Domain Theory and the Logic of Observable Properties

Samson Abramsky

Submitted for the degree of Doctor of Philosophy

Queen Mary College

University of London

October 31st 1987

Abstract

The mathematical framework of Stone duality is used to synthesize a number of hitherto separate developments in Theoretical Computer Science:

- Domain Theory, the mathematical theory of computation introduced by Scott as a foundation for denotational semantics.
- The theory of concurrency and systems behaviour developed by Milner, Hennessy *et al.* based on operational semantics.
- Logics of programs.

Stone duality provides a junction between semantics (spaces of points = denotations of computational processes) and logics (lattices of *properties* of processes). Moreover, the underlying logic is *geometric*, which can be computationally interpreted as the logic of *observable* properties—i.e. properties which can be determined to hold of a process on the basis of a finite amount of information about its execution.

These ideas lead to the following programme:

1. A metalanguage is introduced, comprising
   - types = universes of discourse for various computational situations.
   - terms = programs = syntactic intensions for models or points.
2. A standard denotational interpretation of the metalanguage is given, assigning domains to types and domain elements to terms.
3. The metalanguage is also given a *logical* interpretation, in which types are interpreted as propositional theories and terms are interpreted *via* a program logic, which axiomatizes the properties they satisfy.

4. The two interpretations are related by showing that they are Stone duals of each other. Hence, semantics and logic are guaranteed to be in harmony with each other, and in fact each determines the other up to isomorphism.

5. This opens the way to a whole range of applications. Given a denotational description of a computational situation in our meta-language, we can turn the handle to obtain a logic for that situation.

Organization

Chapter 1 is an introduction and overview. Chapter 2 gives some background on domains and locales. Chapters 3 and 4 are concerned with 1–4 above. Chapters 5 and 6 each develop a major case study along the lines suggested by 5, in the areas of concurrency and λ-calculus respectively. Finally, Chapter 7 discusses directions for further research.

Preface

Acknowledgements

My warmest thanks to the many people who have helped me along the way:

- To my colleagues at Queen Mary College (1978–83) for five very happy and productive years.
- To my supervisor, Richard Bornat, who gave me so much of his time during my two years as a full-time Research Student, and also gave me confidence in the worth of my ideas.
- To Tom Maibaum for our regular meetings to work on semantics in 1982–3; these were a life-line when my theoretical work had previously been done in a vacuum.
- To my colleagues in the Theory and Formal Methods Group in the Department of Computing, Imperial College: Mark Dawson, Dov Gabbay, Chris Hankin, Yves Lafont, Tom Maibaum, Luke Ong, Iain Phillips, Martin Sadler, Mike Smyth, Richard Sykes, Paul Taylor and Steve Vickers, for creating such a stimulating and inspiring environment in which to work.
- To Axel Poigné, who has just returned to Germany to take up a post at GMD, for being the most inspiring of colleagues, whose interest in and encouragement of my work has meant a great deal to me.
- To Mark Dawson, for unfailingly finding elegant solutions to all my computing problems.

- To my hosts for two very enjoyable visits when much of the work reported in Chapters 5 and 6 was done: the Programming Methodology Group, Chalmers Technical University, Göteborg, Sweden, March 1984; and Professor Raymond Boute and the Functional Languages and Architectures Group, University of Nijmegen, the Netherlands, March–April and August, 1986.

- To a number of colleagues for conversations, lectures and writings which have provided inspiration and stimulus to this work: Henk Barendregt, Peter Dybjer, Matthew Hennessy, Per Martin-Löf, Robin Milner, Gordon Plotkin, Jan Smith, Mike Smyth, Colin Stirling and Glynn Winskel. Glynn's persistent enthusiasm for and encouragement of this work have meant a great deal.

The ideas of Mike Smyth, Gordon Plotkin and Per Martin-Löf have been of particular importance to me in my work on this thesis. Equally important has been the paradigm of how to do Computer Science which I like many others have found in the work of Robin Milner and Gordon Plotkin. I thank them all for their inspiration and example.

I thank the Science and Engineering Research Council for supporting my work, firstly with a Research Studentship and then with a number of Research Grants. Thanks also to the Alvey Programme for funding such "long-term" research, and in particular for providing the equipment on which this document was produced (by me).

Finally, I thank my family for their love and support and, over the past few months, their forbearance.

Chronology

It may be worthwhile to make a few remarks about the chronology of the work reported in this thesis, as a number of manuscripts describing different versions of some of the material have been in circulation over the past few years. My first version of "Domain Logic" was worked out in October and November of 1983, and presented to the Logic Programming Seminar at Imperial (the invitation was never repeated), and again at a seminar at Manchester arranged by Peter Aczel the following February. The slides of the talk, under the title "Intuitionistic Logic of Computable Functions", were copied to a few researchers. The main results of Chapter 6 were obtained, in the setting of Martin-Löf's Domain Interpretation of his Type Theory, during and shortly after a visit to Chalmers in March 1984. A draft paper was begun in 1984 but never completed; it formed the basis of a talk given at the CMU Seminar on Concurrency in July 1984. The outline of Chapter 5 was developed, with the benefit of many discussions with Axel Poigné, in October and November 1984. Thus the main ideas of the thesis had been formulated, admittedly in rather inchoate form, by the end of 1984. The following year was mainly taken up with other things; but a manuscript on "Domain Theory in Logical Form", essentially the skeleton of the present Chapter 4, minus the endogenous logic, was written in December 1985, and circulated among a few researchers. A manuscript on "A Domain Equation for Bisimulation" was written during a visit to the University of Nijmegen in March–April 1986, and another on "Finitary Transition Systems" soon afterwards. A talk on "The Lazy λ-Calculus" was given at Nijmegen in August 1986. Chapters 3, 5 and 6 were written in September–December 1986, together with a skeletal version of Chapter 4, which was presented at the Second Symposium on Logic in Computer Science at Cornell, June 1987 [Abr87a].

Contents

Chapter 1

Introduction

The main aim of this thesis is to synthesize a number of hitherto separate developments in Theoretical Computer Science and Logic:

- Domain Theory, the mathematical theory of computation introduced by Scott as a foundation for denotational semantics.
- The theory of concurrency and systems behaviour developed by Milner, Hennessy *et al.* based on operational semantics.
- Logics of programs.
- Locale Theory.

The key to our synthesis is the mathematical theory of Stone duality, which provides a junction between semantics (topological spaces) and the *logic of observable properties* (locales). As a worked example, we show how Domain Theory can be construed as a logic of observable properties; and explore some applications to the study of programming languages.

1.1 Background

Domain Theory has been extensively studied since it was introduced by Scott [Sco70], both as regards the basic mathematical theory [Plo81], and the applications, particularly in denotational semantics [MS76], [Sto77], [Gor79], [Sch86], and more recently in static program analysis [Myc81], [Nie84], [AH87]. In the course of this development, a number of new perspectives have emerged.

Syntax vs. Semantics

Domain theory was originally presented as a model theory for computation, and this aspect was emphasised in [Sco70, Sco80a]. However, the effective character of domain constructions was immediately evident, and made fully explicit in [EC76, Sco76, Smy77, Kan79]. Moreover, in recent presentations of domains via neighbourhood systems and information systems [Sco81, Sco82], Scott has shown how the theory can be based on elementary, and finitary, set-theoretic representations, which in the case of information systems are deliberately suggestive of proof theory.

A further step towards explicitly syntactic presentations of domain theory was taken by Martin-Löf, in his Domain Interpretation of Intuitionistic Type Theory [Mar83]. His formulation also traces a line of descent from Kreisel's definition of the continuous functionals [Kre59], via [Mar70, Ers72].

The general tendency of these developments is to suggest that domains may as well be viewed in terms of *theories* as of *models*. Our work should not only confirm this suggestion, but also show how it may be put to use.

Points vs. Properties

An important recent development in mathematics has been the rise of *locale theory*, or "topology without points" [Joh82], in which the open-set lattices rather than the spaces of points become the primary objects of study. That these mathematical developments have direct bearing on Computer Science was emphasised by Smyth in [Smy83b]. If we think of the open sets as *properties* or *propositions*, we can think of spaces as logical *theories*; continuous maps act on these theories under inverse image as *predicate transformers* in the sense of Dijkstra [Dij76], or modal operators as studied in *dynamic logic* [Pra81, Har79].

There is also an important theme in Computer Science which emerges as confluent with these mathematical developments; namely, the use of notions of *observation* and *experiment* as a basis for the behavioural semantics of systems. This plays a major role in the work of Milner, Hennessy *et al.* on concurrent systems [Mil80, HM85, Win80], and also in the theory of higher-order functional languages, e.g. [Plo77, Mil77, BC85, BCL85]. The leading idea here is to take some notion of *observable event* or *experiment* as an "information quantum", and to construct the meaning of a system out of its information quanta. This corresponds to the leading idea of locale theory, that "points" are nothing but constructions out of properties. By exploiting this correspondence, we may hope to obtain a *rapprochement* between domain theory and denotational semantics, on the one hand, and operationally formulated notions such as *observation equivalence* [HM85] on the other.

Denotational vs. Axiomatic

Another area in programming language theory which has received intensive development over the past 15 years has been *logics of programs*, e.g. Hoare logic [Hoa69, dB80], dynamic logic [Pra81, Har79], temporal logic [Pnu77], etc. However, to date there has not been a satisfactory integration of this work with domain theory. For example, dynamic logic deals with sets and relations, which from the perspective of domain theory corresponds only to an extremely naive and restricted fragment of programming language semantics. One would like to see a dynamic logic of *domains* and *continuous functions*, which would encompass higher-order functions, quasi-infinite (or "lazy") data structures, self-application, non-determinism, and all the other computational phenomena for which domain theory provides a mathematical foundation.

