 39
action, 84
  in CCS, 97
  input and output, 85
  internal, 85
  silent or unobservable, 84
adjunction, 7
agent, 85
  buffer $\mathcal{B}$, 87
  buffered, 87
  composition with function, 93
  domain extension and restriction, 93
  feedback, 94
  hiding, 93
  in-buffered, 87
  in-queued, 87
  input and output action, 85
  internal action, 85
  isomorphism of, 86
  operations on, 93
  out-buffered, 87
  out-buffered with feedback, 94
  out-queued, 87
  output action, 85
  parallel composition
    with interaction, 94
    without interaction, 93
  queue $\mathcal{Q}$, 87
  queued, 87
  renaming, 93
  self-composition, 94
  sequential composition, 85
  silent action, 84
  symmetric monoidal structure, 93
algebra, 13
  dcpo-algebra, 17, 41
  free, 14
  ordered, 16, 40
  polynomial, 15
  quotient, 13
  term algebra, 14
algebraic signature, 13
algebraic variety, 14
  of combinatory algebras, 22
  of lambda algebras, 24
$\forall$-diagram, 73
$\alpha$-equivalence, 20
antisymmetry, 10
applicative structure, 22
  extensional, 30
  order-extensional, 49
  ordered, 47
  partial, 51
  unorderable, 36
arity, 13
asynchronous CCS, 96
asynchrony, 83, 87

backwards output-determinacy, 95
base category, 72
Beck-Chevalley condition, 73
Berry order, 11
$\beta$-conversion, 20
$\beta$-reduction, 21
bilimit, 12
binary product, 8
bisimulation
  strong, 84
  weak, 84
bound name, in join calculus, 99
bound variable, in lambda calculus, 20, 64, 78
bound, upper and lower, 10
bounded complete cpo, 11
bounded tree, 50
buffer $\mathcal{B}$, 87
buffered agent, 87

calculus of communicating systems, 96
cartesian-closed category, *see* ccc
categorical model
  of conversion, 30
  of reduction, 48
categories

equivalence of, 7
category, 5
cartesian-closed, *see* ccc
cocomplete, 8
complete, 8
discrete, 5
dominated by a poset, 64
dual, 5
functor category, 7
of complete partial orders **CPO**, 11
of directed complete partial orders **DCPO**, 11
of non-empty sets $\mathscr{S}^+$, 62
of presheaves $\mathscr{S}^{\mathbf{C}^{op}}$, 9
of presheaves over a poset $\mathscr{S}^P$, 63
of sets $\mathscr{S}$, 5, 62
$PL$-category, 72
small, 5
standard structure $\widetilde{\mathbf{D}}$, 75
ccc, 9
associated to a theory, 67
congruence, 58
free, 67
representation, 9
Henkin representation, 58
special, 62, 70
CCS, 96
chain, 11
channel
asynchronous, 83, 87
synchronous, 83
Church numerals, 21
Church-Rosser property, 21
closed combinatory term, 22
closed lambda term, 20
closed term algebra, 22, 36
co-name, in CCS, 97
co-unit of adjunction, 7
cocomplete category, 8
cocone, 8
codomain of a morphism, 5
colimit, 8
collectively monic, 6
combinator, 22
combinatory algebra, 22
homomorphism of, 22
unorderable, 36
valuation in, 22
combinatory completeness, 23
combinatory logic, 22
derived lambda abstractor, 23
combinatory term, 22
communication
asynchronous, 83, 87
synchronous, 83
commutativity
input, 90
output, 89
compatible elements, 10, 38, 50
compatible relation, 13
preorder, 36, 39
complete category, 8
complete lattice, 10
complete partial order, 11
bounded complete, 11
meet cpo, 11
composition, 5
agent and function, 93
parallel
with interaction, 94
without interaction, 93
sequential, 85, 94
cone, 7
collectively monic, 6
limiting, 8
confluence
Church-Rosser, 21
input, 90
output, 89
congruence
on algebra, 13
on ccc, 58
on $PL$-category, 74
consistency
of the lambda calculus, 22
of the Mal'cev axioms, 42
constant, 20
individual, 64, 78
process, 97
type, 64
continuity
$\omega$-continuity, 11
Scott-continuity, 11
continuous functor, 12
locally, 13
continuously complete model, 38
contravariant functor, 6
conversion, 20
categorical model of, 30
syntactical model of, 48
core join calculus, 99
core of an extended theory, 70
covariant functor, 6
cover, 60

split, 60
cpo, 11
bounded complete, 11
meet cpo, 11
Curry algebra, 30
Curry axioms for lambda algebras, 24
currying, 9