The key mathematical idea which forms the basis of our attempt to draw all these diverse strands together is *Stone Duality*, which we now briefly review; a fuller discussion will be found in Chapter 2.

1.2 Overview: Stone Duality

The classic Stone Representation Theorem for Boolean algebras [Sto36] is aimed at solving the following problem:

> show that every (abstract) Boolean algebra can be represented as a field of sets, in which the operations of meet, join and complement are represented by intersection, union and set complement.

Stone's solution to the problem begins with observation that for any topological space X, the lattice $\mathsf{Clop}\ X$ of clopen subsets of X forms a field of sets. His radical step was to construct, from any Boolean lagebra B, a topological space $\mathsf{Spec}\ B$. To understand the construction, think of B as (the

Lindenbaum algebra of) a classical propositional theory. The elements of B are thus to be thought of as (equivalence classes of) formulae, and the operations as logical conjunction, disjunction and negation. Now a *model* of B is an assignment of "truth-values" 0 or 1 to elements of B, in a manner consistent with the logical structure; e.g. so that $\neg b$ is assigned 1 if and only if b is assigned 0. In short, a model is a Boolean algebra homomorphism $f : B \to \mathbf{2}$, where $\mathbf{2} = \{0, 1\}$ is the two-element lattice. Identifying such an f with $f^{-1}(1) \subseteq B$, which as is well-known is an *ultrafilter* over B (see e.g. [Joh82]), we can take $\mathsf{Spec}\ B$ as the set of ultrafilters over B, with the topology generated by

$$U_a \equiv \{x \in \mathsf{Spec}\ B : a \in x\} \quad (a \in B).$$

The spaces arising as $\mathsf{Spec}\ B$ for Boolean algebras B in this way were characterised by Stone as the totally disconnected compact Hausdorff spaces (subsequently named *Stone spaces* in his honour). Moreover, we have the isomorphisms

$$B \cong \mathsf{Clop}\ \mathsf{Spec}\ B \tag{1.1}$$

$$b \mapsto \{x \in \mathsf{Spec}\ B : b \in x\}$$

$$S \cong \mathsf{Spec}\ \mathsf{Clop}\ S \tag{1.2}$$

$$s \mapsto \{U \in \mathsf{Clop}\ S : s \in U\}.$$

The first of these isomorphisms solves the representation problem, and comprises Stone's Theorem in its classical form. But we can go further; these correspondences also extend (contravariantly) to morphisms:

$$\frac{S \xrightarrow{f} T}{\mathsf{Clop}\ S \xleftarrow{f^{-1}} \mathsf{Clop}\ T} \qquad \frac{A \xleftarrow{h^\star} B}{\mathsf{Spec}\ A \xrightarrow{h} \mathsf{Spec}\ B}$$

where

$$h : x \mapsto \{b \in B : h^\star b \in x\}.$$

In modern terminology, this yields a *duality* (= contravariant equivalence of categories):

$$\mathbf{Stone} \simeq \mathbf{Bool}^{\mathsf{op}}.$$

This is the prototype for a whole family of "Stone-type duality theorems", and leads to locale theory, as "pointless topology" or junior-grade (propositional) topos theory. (An excellent reference for these topics is [Joh82]).

But what has all this to do with Computer Science? Two interpretations of Stone duality can be found in the existing literature from mathematics and logic:

- The topological view: Points vs. Open sets.

- The logical view: Models vs. Formulas.

We wish to add a third interpretation:

- The Computer Science view: (Denotations of) computational processes vs. (extensions of) specifications.

The importance of Stone duality for Computer Science is that *it provides the right framework for understanding the relationship between denotational semantics and program logic.* The fundamental logical relationship of program development is

$$P \models \phi$$

to be read "P satisfies ϕ", where P is a program (a syntactic description of a computational process), and ϕ is a formula (a syntactic description of a property of computations). Thus P is the "how" and ϕ the "what" in the dichotomy standardly used to explain the distinction between programs and specifications. We can easily describe the main formal activities of the program development process in terms of this relation:

- *Program specification* is the task of defining (a list of) properties ϕ to be satisfied by the program.

- *Program synthesis* is the task of finding P given (a list of) ϕ.

- *Program verification* is the task of proving that $P \models \phi$.

The two sides of Stone duality—the spatial and the logical or localic—yield alternative but equivalent perspectives on this fundamental relationship:

- The spatial side of the duality, where points are taken as primary, properties are constructed as (open) sets of points, and the fundamental relationship is interpreted as $s \in U$ (s a point, U a property), corresponds to *denotational semantics*, where the data domains (i.e. the *types*) of a programming language are interpreted as spaces of points, and programs are given denotations as points in these spaces; this denotational perspective yields a topological interpretation of program logic.

- The logical or localic side of the duality, where properties, as elements of an abstract (logical) lattice, are taken as primary, and points are constructed as sets (prime filters) of properties, with the fundamental relationship interpreted as $a \in x$ (a a property, x a point), corresponds to program logic, and yields a *logical interpretation of denotational semantics*. The idea is that the structure of the open-set lattices and prime filters are presented *syntactically*, via axioms and inference rules, as a formal system.

We extract the following concrete research programme from these general perspectives on Stone duality:

1. A metalanguage is introduced, comprising
   - types = data domains = universes of discourse for various computational situations.
   - terms = programs = syntactic intensions for models or points.

2. A standard denotational interpretation of the metalanguage, assigning domains to types and domain elements to terms, can be given using the spatial side of Stone duality.

3. The metalanguage is also given a *logical* interpretation, in which the localic side of the duality is presented as a formal system with axioms and inference rules. Each type is interpreted as a propositional theory; and terms are interpreted by axiomatising the satisfaction relation $P \models \phi$. This gives a program logic.

4. The denotational semantics from 2 and the program logic from 3 are related by showing that they are Stone duals of each other—a strengthened form of the logician's "Soundness and Completeness". As a consequence of this, semantics and logic are guaranteed to be in harmony

with each other, and in fact each determines the other up to isomorphism.

5. The framework developed in 1–4 is very *general*. The metalanguage can be used to describe a wide variety of computational situations, following the ideas of "classical" denotational semantics. Given such a description, we can turn the handle to obtain a logic for that situation. This offers two exciting prospects: of replacing *ad hoc* ingenuity in the design of program logics to match a given semantics by the routine application of systematic general theory; and of bringing hitherto divergent fields of programming language theory (e.g. λ-calculus and concurrency) within the scope of a single unified framework.

The main objective of this thesis is to elaborate the programme outlined in 1–5. Chapter 2 is devoted to filling in some background on domains and locales. Then Chapters 3 and 4 are concerned with 1–4 above. Chapters 5 and 6 each develop a major case study along the lines suggested by 5, in the areas of concurrency and λ-calculus respectively. Finally, Chapter 7 discusses directions for further research.

Chapter 2

Background: Domains and Locales

The purpose of this Chapter is to summarise what we assume, to fix notation, and to review some basic definitions and results.

2.1 Notation

Most of the notation from elementary set theory and logic which we will use is standard and should cause no problems to the reader. We shall use $\equiv$ for *definitional equality*; thus $M \equiv N$ means "the expression M is by definition equal to" (or just: "is defined to be") "N". We shall use ω to denote the natural numbers $\{0, 1, \ldots\}$ (thought of sometimes as an ordinal, and sometimes as just a set); and $\mathbb{N}$ to denote the set of *positive* integers $\{1, 2, \ldots\}$. Given a set X, we write $\wp X$ for the powerset of X, $\wp_{\mathsf{f}} X$ for the set of *finite* subsets of X, and $\wp_{\mathsf{fne}} X$ for the *finite non-empty* subsets. We write $X \subseteq_{\mathsf{f}} Y$ for "X is a finite subset of Y".

We write substitution of N for x in M, where M, N are expressions and x is a variable, as $M[N/x]$. We shall assume the usual notions of free and bound variables, as expounded e.g. in [Bar84]. We shall always take expressions modulo α-conversion, and treat substitution as a *total* operation in which variable capture is avoided by suitable renaming of bound variables.

Our notations for semantics will follow those standardly used in denotational semantics. One operation we will frequently need is *updating* of environments. Let $\mathsf{Env} = \mathsf{Var} \to \mathcal{V}$, where Var is a set of variables, and $\mathcal{V}$ some value space. Then for $\rho \in \mathsf{Env}$, $x \in \mathsf{Var}$, $v \in \mathcal{V}$, the expression $\rho[x \mapsto v]$ denotes the environment defined by

$$(\rho[x \mapsto v])y = \begin{cases} v, & x = y \\ \rho y, & \text{otherwise.} \end{cases}$$

Next, we recall some notions concerning posets (partially ordered sets). Given a poset P and $X \subseteq P$, we write

$$\begin{aligned} \downarrow(X) &= \{y \in P : \exists x \in X.\, y \leq x\} \\ \uparrow(X) &= \{y \in P : \exists x \in X.\, x \leq y\} \\ \mathsf{Con}(X) &= \{y \in P : \exists x, z \in X.\, x \leq y \leq z\} \end{aligned}$$

We write $\downarrow(x)$, $\uparrow(x)$ for $\downarrow(\{x\})$, $\uparrow(\{x\})$. A set X is *left-closed* (or *lower-closed*) if $X = \downarrow(X)$, *right-closed* (or *upper-closed*) if $X = \uparrow(X)$, and *convex-closed* if $X = \mathsf{Con}(X)$. When it is important to emphasise P we write $\downarrow_P(X)$, $\uparrow_P(X)$ etc. We also have the lower, upper and Egli-Milner *preorders* (reflexive and transitive relations) on subsets of P:

$$\begin{aligned} X \sqsubseteq_l Y &\equiv \forall x \in X.\, \exists y \in Y.\, x \leq y \\ X \sqsubseteq_u Y &\equiv \forall y \in Y.\, \exists x \in X.\, x \leq y \\ X \sqsubseteq_{EM} Y &\equiv X \sqsubseteq_l Y \,\&\, X \sqsubseteq_u Y \end{aligned}$$

We write $\mathbf{2}$ for the two-element lattice $\{0, 1\}$ with $0 < 1$, and $\mathbb{O}$ for *Sierpinski space*, which has the same carrier as $\mathbf{2}$, and topology $\{\varnothing, \{1\}, \{0, 1\}\}$. As we shall see in the section on domains and locales, $\mathbf{2}$ and $\mathbb{O}$ are really two faces of the same structure (a “schizophrenic object” in the terminology of [Joh82, Chapter 6]), since $\mathbb{O}$ arises from the Scott topology on $\mathbf{2}$, and $\mathbf{2}$ from the specialisation order on $\mathbb{O}$. For other basic notions of the theory of partial orders and lattices, we refer to [GHK*80, Joh82].

Finally, we shall assume a modicum of familiarity with elementary category theory and general topology; suitable references are [ML71] and [Dug66] respectively.

2.2 Domains

We shall assume some familiarity with [Plo81], and use it as our reference for Domain theory. We shall not review such basic definitions as *cpo* (complete partial order—[Plo81, Chapter 1 p. 7]), *continuous function* (*loc. cit.*) etc. here.

By a *category of domains* we shall mean a sub-category of **CPO**, the category of complete partial orders and continuous functions (*loc. cit.*). $\mathbf{CPO}_\bot$ is the category of *strict* functions ([Plo81, Chapter 1 p. 11]).

The properties of **CPO** which make it a suitable mathematical universe for denotational semantics—a "tool for making meanings" in Plotkin's phrase—are:

1. It admits recursive definitions, both of elements of domains, and of domains themselves.

2. It supports a rich type structure.

The mathematical content of (1) is given by the least fixed point theorem for continuous functions on cpo's ([Plo81, Chapter 1 Theorem 1]), and the initial fixed point theorem for continuous functors on **CPO** ([Plo81, Chapter 5 Theorem 1]). As for (2), the type constructions available over **CPO** are extensively surveyed in [Plo81, Chapters 2 and 3]. In order to fix notation, we shall catalogue the constructions of which mention will be made in this thesis, with references to the definitions in [Plo81]:

$A \times B$	product	Ch. 2 p. 2
$(A \rightarrow B)$	function space	Ch. 2 p. 9
$A \oplus B$	coalesced sum	Ch. 3 p. 6
$(A)_\bot$	lifting	Ch. 3 p. 9
$(A \rightarrow_\bot B)$	strict function space	Ch. 1 p. 13
$P_l A$	lower (Hoare) powerdomain	Ch. 8 p. 14
$P_u A$	upper (Smyth) powerdomain	Ch. 8 p. 45
$P_p A$	convex (Plotkin) powerdomain	Ch. 8 p. 28

(Note that *separated sum* $A + B$ can be defined by: $A + B \equiv (A)_\perp \oplus (B)_\perp$.)

In this thesis, we shall mainly be concerned with *algebraic* domains, i.e. sub-categories of ω**ALG**, the category of ω-algebraic cpo's [Plo81, Chapter 6 p. 2]. In particular, we shall be concerned with the following three full sub-categories of ω**ALG**:

1. **AlgLat**: the category of ω-algebraic lattices [Plo81, Chapter 6 p. 13].

2. **SDom**: the category of *Scott domains*, i.e. the consistently complete ω-algebraic cpo's (*loc. cit.*). (The name comes from the fact that this is exactly the category presented in [Sco81, Sco82].)

3. **SFP**: the category of *strongly algebraic* cpo's [Plo81, Chapter 6 p. 17]. The name is an acronym for "Sequences of Finite Posets"—in more standard terminology, these are the ω-profinite cpo's. This category was introduced in [Plo76].

Each of these categories is a full sub-category of the next.

The justification for studying these categories comes from the fact that **SFP** is closed under all the type constructions listed above, while **SDom** is closed under all but the Plotkin powerdomain. In particular, both are cartesian closed; indeed, **SFP** is the *largest* cartesian closed full sub-category of ω**ALG** [Smy83a], while **SDom** is the largest "basis elementary" such sub-category [Gun86]. Moreover, both categories admit initial solutions of domain equations built from these constructions (obviously excluding the Plotkin powerdomain in the case of **SDom**). Almost all the domains needed in denotational semantics to date can be defined from these constructions by composition and recursion (some exceptions of three different kinds: [Abr83b], [Ole85], [Plo82]). The reason for including **AlgLat** is that it is a usefully simpler special case, which will be applicable to our work in Chapter 6.

Given an algebraic domain D, we shall write $\mathcal{K}(D)$ for its *basis*, i.e. the sub-poset of finite elements. Now algebraic domains are *freely constructed* from their bases, i.e.

$$D \cong \mathsf{Idl}(\mathcal{K}(D))$$

where Idl is the ideal completion described in [Plo81, Chapter 6 p. 5]. Thus we can in fact completely describe such categories as **SDom** and **SFP** in an elementary fashion in terms of the bases; various ways of doing this for **SDom** are presented in [Sco81, Sco82].

An important part of this programme is to describe the type constructions listed above in terms of their effect on the bases. We shall fix some concrete definitions of the constructions for use in later chapters.

- $\mathcal{K}(A \times B) = \mathcal{K}(A) \times \mathcal{K}(B)$; the ordering is component-wise.
- $\mathcal{K}(A \oplus B) = \mathcal{K}(A) \oplus \mathcal{K}(B)$, i.e.

  $$\{\bot\} \cup (\{0\} \times (\mathcal{K}(A) - \{\bot_A\})) \cup (\{1\} \times (\mathcal{K}(B) - \{\bot_B\}))$$

  with the ordering defined by

  $$\begin{array}{rcl} x \sqsubseteq y & \equiv & x = \bot \\ & & \text{or } x = (0, a) \ \&\ y = (0, b) \ \&\ a \sqsubseteq_A b \\ & & \text{or } x = (1, c) \ \&\ y = (1, d) \ \&\ c \sqsubseteq_B d. \end{array}$$

- $\mathcal{K}((A)_\bot) = \{\bot\} \cup (\{0\} \times \mathcal{K}(A))$, with the ordering defined by

  $$\begin{array}{rcl} x \sqsubseteq y & \equiv & x = \bot \\ & & \text{or } x = (0, a) \ \&\ y = (0, b) \ \&\ a \sqsubseteq_A b. \end{array}$$

- $\mathcal{K}(P_l(A)) = \{\downarrow_{\mathcal{K}(A)}(X) : X \in \wp_{\mathsf{fne}}(\mathcal{K}(A))\}$, with the subset ordering.
- $\mathcal{K}(P_u(A)) = \{\uparrow_{\mathcal{K}(A)}(X) : X \in \wp_{\mathsf{fne}}(\mathcal{K}(A))\}$, with the superset ordering.
- $\mathcal{K}(P_p(A)) = \{\mathsf{Con}_{\mathcal{K}(A)}(X) : X \in \wp_{\mathsf{fne}}(\mathcal{K}(A))\}$, with the Egli-Milner ordering (which *is* a partial order on the convex-closed sets).

All these definitions are valid for *any* algebraic cpo. Since ω**ALG** is not cartesian closed, we must obviously describe the function space construction for one of its cartesian closed sub-categories. As the description for **SFP** is rather complicated (see [Gun85]), we shall give the simpler description for **SDom**.

Definition 2.2.1 (i) ([Plo81, Chapter 6 p. 1]). Let A, B be algebraic domains. For $a \in \mathcal{K}(A)$, $b \in \mathcal{K}(B)$,

$$[a, b] : A \to B$$

is the *one-step* function defined by

$$[a, b]d = \begin{cases} b & \text{if } a \sqsubseteq d \\ \bot & \text{otherwise} \end{cases}$$

(ii) ([Plo81, Chapter 6 p. 13]). $X \subseteq A$ is *consistent*:

$$\triangle(X) \equiv \exists d \in A. \forall x \in X. x \sqsubseteq d.$$

We write $x \triangle y$ for $\triangle\{x, y\}$.

Note that Plotkin writes $(a \Rightarrow b)$ for $[a, b]$, and $\uparrow X$ for $\triangle(X)$.

Proposition 2.2.2 *([Plo81, Chapter 6 pp. 14–15]). Let A, B be Scott domains, and $\{a_i\}_{i \in I} \subseteq \mathcal{K}(A)$, $\{b_i\}_{i \in I} \subseteq \mathcal{K}(B)$ for some finite set I.*
(i) $\triangle\{[a_i, b_i] : i \in I\}$ if and only if

$$\forall J \subseteq I. \triangle \{a_j : j \in J\} \;\Rightarrow\; \triangle\{b_j : j \in J\}$$

(ii) $\triangle\{[a_i, b_i] : i \in I\}$ implies that $\bigsqcup\{[a_i, b_i] : i \in I\}$ exists and is defined by

$$(\bigsqcup\{[a_i, b_i] : i \in I\})d = \bigsqcup\{b_i : a_i \sqsubseteq d\}.$$

Now we finally get our description of the function space:

- For Scott domains A, B:

$$\begin{aligned} \mathcal{K}(A \to B) \;=\; & \{\bigsqcup\{[a_i, b_i] : i \in I\} : I \text{ finite}, \\ & \{a_i\}_{i \in I} \subseteq \mathcal{K}(A),\ \{b_i\}_{i \in I} \subseteq \mathcal{K}(B), \\ & \triangle\{[a_i, b_i] : i \in I\}\}. \end{aligned}$$

2.3 Locales

Our reference for locale theory and Stone duality will be [Joh82]. Since locale theory is not yet a staple of Computer Science, we shall briefly review some of the basic ideas.