$D_\infty$-model, 13, 54
dcpo, 11
dcpo-algebra, 17, 41
dcpo-variety, 17, 41
defined name, in join calculus, 99
derived lambda abstractor, 23
determinacy
input, 90
output, 89
backwards, 95
weak, 98
diagonal axiom, 42
diagram, 7
$\forall$-diagram, 73
binary product, 8
exponential, 9
limit of, 8
partial $\forall$-diagram, 74
partial exponential, 59
product, 8
diamond property, 21
directed complete partial order, 11
directed equality, 51
directed poset, 11
discrete category, 5
discrete preorder, 36
domain, 12
domain equations, 12
domain extension and restriction, 93
domain of a morphism, 5
dominated category, 64
downdeal, 10
downward closed set, 10
dual category, 5
dummy functor, 73, 74

embedding
Henkin, 62–71
Henkin-*PL*, 75
of categories, 7
Yoneda, 9
embedding-projection pair, 12
emptiness assertion, 69
empty types, 68, 69, 82
epic, 6
split, 6
epimorphism, 6
equalizer, 8
equation
defining CCS process, 97
extended, 70
in algebra, 14
inequation, 16
of polymorphic lambda calculus, 78
of simply-typed lambda calculus, 65
equivalence of categories, 7
$\eta$-conversion, 20
$\eta$-reduction, 21
expanding sequence, 12
exponential diagram, 9
extended equation, 70
extended theory, 70
core of, 70
principal, 70
extensionality, 30
order, 49
strong, 50
weak, 29

faithful functor, 7
faithful subcategory, 7
feedback, 94
fiber, 72
finitely separable lambda algebra, 36
flat poset, 10
free
algebra, 14
ccc, 67
dcpo-algebra, 17
ordered algebra, 16, 40
variable, 20, 64, 78
full functor, 7
full subcategory, 7
function
continuous, 11
monotone, 10
stable, 11
function symbol, 13
functor, 6
adjoint, 7
ccc-representation, 9
continuous, 12
contravariant, 6
covariant, 6
embedding, 7
faithful, 7

full, 7
Henkin representation, 58
Henkin-$PL$-representation, 74
inclusion, 7
kernel, 58
left-full, 63
locally continuous, 13
$PL$-representation, 73
representable, 7
Yoneda, 9
functor category, 7

generalized Mal'cev operators, 40
greatest lower bound, 10
guard in CCS, 97

height of a bounded tree, 50
Henkin natural transformation, 74
Henkin representation, 58
in $\mathscr{S}$, 62
in $\mathscr{S}^P$, 63
in $\mathscr{S}^+$, 62
Henkin-$PL$-embedding, 75
Henkin-$PL$-representation, 74
in $\widetilde{\mathscr{S}^P}$, 77
in $\widetilde{\mathscr{S}^+}$, 77
hiding, 93
hom-set, 5
homomorphism, 13

ideal completion, 41
ideal in a poset, 41
identity morphism, 5
in-buffered agent, 87
first-order axioms for, 90
second-order axioms for, 103
in-queued agent, 87
first-order axioms for, 90
second-order axioms for, 103
inclusion functor, 7
indeterminate, 15
indiscrete preorder, 36
individual constant, 64, 78
individual variable, 78
inequation, 16
infimum, 10
initial state of labeled transition system, 84
input actions, 85
input and output
labeled transition system with, *see* agent
input-commutativity, 90
input-confluence, 90
input-determinacy, 90
input-receptivity, 90
internal actions, 85
interpretation
in ccc, 65
non-strict, 67
in $PL$-category, 79
non-strict, 81
inverse, 6
iso, 6
isomorphism, 6
natural, 7
of labeled transition systems, 86

join and meet, 10
join calculus, 99

kernel
of a functor, 58
of a Henkin representation, 58
of a Henkin-$PL$-representation, 74
of a homomorphism, 13
of a $PL$-representation, 74
Kleene equality, 52
Kripke lambda model
polymorphic, 82
simply-typed, 72