Classically, the study of general topology is based on the category **Top** of topological spaces and continuous maps. However, in recent years mathematicians influenced by categorical and constructive ideas have advocated that attention be shifted to the open-set lattices as the primary objects of study. Given a space X, we write $\Omega(X)$ for the lattice of open subsets of X ordered by inclusion. Since $\Omega(X)$ is closed under arbitrary unions and finite intersections, it is a complete lattice satisfying the infinite distributive law

$$a \wedge \bigvee S = \bigvee \{a \wedge s : s \in S\}.$$

(By the Adjoint Functor Theorem, in any complete lattice this law is equivalent to the existence of a right adjoint to conjunction, i.e. to the fact that implication can be defined in a canonical way.) Such a lattice is a *complete Heyting algebra*, i.e. the Lindenbaum algebra of an *intuitionistic* theory. The continuous functions between topological spaces preserve unions and intersections, and hence all joins and finite meets of open sets, under inverse image; thus we get a functor

$$\Omega : \mathbf{Top} \rightarrow \mathbf{Loc}$$

where **Loc**, the category of *locales*, is the opposite of **Frm**, the category of *frames*, which has complete Heyting algebras as objects, and maps preserving all joins and finite meets as morphisms. Note that **Frm** is a concrete category of structured sets and structure-preserving maps, and consequently convenient to deal with (for example, it is monadic over **Set**). Thus we study **Loc** *via* **Frm**; but it is **Loc** which is the proposed alternative or replacement for **Top**, and hence the ultimate object of study.

Notation. Given a morphism $f : A \rightarrow B$ in **Loc**, we write $f^{\star}$ for the corresponding morphism $B \rightarrow A$ in **Frm**.

Now we can define a functor

$$\mathsf{Pt} : \mathbf{Loc} \rightarrow \mathbf{Top}$$

as follows (for motivation, see our discussion of Stone's original construction in Chapter 1): $\mathsf{Pt}(A)$ is the set of all frame morphisms $f : A \rightarrow \mathbf{2}$, where $\mathbf{2}$ is the two-point lattice. Any such f can be identified with the set $F = f^{-1}(1)$, which satisfies:

$$1 \in F$$

$$a, b \in F \;\Rightarrow\; a \wedge b \in F$$

$$a \in F, a \leq b \;\Rightarrow\; b \in F$$

$$\bigvee_{i \in I} a_i \in F \;\Rightarrow\; \exists i \in I.\, a_i \in F.$$

Such a subset is called a *completely prime filter*. Conversely, any completely prime filter F determines a frame homomorphism $\chi_F : A \to \mathbf{2}$. Thus we can identify $\mathsf{Pt}(A)$ with the completely prime filters over A. The topology on $\mathsf{Pt}(A)$ is given by the sets U_a $(a \in A)$:

$$U_a \equiv \{x \in \mathsf{Pt}(A) : a \in F\}.$$

Clearly,

$$\mathsf{Pt}(A) = U_1, \quad U_a \cap U_b = U_{a \wedge b}, \quad \bigcup_{i \in I} U_{a_i} = U_{\bigvee_{i \in I} a_i},$$

so this is a topology. Pt is extended to morphisms by:

$$\frac{A \xleftarrow{f^\star} B}{\mathsf{Pt}(A) \xrightarrow{\mathsf{Pt}(f)} \mathsf{Pt}(B)}$$

$$\mathsf{Pt}(f)x = \{b : f^\star b \in x\}.$$

We now define, for each X in **Top** and A in **Loc**:

$$\eta_X : X \to \mathsf{Pt}(\Omega(X))$$

$$\eta_X(x) = \{U : x \in U\}$$

$$\epsilon_A : \Omega(\mathsf{Pt}(A)) \to A$$

$$\epsilon_A^\star(a) = \{x : a \in x\}.$$

Now we have

Theorem 2.3.1 *([Joh82, II.2.4]).* $(\Omega, \mathsf{Pt}, \eta, \epsilon) : \mathbf{Top} \rightharpoonup \mathbf{Loc}$ *defines an adjunction between* **Top** *and* **Loc***; moreover ([Joh82, II.2.7]), this cuts down to an equivalence between the full sub-categories* **Sob** *of sober spaces and* **SLoc** *of spatial locales.*

The equivalence between **Sob** and **SLoc** (and therefore the *duality* or contravariant equivalence between **Sob** and **SFrm**) may be taken as the most general purely topological version of Stone duality. For our purposes, some dualities arising as restrictions of this one are of interest.

Definition 2.3.2 A space X is *coherent* if the compact-open subsets of X (notation: $K\Omega(X)$) form a basis closed under finite intersections, i.e. for which $K\Omega(X))$ is a distributive sub-lattice of $\Omega(X)$.

Theorem 2.3.3 *(i) ([Joh82, II.2.11]). The forgetful functor from* **Frm** *to* **DLat***, the category of distributive lattices, has as left adjoint the functor* Idl*, which takes a distributive lattice to its ideal completion.*
(ii) ([Joh82, II.3.4]). Given a distributive lattice A*, define* $\mathsf{Spec}\ A$ *as the set of prime filters over* A *(i.e. sets of the form* $f^{-1}(1)$ *for lattice homomorphisms* $f : A \to \mathbf{2}$*), with topology generated by*

$$U_a \equiv \{x \in \mathsf{Spec}\ A : a \in x\} \quad (a \in A).$$

Then $\mathsf{Spec}\ A \cong \mathsf{Pt}(\mathsf{Idl}(A))$*.*
(iii) ([Joh82, II.3.3]). The duality of Theorem 2.3.1 cuts down to a duality

$$\mathbf{CohSp} \simeq \mathbf{CohLoc} \simeq \mathbf{DLat}^{\mathsf{op}}$$

where **CohSp** *is the category of coherent* T_0 *spaces, and continuous maps which preserve compact-open subsets under inverse image; and* $\mathbf{CohLoc}^{\mathsf{op}}$ *is the image of* **DLat** *under the functor* Idl*.*

The logical significance of the coherent case is that finitary syntax—specifically finite disjunctions—suffices. The original Stone duality theorem discussed in Chapter 1 is obtained as the further restriction of this duality to coherent Hausdorff spaces (which turns out to be another description of the Stone spaces) and Boolean algebras, i.e. complemented distributive lattices. Note that under the compact Hausdorff condition, *all* continuous maps satisfy the special property in part (iii) of the Theorem.

As a further special case of Stone duality, we note:

Theorem 2.3.4 *(i) The forgetful functor from distributive lattices to the category* **MSL** *of meet-semilattices has a left adjoint* L*, where* $\mathsf{L}(A) = \{\downarrow(X) : X \in \wp_{\mathsf{f}}(A)\}$*, ordered by inclusion. (Notice that this is the same construction as for the lower powerdomain; this fact* is *significant, but not in the scope of this thesis.)*
(ii) For any meet-semilattice A*, define* $\mathsf{Filt}(A)$ *as the set of all filters over* A*, with topology defined exactly as for* $\mathsf{Spec}(A)$*. Then*

$$\mathsf{Filt}(A) \cong \mathsf{Spec}(\mathsf{L}(A)) \cong \mathsf{Pt}(\mathsf{Idl}(\mathsf{L}(A))).$$

(iii) The duality of Theorem 2.3.3 cuts down to a duality

$$\mathbf{CohAlgLat} \simeq \mathbf{MSL}^{\mathsf{op}}$$

where **CohAlgLat** *is the full sub-category of* **CohSp** *of algebraic lattices with the Scott topology (to be defined in the next section).*

An extensive treatment of locale theory and Stone-type dualities can be found in [Joh82]. Our purpose in the remainder of this section is to give some conceptual perspectives on the theory.

Firstly, a *logical* perspective. As already mentioned, locales are the Lindenbaum algebras of intuitionistic theories, more particularly of *propositional geometric theories*, i.e. the logic of finite conjunctions and infinite conjunctions. The morphisms preserve this geometric structure, but are *not* required to preserve the additional "logical" structure of implication and negation (which can be defined in any complete Heyting algebra). Thus from a logical point of view, locale theory is propositional geometric logic. Moreover, Stone duality also has a logical interpretation. The *points* of a space correspond to *models* in the logical sense; the *theory* of a model is the completely prime filter of opens it satisfies, where the satisfaction relation is just

$$x \models a \equiv x \in a$$

in terms of spaces, (i.e. with $x \in X$ and $a \in \Omega(X)$), and

$$x \models a \equiv a \in x$$

in terms of locales (i.e. with $x \in \mathsf{Pt}(A)$ and $a \in A$). Spatiality of a class of locales is then a statement of *Completeness*: every consistent theory has a model.

Secondly, a *computational* perspective. If we view the points of a space as the denotations of computational processes (programs, systems), then the elements of the corresponding locale can be seen as *properties* of computational processes. More than this, these properties can in turn be thought of as computationally meaningful; we propose that they be interpreted as *observable properties*. Intuitively, we say that a property is observable if we can tell whether or not it holds of a process on the basis of only a finite amount of information about that process[1]. Note that this is really *semi*-observability, since if the property is *not* satisfied, we do not expect that this is finitely observable. This intuition of observability motivates the asymmetry between conjunction and disjunction in geometric logic and topology. Infinite disjunctions of observable properties are still observable—to see that $\bigvee_{i \in I} a_i$ holds of a process, we need only observe that *one* of the a_i holds—while infinite conjunctions clearly do not preserve finite observability in general. More precisely, consider Sierpinski space $\mathbb{O}$. We can regard this space as representing the possible outcomes of an experiment to determine whether a property is satisfied; the topology is motivated by semi-observability, so an observable property on a space X should be a *continuous* function to $\mathbb{O}$. In fact, we have

$$\Omega(X) \cong (X \to \mathbb{O})$$

where $(X \to \mathbb{O})$ is the continuous function space, ordered pointwise (thinking of $\mathbb{O}$ as $\mathbf{2}$). Now for infinite I, I-ary disjunction, viewed as a function

$$\mathbb{O}^I \to \mathbb{O}$$

is continuous, while I-ary conjunction is not. Similarly, implication and negation, taken as functions

$$\Rightarrow : \mathbb{O}^2 \to \mathbb{O}, \quad \neg : \mathbb{O} \to \mathbb{O}$$

are not continuous. Thus from this perspective,

geometric logic = observational logic.