labeled transition system, 84
with input and output, *see* agent
lambda algebra, 24
and reflexive ccc models, 31
finitely separable, 36
homomorphism of, 24
soundness and completeness, 28
soundness of ($\xi$)-rule, 27
lambda calculus
absolute interpretation, 26
closed term algebra, 22
consistency, 22
conversion, 20
local interpretation, 24
model, *see* model
open term algebra, 22
polymorphic, 78
reduction, 21
simply-typed, 64
untyped, 20
lambda conversion, 20
categorical model of, 30
syntactical model of, 48
lambda model, 29

lambda reduction, 21
lambda term
  boolean, 21
  Church numeral, 21
  closed, 20
  normal form of, 21
  raw, 20, 64, 78
  substitution of, 20
  untyped, 20
lambda theories
  category of, 28
lambda theory, 20
  $\underline{\lambda\beta}$, 20
  $\underline{\lambda\beta\eta}$, 20
  pure, 20, 65
lambda-order, 43
lambda-preorder, 43
lattice, 10
  complete, 10
least upper bound, 10
left-full functor, 63
lift of a model of reduction, 54
limit, 8
limit-colimit coincidence, 12
limiting cone, 8
limiting morphism, 8
linear order, 11
local interpretation
  failure of rule ($\xi$), 25
  of combinatory logic, 23
  of the lambda calculus, 24
locally continuous functor, 13
locally well-pointed object, 32
lower bound, 10
LTS, *see* labeled transition system

Mal'cev axioms, 40
Mal'cev operator, 40
Mal'cev variety, 40
map, *see* function
maximum and minimum, 10
meet and join, 10
meet cpo, 11
message, in join calculus, 99
Meyer-Scott axiom, 29
minimum and maximum, 10
model
  continuously complete, 38
  $D_\infty$, 13
  finitely separable, 36
  Kripke, 72, 82
  non-strict, 67, 81
  of lambda conversion
    categorical, 30
    syntactical, 48
  of lambda reduction
    categorical, 48
    syntactical, 48
  of polymorphic lambda calculus, 79, 81
  of simply-typed lambda calculus, 65, 67
  partial, 51
  reflexive ccc model, 30
  set-theoretic, 68, 82
  standard models of polymorphism, 82
  strict, 65, 79
  topological, 38
  tree model, 50
  with empty types, 68, 69, 82
  with non-empty types, 69, 82
monic, 6
  collective, 6
  cone, 6
  split, 6
monomorphism, 6
monotone function, 10
morphism, 5
  colimiting, 8
  cover, 60
  currying, 9
  epic, 6
  identity, 5
  inverse, 6
  iso, 6
  limiting, 8
  monic, 6
  pairing, 9
  projection, 8
  uncurrying, 9

$n$-permutability, 40
name
  defined, 99
  free and bound, 99
  in CCS, 97
  in join calculus, 99
natural isomorphism, 7
natural transformation, 7
  Henkin, 74
(*non-empty*) rule, 68
non-empty types, 69, 82
non-strict interpretation
  of polymorphic lambda calculus, 81
  of simply-typed lambda calculus, 67
normal form, 21

object in category, 5
$\omega$-chain, 11
$\omega$-complete poset, *see* cpo
$\omega$-continuity, 11
open term algebra, 22, 36
operation in algebra, 14
order, 10
  Berry, 11
  complete, *see* cpo
  directed, 11
  directed complete, 11
  linear, 11
  $\omega$-complete, *see* cpo
  partial, 10
  pointwise, 10
  preorder, 10
  stable, 11
order-extensionality, 49
ordered algebra, 16, 40
ordered applicative structure, 47
  order-extensional, 49
  strongly extensional, 50
ordered variety, 16, 40
out-buffered agent, 87
  first-order axioms for, 89
  second-order axioms for, 101
  with feedback, 94
    first-order axioms for, 95
out-queued agent, 87
  first-order axioms for, 90
  second-order axioms for, 103
output actions, 85
output-commutativity, 89
output-confluence, 89
output-determinacy, 89
  backwards, 95
  weak, 98

pairing, 9
parallel composition
  with interaction, 94
  without interaction, 93
partial $\forall$-diagram, 74
partial applicative structure, 51
partial exponential diagram, 59
partial initial object, 60
partial model, 51
partial order, *see* order
  complete, *see* cpo
  directed complete, 11
partial syntactical lambda model, 51
*PL*-category, 72
  base, 72
  congruence, 74
  fiber, 72
  representation of, 73
    Henkin-*PL*-representation, 74
pointed poset, 10
pointwise order, 10
polymorphic Kripke model, 82
polymorphic lambda calculus, 78
polymorphic signature, 78
polynomial, 15
polynomial algebra, 15
poset, *see* order
  directed, 11
  directed complete, 11
  flat, 10
  linearly ordered, 11
  $\omega$-complete, *see* cpo
  pointed, 10
pre-structure, 74
preorder, 10
  discrete, 36
  indiscrete, 36
  symmetric, 36
  trivial, 36
presheaf, 9, 63
principal extended theory, 70
process
  in CCS, 97
  in join calculus, 99
process constant in CCS, 97
product, 8
  binary, 8
  of categories, 5
projection morphism, 8
projection-embedding pair, 12
pullback, 8
pure lambda theory, 20
  polymorphic, 79
  simply-typed, 65

queue $Q$, 87
queued agent, 87
quotient algebra, 13

raw lambda term, 20
  polymorphic, 78
  simply-typed, 64
reachable state, 100
reaction in join calculus, 99
receptivity, 90
redex, 21

$\sim$-reduced agent, 84
$\approx$-reduced agent, 84, 95
reduction, 21
categorical model of, 48
syntactical model of, 48
reflexive ccc model, 30, 37
and lambda algebras, 31
reflexive object, 30
reflexivity, 10
relabeling function in CCS, 97
relation
compatible, 13
congruence, 13
representable functor, 7
representation
Henkin-*PL*, 74
of ccc's, 9
of *PL*-categories, 73
rule (*non-empty*), 68
rule, in join calculus, 99

Scott-continuity, 11
self-composition of agent, 94
separable subset of lambda algebra, 36
sequence, expanding, 12
sequential composition, 85, 94
set-theoretic model
of polymorphism, 82
of simply-typed lambda calculus, 68
with empty types, 68, 69, 82
with non-empty types, 69, 82
$\Sigma$-algebra, 13
$\Sigma$-term, 14
signature
algebraic, 13
polymorphic, 78
simply-typed, 64
silent action, 84
simple type, 64
simply-typed lambda calculus, 64
simply-typed signature, 64
small category, 5
source of a morphism, 5
special ccc, 62, 70
split cover, 60
split epic, 6
split monic, 6
stable function, 11
stable order, 11
standard model, 82
standard structure, 75
standard term algebra, 22, 36
state
in join calculus, 99
initial, 84
of labeled transition system, 84
reachable, 100
strict interpretation
of polymorphic lambda calculus, 79
of simply-typed lambda calculus, 65
strong bisimulation, 84
strong extensionality, 50
structural transition in join calculus, 99
subalgebra, 13
subcategory, 7
faithful, 7
full, 7
substitution, 20
supremum, 10
symmetric preorder, 36
synchrony, 83
syntactical model
of conversion, 48
of reduction, 48

$\mathbf{T}$-algebra, 14
target of a morphism, 5
term
combinatory, 22
lambda, 20
$\Sigma$-term, 14
term algebra, 14
open and closed, 22, 36
terminal object, 8
terminator, 8
theory
extended, 70
of combinatory logic, 22
polymorphic, 79
simply-typed, 65
untyped, 20
topological completeness problem, 38
topological model, 38
transition relation, 84
in join calculus, 99
transition system, *see* labeled transition system
transitivity, 10
translation of lambda theories, 28
tree, 50
bounded, 50
tree model, 50
trivial preorder, 36
type
constant, 64, 78

of a labeled transition system, 84
polymorphic, 78
simple, 64
variable, 78
type assignment
polymorphic, 78
simply-typed, 65
typed lambda calculus, 64, 78
typing judgment
polymorphic, 78
simply-typed, 65

un-$\lambda$-orderable, 43
un-$\lambda$-preorderable, 43
un-preorderable, 43
uncurrying, 9
unit of adjunction, 7
unobservable action, 84
unorderable
absolutely, 39
combinatory algebra, 36, 43
$\mathbf{T}$-algebra, 39
untyped lambda calculus, 20
updeal, 10
upper bound, 10

valid typing judgment
polymorphic, 78
simply-typed, 65
valuation
in algebra, 14
in applicative structure, 22
in ordered applicative structure, 47
variable, 14, 20, 64
free and bound, 20, 64, 78
individual, 78
type, 78
variety
algebraic, 14
dcpo, 17, 41
ordered, 16, 40

weak bisimulation, 84
weak extensionality, 29
weak output-determinacy, 98
well-pointed object, 32, 59
locally, 32
well-supported object, 60

Yoneda embedding, 9

0-fiber, 73