[1]This is really only one facet of observability. Another is *extensionality*, i.e. that we regard a process as a black box with some specified interface to its environment, and only take what is observable via this interface into account in determining the meaning of the process. Extensionality in this sense is obviously *relative* to our choice of interface; it is orthogonal to the notion being discussed in the main text.

These ideas follow those proposed by Smyth in his pioneering paper [Smy83b], but with some differences. In [Smy83b], Smyth interprets "open set" as *semi-decidable* property; this represents an ultimate commitment to interpret our mathematics in some effective universe. My preference is to do Theoretical Computer Science in as ontologically or foundationally *neutral* a manner as possible. The distinction between semi-observability and semi-decidability is analogous to the distinction between the computational motivation for the basic axioms of domain theory in terms of "physical feasibility" given in [Plo81, Chapter 1], without any appeal to notions of recursion theory; and a commitment to only considering computable elements and morphisms of effectively given domains, as advocated in [Kan79]. It should also be said that the link between observables and open sets in domain theory was clearly (though briefly!) stated in [Plo81, Chapter 8 p. 16], and used there to motivate the definition of the Plotkin powerdomain.

A final perpective is *algebraic*. The category **Frm** is algebraic over **Set** ([Joh82, II.1.2]); thus working with locales, we can view topology as a species of (infinitary) algebra. In particular, constructions of universal objects of various kinds by "generators and relations" are possible. Two highly relevant examples in the locale theory literature are [Joh85] and [Hyl81]. This provides a link with the information systems approach to domain theory as in [Sco82, LW84]. Some of our work in Chapters 3 and 4 can be seen as a systematization of these ideas in an explicitly syntactic framework.

2.4 Domains and Locales

We now turn to the connections between domains and locales. Firstly, it is standard that domains can be viewed topologically.

Definition 2.4.1 ([Plo81, Chapter 1 p. 16]). Given a poset P, the *Scott topology* on P has as open sets those $U \subseteq P$ satisfying

1. U is upper-closed, i.e. $U = \uparrow(U)$.

2. U is inaccessible by ω-chains, i.e.

$$\bigsqcup_{n \in \omega} x_n \in U \;\Rightarrow\; \exists n.\, x_n \in U.$$

We write $\sigma(D)$ for the Scott topology on a domain D.

Proposition 2.4.2 *(i) (loc. cit.) Let D, E be cpo's; a function $f : D \to E$ is continuous in the cpo sense iff it is continuous with respect to the Scott topology.*
(ii) ([Plo81, Chapter 6 p. 3]). For algebraic domains D, the Scott topology has a particularly simple form: namely all sets of the form

$$\bigcup_{i \in I} \uparrow(b_i) \quad (b_i \in \mathcal{K}(D), i \in I)$$

Moreover, the compact-open sets are just those of this form with I finite.

Given a space X, we define the *specialisation order* on X by

$$x \leq_{\mathsf{spec}} y \equiv \forall U \in \Omega(X).\, x \in U \ \Rightarrow \ y \in U.$$

Proposition 2.4.3 *([Plo81, Chapter 1 p. 16]). Let D be a cpo. The specialisation order on the space $(D, \sigma(D))$ coincides with the original ordering on D.*

Thus we may regard domains indifferently as posets or as spaces with the Scott topology, justifying some earlier abuses of notation.

We now relate domains to coherent spaces.

Theorem 2.4.4 (The 2/3 SFP Theorem) *([Plo81, Chapter 8 p. 41]). An algebraic cpo is coherent as a space iff it is "2/3 SFP" in the terminology of (loc. cit.). Since coherent spaces are sober ([Joh82] II.3.4), any such domain D satisfies*

$$D \cong \mathsf{Spec}(K\Omega(D)).$$

We shall refer to such domains as *coherent algebraic*. Thus **SDom** and **SFP** are categories of coherent spaces, and we need only consider the lattices of compact-open sets on the logical side of the duality.

We conclude with some observations which show how the finite elements in a coherent algebraic domain play an ambiguous role as both points and properties. Firstly, we have

$$D \cong \mathsf{Idl}(\mathcal{K}(D))$$

so the finite elements determine the structure of D on the spatial side. We can also recover the finite elements in purely lattice-theoretic terms from $A = K\Omega(D)$. Say that $a \in A$ is *consistent* if $a \neq 0$, and *prime* if $a \leq b \vee c$ implies $a \leq b$ or $a \leq c$. (We should probably say coprime rather than prime, but as we will have no need for the dual concept, we will use the shorter term.) Writing $cpr(A)$ for the set of consistent primes of A, we have

$$\mathcal{K}(D) = (cpr(A))^{\mathsf{op}}, \quad A \cong \mathsf{L}((\mathcal{K}(D))^{\mathsf{op}}). \tag{2.1}$$

(The fact that the latter construction produces a distributive lattice even though $\mathcal{K}(D)$ is not a meet-semilattice follows from the MUB axioms characterizing the coherent algebraic domains [Plo81, Chapter 8 p. 41].)

Theorem 2.4.5 *Let A be a distributive lattice.* $\mathsf{Spec}(A)$ *is coherent algebraic iff the following conditions are satisfied:*

(1) $1_A \in cpr(A)$

(2) $\forall a \in A. \exists b_1, \ldots, b_n \in cpr(A). \, a = \bigvee_{i=1}^{n} b_i$.

Of these, (1) ensures the existence of a bottom point, and (2) says "there are enough primes". This result will be proved as part of our work in the next Chapter.

Chapter 3

Domains and Theories

3.1 Introduction

In this Chapter, we lay some of the foundations for the domain logic to be presented in Chapter 4. In section 2, a category of domain prelocales (coherent propositional theories) and approximable mappings is defined, and proved equivalent to **SDom**. This is the category in which, implicitly, all the work of Chapter 4 is set. In section 3, following the ideas of a number of authors, particularly Larsen and Winskel in [LW84], a large cpo of domain prelocales is defined, and used to reduce the solution of domain equations to taking least fixpoints of continuous functions over this cpo. In section 4, a number of type constructions are defined as operations over domain prelocales. We prove in detail that these operations are naturally isomorphic to the corresponding constructions on domains. In section 5 a semantics for a language of recursive type expressions is given, in which each type is interpreted as a logical theory. This is related to a standard semantics in which types denote domains by showing that for each type its interpretation in the logical semantics is the Stone dual of its denotation in the standard semantics.

Important Notational Convention. Throughout this Chapter and the next, we shall use I, J, K, L to range over *finite* index sets.

3.2 A Category of Pre-Locales

Definition 3.2.1 A *coherent prelocale* is a structure

$$A = (|A|, \leq_A, =_A, 0_A, \vee_A, 1_A, \wedge_A)$$

where

- $|A|$ is a set, the *carrier*
- $\leq_A$, $=_A$ are binary relations over $|A|$
- 0_A, 1_A are constants, i.e. elements of $|A|$
- $\vee_A$, $\wedge_A$ are binary operations over $|A|$

subject to the following axioms (subscripts omitted):

$$(p1) \quad a \leq a \quad \frac{a \leq b \;\; b \leq c}{a \leq c} \quad \frac{a \leq b \;\; b \leq a}{a = b} \quad \frac{a = b}{a \leq b \;\; b \leq a}$$

$$(p2) \quad 0 \leq a \quad \frac{a \leq c \;\; b \leq c}{a \vee b \leq c} \quad a \leq a \vee b \quad b \leq a \vee b$$

$$(p3) \quad a \leq 1 \quad \frac{a \leq b \;\; a \leq c}{a \leq b \wedge c} \quad a \wedge b \leq a \quad a \wedge b \leq b$$

$$(p4) \quad a \wedge (b \vee c) \leq (a \wedge b) \vee (a \wedge c)$$

Evidently, the quotient structure

$$\tilde{A} = (|A|/=_A, \leq/=_A)$$

is a distributive lattice.

Definition 3.2.2 Given a prelocale A, we define

$$\begin{array}{rrcl}
(i) & pr(A) & \equiv & \{a \in |A| : \forall b, c \in |A|.\, a \leq b \vee c \Rightarrow a \leq b \text{ or } a \leq c\} \\
(ii) & con(A) & \equiv & \{a \in |A| : \neg(a =_A 0)\} \\
(iii) & cpr(A) & \equiv & con(A) \cap pr(A) \\
(iv) & t(A) & \equiv & \{a \in |A| : \neg(a =_A 1)\}
\end{array}$$

Definition 3.2.3 A *domain prelocale* is a coherent prelocale A which satisfies the following additional axioms:

$$(d1) \quad \forall a \in |A|.\, \exists b_1, \ldots b_n \in pr(A).\, a =_A \bigvee_{i=1}^{n} b_i$$

$$(d2) \quad 1_A \in cpr(A)$$

$$(d3) \quad a, b \in pr(A) \;\Rightarrow\; a \wedge b \in pr(A)$$

We now introduce a notion of morphism for domain prelocales, based on Scott's *approximable mappings* [Sco81, Sco82].

Definition 3.2.4 Let A, B, be domain prelocales. An *approximable mapping* $R : A \to B$ is a relation $R \subseteq |A| \times |B|$ satisfying

$$(r1) \quad aR1$$

$$(r2) \quad aRb \,\&\, aRc \;\Rightarrow\; aR(b \wedge c)$$

$$(r3) \quad 0Rb$$

$$(r4) \quad aRc \,\&\, bRc \;\Rightarrow\; (a \vee b)Rc$$

$$(r5) \quad a \leq a'Rb' \leq c \;\Rightarrow\; aRb$$

$$(r6) \quad aR0 \;\Rightarrow\; a =_A 0$$

$$(r7) \quad a \in pr(A) \,\&\, aR(b \vee c) \;\Rightarrow\; aRb \text{ or } aRc.$$

Approximable mappimgs are closed under relational composition. We verify the least trivial closure condition, $(r7)$. Suppose $R : A \to B$, $S : B \to C$, $a \in pr(A)$ and $a(R \circ S)b \vee c$. For some $d \in |B|$, aRd and $dSb \vee c$. By $(d1)$,

$$d =_B \bigvee_{i \in I} d_i \quad (d_i \in pr(B), i \in I).$$

If $I = \varnothing$, $d =_B 0_B$, hence by $(r3)$ dRb, and so $a(R \circ S)b$. Otherwise, by $(r7)$, aRd_i for some $i \in I$. Now

$$d_i \leq \bigvee_{i \in I} d_i S(b \vee c)$$

$$\Rightarrow \quad d_i S(b \vee c) \quad (r5)$$

$$\Rightarrow \quad d_i S b \text{ or } d_i S c \quad (r7)$$

$$\Rightarrow \quad a(R \circ S)b \text{ or } a(R \circ S)c$$

as required. Identities with respect to this composition are given by

$$a \,\mathrm{id}_A\, b \;\equiv\; a \leq_A b.$$

Hence we can define a category **DPL** of domain prelocales and approximable mappings.

Definition 3.2.5 A *pre-isomorphism* $\varphi : A \simeq B$ of domain prelocales is a surjective function

$$\varphi : |A| \to |B|$$

satisfying

$$\forall a, b \in |A|.\, a \leq_A b \;\Leftrightarrow\; \varphi(a) \leq_B \varphi(b).$$

Proposition 3.2.6 *If $\varphi : A \simeq B$ is a preisomorphism, the relation*

$$aR_\varphi b \;\equiv\; \varphi(a) \leq_B b$$

is an isomorphism in **DPL**. ∎

Theorem 3.2.7 **DPL** *is equivalent to* **SDom**.

Proof. We define functors

$$F : \mathbf{SDom} \to \mathbf{DPL}$$

$$G : \mathbf{DPL} \to \mathbf{SDom}$$

as follows:

$$F(D) = (K\Omega(D), \subseteq, =, \varnothing, \cup, D, \cap)$$

i.e. the distributive lattice of compact-open subsets of D;

$$F(f) = R_f,$$

where

$$aR_f b \ \equiv \ a \subseteq f^{-1}(b).$$

The verification that F is well-defined is routine. Note that:

- $pr(F(D)) = \{\uparrow u : u \in K(D)\} \cup \{\varnothing\}$
- $a \in con(F(D)) \ \Leftrightarrow \ a \neq \varnothing$
- $\uparrow u \cap \uparrow v \in con(F(D)) \ \Leftrightarrow \ u \triangle \ v$

To verify $(r7)$ for R_f, note that, for $u \in K(D)$:

$$\begin{aligned}\uparrow u \subseteq f^{-1}(b \cup c) \ &\Leftrightarrow \ u \in f^{-1}(b \cup c) \\ &\Leftrightarrow \ f(u) \in b \cup c \\ &\Leftrightarrow \ f(u) \in b \text{ or } f(u) \in c \\ &\Leftrightarrow \ \uparrow u \subseteq f^{-1}(b) \text{ or } \uparrow u \subseteq f^{-1}(c).\end{aligned}$$

$$G(A) \equiv \hat{A},$$

where $\hat{A}$ is the set of prime proper filters of A, i.e. sets $x \subseteq |A| - \{0_A\}$ closed under finite conjunction and entailment and satisfying

$$a \vee b \in x \ \Rightarrow \ a \in x \text{ or } b \in x.$$

$\hat{A}$ is a partial order under set inclusion; or, equivalently, (via the specialisation order) a topological space with basic opens

$$U_a \equiv \{x \in \hat{A} : a \in x\} \ \ (a \in |A|).$$

Note that, with either structure,

$$\hat{A} \ \cong \ \text{Spec } \tilde{A}.$$

$$G(R) = f_R,$$

where

$$f_R(x) = \{b \mid \exists a \in x.\, aRb\}.$$

We check that G is well defined. By $(d2)$, the filter generated by 1 is prime, hence a least element for $\hat{A}$; while it is easy to see that $\hat{A}$ is closed under unions of directed families. Thus $\hat{A}$ is a cpo. Moreover, the principal filters $\uparrow(a)$ with $a \in cpr(A)$ are prime, and (using $(d1)$) form a basis of finite elements. Finally, by $(d3)$ this basis is closed under consistent finite joins. Thus $\hat{A}$ is a Scott domain.

Now we check that f_R is well defined and continuous. Given $x \in \hat{A}$, it is easy to see that $f_R(x)$ is a filter. To check that it is prime, suppose $b \vee c \in f_R(x)$. Then for some $a \in x$, we must have $aR(b \vee c)$. By $(d1)$,

$$a =_A \bigvee_{i \in I} a_i, \quad (a_i \in cpr(A), i \in I).$$

Since x is a proper filter, $a \neq 0$, hence $I \neq \varnothing$. Then since x is prime, for some $i \in I$ $a_i \in x$. Now by $(r7)$,

$$a_i R(b \vee c) \ \Rightarrow \ a_i Rb \text{ or } a_i Rc$$

and so $b \in f_R(x)$ or $c \in f_R(x)$. Since directed joins in $\hat{A}$ are just unions, continuity of f_R is trivial.

The remainder of the verification that G is a functor is routine.

We now define natural transformations

$$\eta : I_{\mathbf{SDom}} \to GF$$

$$\epsilon : I_{\mathbf{DPL}} \to FG$$

$$\eta D(d) = \{U \in K\Omega(D) : d \in U\}$$

$$\epsilon A = R_{\varphi A},$$

where $\varphi A : A \simeq K\Omega(\hat{A})$ is the pre-isomorphism defined by

$$\varphi A(a) = \{x \in \hat{A} : a \in x\}.$$

Note that η, φ are the natural isomorphisms in the Stone duality for distributive lattices. This shows that the components of η, ϵ are isomorphisms, while naturality is easily checked to extend to our setting.

Altogether, we have shown that

$$(F, G, \eta, \epsilon) : \mathbf{SDom} \simeq \mathbf{DPL}$$

is an equivalence of categories. ∎

3.3 A Cpo of Pre-locales

In this section, we follow the ideas of Larsen and Winskel [LW84], and define a (large) cpo of domain pre-locales, in such a way that type constructions can be represented as continuous functions over this cpo, and the process of solving recursive domain equations reduced to taking least fixed points of such functions.

Definition 3.3.1 Let A, B be domain prelocales. Then we define $A \Subset B$ iff

- $|A| \subseteq |B|$
- $(|A|, 0_A, \vee_A, 1_A, \wedge_A)$ is a subalgebra of $(|B|, 0_B, \vee_B, 1_B, \wedge_B)$
- $\leq_A \subseteq \leq_B$

Although this inclusion relation is simple, it is too weak, and has only been introduced for organisational purposes. What we need is

Definition 3.3.2 $A \trianglelefteq B$ iff

$(s1)$ $A \Subset B$

$(s2)$ $\forall a, b \in |A|.\, a \leq_B b \;\Rightarrow a \leq_A b$

$(s3)$ $pr(A) \subseteq pr(B)$

Note that apart from $(s3)$ this is just the usual notion of *submodel* (cf. e.g. [CK73]).

Proposition 3.3.3 *The class of domain prelocales under $\trianglelefteq$ is an ω-chain complete partial order.*

Proof. The verification that $\trianglelefteq$ is a partial order is routine. Let $\{A_n\}$ be a $\trianglelefteq$-chain. Set

$$A_\infty \equiv (\bigcup_{n\in\omega} A_n, \bigcup_{n\in\omega} \leq_{A_n}, \ldots etc.).$$

We check that A_∞ is a well-defined domain prelocale, for in that case it is clearly the least upper bound of the chain. We verify $(d1)$ for illustration.

Given $a \in |A_\infty|$, for some n, $a \in |A_n|$, hence

$$a =_{A_n} \bigvee_{i \in I} a_i, \quad (a_i \in pr(A_n), i \in I).$$

Clearly $a =_{A_\infty} \bigvee_{i \in I} a_i$; furthermore, $pr(A_n) \subseteq pr(A_\infty)$. To see this, suppose $b \in pr(A_n)$ and $b \leq_{A_\infty} c \vee d$. For some $m \geq n$, $\{a, b, c\} \subseteq |A_m|$, and so $b \leq_{A_m} c \vee d$. Since $A_n \trianglelefteq A_m$, $pr(A_n) \subseteq pr(A_m)$, and so $b \leq_{A_m} c$ or $b \leq_{A_m} d$, which implies $b \leq_{A_\infty} c$ or $b \leq_{A_\infty} d$, as required. ∎

The class of domain prelocales is not a cpo under $\trianglelefteq$; it does not have a least element. However, we can easily remedy this deficiency.

Definition 3.3.4 $\mathbf{1}$ is the domain prelocale defined as follows. The carrier $|\mathbf{1}|$ is defined inductively by

- $t, f \in |\mathbf{1}|$
- $a, b \in |\mathbf{1}| \;\Rightarrow\; a \wedge b, a \vee b \in |\mathbf{1}|$

The operations are defined "freely" in the obvious way:

$$0_{\mathbf{1}} \equiv f, \quad 1_{\mathbf{1}} \equiv t, \quad a \vee_{\mathbf{1}} b \equiv a \vee b, \quad a \wedge_{\mathbf{1}} b \equiv a \wedge b$$

Finally, $\leq_{\mathbf{1}}$, $=_{\mathbf{1}}$ are defined inductively as the least relations satisfying ($p1$)–($p4$). It is easy to see that $\tilde{\mathbf{1}}$ is the two-point lattice; hence $\mathbf{1}$ is a domain prelocale.

Now let **DPL1** be the class of domain prelocales A such that $\mathbf{1} \trianglelefteq A$. Clearly **DPL1** is still chain-complete. Thus we have

Proposition 3.3.5 **DPL1** *is a large cpo with least element* $\mathbf{1}$. ∎

DPL1 also determines a full subcategory of **DPL**. To see that we are not losing anything in passing from **DPL** to **DPL1**, we note

Proposition 3.3.6 **DPL1** *is equivalent to* **DPL**. ∎

We now relate this partial order of prelocales to the category of domains and embeddings used in the standard category-theoretic treatment of the solution of domain equations [SP82]. Recall that an *embedding-projection* *pair* between domains D, E is a pair of continuous functions $e : D \to E$, $p : E \to D$ satisfying

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

$$e \circ p \sqsubseteq \mathsf{id}_E.$$

Each of these functions uniquely determines the other, since e is left adjoint to p. We write e^R for the projection determined by e.

Proposition 3.3.7 *If $A \trianglelefteq B$, then $e : \hat{A} \to \hat{B}$ is an embedding, where*

$$e : x \mapsto \uparrow_B(x).$$

($\hat{A}$, $\hat{B}$ are defined as in the proof of Theorem 3.2.7).

PROOF. We define $p : \hat{B} \to \hat{A}$ by

$$p(y) = y \cap |A|.$$

Since A is a sublattice of B, p is well defined and continuous (it is the surjection corresponding under Stone duality to the inclusion of A in B). We check that e is well defined, specifically that $e(x)$ is prime, $x \in \hat{A}$. Suppose $b \vee c \in e(x)$. Then for some $a \in x$, $a \leq_B b \vee c$. By $(d1)$,

$$a =_A \bigvee_{i \in I} a_i, \quad (a_i \in pr(A), i \in I).$$

Since x is a prime proper filter, $a_i \in x$ for some $i \in I$. Since $A \trianglelefteq B$, $a_i \in pr(B)$, and so

$$\begin{aligned} a_i \leq_B a \leq_B b \vee c \quad &\Rightarrow \quad a_i \leq_B b \text{ or } a_i \leq_B c \\ &\Rightarrow \quad b \in e(x) \text{ or } c \in e(x). \end{aligned}$$

Moreover,

$$p \circ e(x) = \uparrow_B(x) \cap |A| = x$$

$$e \circ p(y) = \uparrow_B(y \cap |A|) \subseteq \uparrow_B(y) = y.$$

Finally, e preserves all joins since it is a left adjoint; in particular, it is continuous. ∎

Now given a (unary) type construction T, we will seek to represent it as a function

$$f_T : \mathbf{DPL1} \to \mathbf{DPL1}$$

which is $\trianglelefteq$-monotonic and chain continuous. We can then construct the initial solution of the domain equation

$$D = T(D)$$

as the least fixpoint of the function f_T, given in the usual way as

$$\bigsqcup_{n \in \omega} f_T^{(n)}(\mathbf{1}).$$

More generally, we can consider systems of domain equations by using powers of **DPL1**; while T can be built up by composition from various primitive operations. As long as each basic type construction is $\trianglelefteq$-monotonic and continuous, this approach will work.

The task of verifying continuity is eased by the following observation, adapted from [LW84].

Proposition 3.3.8 *Suppose* $f : \mathbf{DPL1} \to \mathbf{DPL1}$ *is* $\trianglelefteq$*-monotonic and continuous on carriers, i.e. given a chain* $\{A_n\}_{n \in \omega}$*,*

$$|f(\bigsqcup_{n \in \omega} A_n| = \bigcup_{n \in \omega} |f(A_n)|,$$

then f *is continuous.*

PROOF. Firstly, note that $A \trianglelefteq B$ and $|A| = |B|$ implies $A = B$. Now given a chain $\{A_n\}$, let

$$B \equiv \bigsqcup_n f(A_n),$$

$$C \equiv f(\bigsqcup_n A_n).$$

By monotonicity of f, $B \trianglelefteq C$, while by continuity on carriers, $|B| = |C|$. Hence $B = C$, and f is continuous. ∎

3.4 Constructions

In this section, we fill in the programme outlined in the previous section by defining a number of type constructions as $\trianglelefteq$-monotonic and continuous functions over **DPL1**. These definitions will follow a common pattern. We take a binary type construction $T(A, B)$ for illustration. Specific to each such construction will be a set of *generators* $G(T(A, B))$. Then the carrier $|T(A, B)|$ is defined inductively by

- $G(T(A, B)) \subseteq |T(A, B)|$
- $t, f \in |T(A, B)|$
- $\dfrac{a, b \in |T(A, B)|}{a \wedge b, a \vee b \in |T(A, B)|}$

The operations $0, 1, \wedge, \vee$ are then defined "freely" in the obvious way, i.e.

$$0_{T(A,B)} \equiv f, \quad a \vee_{T(A,B)} b \equiv a \vee b, \quad 1_{T(A,B)} \equiv t, \quad a \wedge_{T(A,B)} b \equiv a \wedge b.$$

Finally, the relations $\leq_{T(A,B)}$, $=_{T(A,B)}$ are defined inductively as the least satisfying $(p1)$–$(p4)$ plus specific axioms on the generators. (Note that our definition of **1** in the previous section is the special case of this scheme where the set of generators is empty.)

As an essential part of the machinery for defining the type constructions, we shall introduce a number of meta-predicates over the carriers $|T(A, B)|$ of the constructed prelocales. These will be used as side-conditions on a number of axiom-schemes and rules. They will serve as "syntactic" analogues of the "semantic" predicates *con*, *pr*, *t* introduced previously. The same predicates will be defined for each contruction:

- PNF, prime normal form.
- CON, T, defined over elements of the form $\bigwedge_{i \in I} a_i$, with each a_i in PNF. CON is *consistency* (i.e. $\mathsf{CON}(a)$ means $a \neq 0$), and T is *termination* (i.e. $\mathsf{T}(a)$ means $a \neq 1$).
- CPNF, consistent prime normal forms, where $\mathsf{CPNF}(a)$ implies $\mathsf{PNF}(a)$ and $\mathsf{CON}(a)$.

Given these definitions, three further predicates are defined as follows:

- CDNF, consistent disjunctive normal form:

$$\mathsf{CDNF}(a) \ \equiv \ a = \bigvee_{i \in I} a_i \ \& \ \forall i \in I.\, \mathsf{CPNF}(a_i)$$

- $a\!\downarrow \ \equiv \ a = \bigvee_{i \in I} a_i \ \& \ \forall i \in I.\, \mathsf{PNF}(a_i) \ \& \ \mathsf{T}(a_i)$

- $\#(a) \ \equiv \ a = \bigvee_{i \in I} a_i \ \& \ \forall i \in I.\, \mathsf{PNF}(a_i) \ \& \ \neg\mathsf{CON}(a_i).$

It will follow from our general scheme of definition and the way that the generators are defined that the following points are immediate, for A, A', B, B' in **DPL1** with $A \trianglelefteq A'$ and $B \trianglelefteq B'$:

- $T(A,B)$ satisfies $(p1)$–$(p4)$
- $\mathbf{1} \trianglelefteq T(A,B)$
- $T(A,B) \Subset T(A',B')$
- T is continuous on carriers.

We are left to focus our attention on proving that:

- $T(A,B)$ satisfies $(d1)$–$(d3)$
- conditions $(s2)$ and $(s3)$ for $T(A,B) \trianglelefteq T(A',B')$ are satisfied.

Our method of establishing this for each T is uniform, and goes via another essential verification, namely that T does indeed correspond to the intended construction over domains. We define a semantic function

$$[\![\cdot]\!]_{T(A,B)} : |T(A,B)| \to K\Omega(F_T(\hat{A}, \hat{B}))$$

where F_T is the functor over **SDom** corresponding to T, and show that $[\![\cdot]\!]_{T(A,B)}$ is a (pre)isomorphism; and moreover natural with respect to embeddings induced by $\trianglelefteq$. This allows us to read off the required "proof-theoretic" facts about T from the known "model-theoretic" ones about F_T. Moreover, we can derive "soundness and completeness" theorems as byproducts.

For each type construction T, we prove the following sequence of results:
T1: Adequacy of Metapredicates. *For each* $a \in \mathsf{PNF}(T(A,B))$*:*

$$
\begin{array}{rl}
(i) & [\![a]\!]_{T(A,B)} \in pr(K\Omega(F_T(\hat{A},\hat{B}))) \\
(ii) & \mathsf{CON}(a) \iff [\![a]\!]_{T(A,B)} \neq \varnothing \\
(iii) & \mathsf{T}(a) \iff \bot_{F_T(\hat{A},\hat{B})} \notin [\![a]\!]_{T(A,B)}.
\end{array}
$$

T2: Normal Forms.

$$\forall a \in |T(A,B)|.\, \exists b \in \mathsf{CDNF}(T(A,B)).\, a =_{T(A,B)} b.$$

T3: Soundness. *For all* $a, b \in |T(A,B)|$*:*

$$a \leq_{T(A,B)} b \;\Rightarrow\; [\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)}.$$

T4: Prime Completeness. *For all* $a, b \in \mathsf{CPNF}(T(A,B))$*:*

$$[\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)} \;\Rightarrow\; a \leq_{T(A,B)} b.$$

T5: Definability.

$$\forall u \in K(F_T(\hat{A},\hat{B})).\, \exists a \in \mathsf{CPNF}(T(A,B)).\, [\![a]\!]_{T(A,B)} = \uparrow(u).$$

T6: Naturality. *Given* $A \trianglelefteq A'$*,* $B \trianglelefteq B'$ *in* **DPL1***, let* $e_1 : \hat{A} \to \hat{A}'$*,* $e_2 : \hat{B} \to \hat{B}'$ *be the corresponding embeddings. Given an embedding* $e : D \to E$*, let* $e^\dagger : K\Omega(D) \to K\Omega(E)$ *be defined by*

$$e^\dagger(\uparrow X) = \uparrow\{e(x) : x \in X\}$$

which is well defined since embeddings map finite elements to finite elements. Let

$$\eta_{T(A,B)} : \hat{C} \to F_T(\hat{A},\hat{B})$$

be the adjoint of $[\![\cdot]\!]_{T(A,B)}$*, where* $C = T(A,B)$*. Then:*

$$
\begin{array}{rrcl}
(A) & (F_T(e_1,e_2))^\dagger \circ [\![\cdot]\!]_{T(A,B)} & = & [\![\cdot]\!]_{T(A',B')} \\
(B) & F_T(e_1,e_2) \circ \eta_{T(A,B)} & = & \eta_{T(A',B')} \circ \downarrow_{T(A',B')}(\cdot)
\end{array}
$$

(These equations make sense since $T(A,B) \Subset T(A',B')$ *by assumption.)*

All the desired properties of our constructions can easily be derived from these results.

T7: Completeness. *For* $a, b \in |T(A,B)|$*:*

$$[\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)} \;\Rightarrow\; a \leq_{T(A,B)} b.$$

PROOF. By (T2),

$$a =_{T(A,B)} \bigvee_{i \in I} a_i, \quad b =_{T(A,B)} \bigvee_{j \in J} b_j,$$

with $a_i, b_j \in \mathsf{CPNF}(T(A,B))$ $(i \in I, j \in J)$. By (T3),

$$[\![a]\!]_{T(A,B)} = [\![\bigvee_{i \in I} a_i]\!]_{T(A,B)}, \quad [\![b]\!]_{T(A,B)} = [\![\bigvee_{j \in J} b_j]\!]_{T(A,B)}.$$

By (T1),

$$[\![a_i]\!]_{T(A,B)} = \uparrow(u_i), [\![b_j]\!]_{T(A,B)} = \uparrow(v_j)$$

$$u_i, v_j \in K(F_T(\hat{A}, \hat{B})) \quad (i \in I, j \in J).$$

Now,

$$\begin{array}{lll} & [\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)} & \\ \Longrightarrow & \bigcup_{i \in I} \uparrow(u_i) \subseteq \bigcup_{j \in J} \uparrow(v_j) & \\ \Longrightarrow & \forall i \in I. \exists j \in J. \uparrow(u_i) \subseteq \uparrow(v_j) & \\ \Longrightarrow & \forall i \in I. \exists j \in J. a_i \leq_{T(A,B)} b_j & \text{by (T4)} \\ \Longrightarrow & \bigvee_{i \in I} a_i \leq_{T(A,B)} \bigvee_{j \in J} b_j & \text{by (p2)} \\ \Longrightarrow & a \leq_{T(A,B)} b & \text{by (p1).} \blacksquare \end{array}$$

(T8): Stone Duality. $T(A,B)$ *is the Stone dual of* $F_T(\hat{A}, \hat{B})$*, i.e.*

(i) $F_T(\hat{A}, \hat{B}) \;\cong\; \hat{C} \quad (C = T(A,B))$

(ii) $[\![\cdot]\!] : |T(A,B)| \to K\Omega(F_T(\hat{A}, \hat{B}))$ is a pre-isomorphism.

PROOF. (i) and (ii) are equivalent since Scott domains are coherent. (ii) is an immediate consequence of (T3), (T5) and (T7). $\blacksquare$

(T9). *T is a well defined, $\trianglelefteq$-monotonic and continuous operation on* **DPL1**.
PROOF. T(A,B) is a domain prelocale by (T8), since $K\Omega(F_T(\hat{A}, \hat{B}))$ is. Given $A \trianglelefteq A'$, $B \trianglelefteq B'$, $T(A, B) \trianglelefteq T(A', B')$ follows from (T6)(A) and the following general properties of $e^\dagger$ for embeddings $e : D \to E$:

1. $e^\dagger$ is an order-mono, i.e. for $U, V \in K\Omega(D)$:

$$U \subseteq V \iff e^\dagger(U) \subseteq e^\dagger(V)$$

2. $e^\dagger$ preserves primes.

To prove (1), we take $U = \uparrow X$, $V = \uparrow Y$, and calculate:

$$\begin{aligned} \uparrow X \subseteq \uparrow Y &\iff X \sqsubseteq_u Y \\ &\iff e(X) \sqsubseteq_u e(Y) \quad \text{e is an order-mono} \\ &\iff \uparrow e(X) \subseteq \uparrow e(Y) \\ &\iff e^\dagger(U) \subseteq e^\dagger(V). \end{aligned}$$

For (2), we recall that $U \in pr(K\Omega(D))$ implies $U = \varnothing$ or $U = \uparrow(u)$ for some $u \in K(D)$. But $e^\dagger(\varnothing) = \varnothing$, $e^\dagger(\uparrow(u)) = \uparrow(e(u))$.

By the remarks at the beginning of the section, the proof is now complete. ∎

Notation. Given a domain prelocale A, we write

$$[\![\cdot]\!]_A : |A| \to K\Omega(\hat{A})$$

for the pre-isomorphism φA defined in the proof of Theorem 3.2.7.

We note a further trivial but useful fact about direct images of embeddings for future use.

Proposition 3.4.1 *If $A \trianglelefteq B$, and $e : \hat{A} \to \hat{B}$ is the induced embedding, then*

$$e^\dagger \circ [\![\cdot]\!]_A = [\![\cdot]\!]_B. \ ∎$$

Definition 3.4.2 The *function space* construction $A \to B$.
(i) The generators:

$$G(A \to B) \ \equiv \ \{(a \to b) : a \in |A|, b \in |B|\}.$$

This fixes $|A \to B|$ according to the general scheme described above.
(ii) The metapredicates:

$$\begin{aligned}
\mathsf{PNF}(A \to B) &\equiv \{\bigwedge_{i\in I}(a_i \to b_i) : a_i \in pr(A), b_i \in pr(B), i \in I\} \\
\mathsf{CON}(\bigwedge_{i\in I}(a_i \to b_i)) &\equiv \forall J \subseteq I. \\
&\quad \bigwedge_{j\in J} a_j \in con(A) \implies \bigwedge_{j\in J} b_j \in con(B) \\
\mathsf{T}(\bigwedge_{i\in I}(a_i \to b_i)) &\equiv \exists i \in I.\, a_i \in con(A) \& b_i \in t(B) \\
\mathsf{CPNF}(\bigwedge_{i\in I}(a_i \to b_i)) &\equiv \mathsf{CON}(\bigwedge_{i\in I}(a_i \to b_i)) \\
&\quad \& \ \forall i \in I.\, a_i \in con(A) \ \& \ b_i \in con(B)
\end{aligned}$$

The predicates CDNF, $\#(.)$, $\_\downarrow$ are then defined according to our general scheme.
(iii) The relations $\leq_{A\to B}$, $=_{A\to B}$ are then defined inductively by the following axioms and rules in addition to $(p1)$–$(p4)$ (subscripts omitted).

$$(\to - \leq) \quad \frac{a' \leq a, \ b \leq b'}{(a \to b) \leq (a' \to b')}$$

$$(\to - \wedge) \quad (a \to \bigwedge_{i\in I} b_i) = \bigwedge_{\in I}(a \to b_i)$$

$$(\to - \vee - L) \quad (\bigvee_{i\in I} a_i \to b) = \bigwedge_{i\in I}(a_i \to b)$$

$$(\to - \vee - R) \quad (a \to \bigvee_{i\in I} b_i) = \bigvee_{i\in I}(a \to b_i) \quad (a \in cpr(A))$$

$$(\#) \quad a \leq 0 \quad (\#(a))$$

(iv) The semantic function

$$[\![\cdot]\!]_{A\to B} : |A \to B| \longrightarrow K\Omega([\hat{A} \to \hat{B}])$$

is defined by

$$[\![(a \to b)]\!]_{A\to B} = ([\![a]\!]_A, [\![b]\!]_B)$$

where for spaces X, Y and subsets $U \in K\Omega(X)$, $V \in K\Omega(Y)$,

$$(U, V) \;\equiv\; \{f : X \to Y \mid f \text{ continuous}, \; f(U) \subseteq V\}$$

is a sub-basic open set in the compact-open topology. The further clauses

$$[\![\bigwedge_{i \in I} a_i]\!] = \bigcap_{i \in I} [\![a_i]\!]$$

$$[\![\bigvee_{i \in I} a_i]\!] = \bigcup_{i \in I} [\![a_i]\!]$$

will apply to all type constructions.

We will now establish that the function space construction satisfies (T1)–(T6) in a sequence of propositions.

Proposition 3.4.3 (T1) *For all $a \in \mathsf{PNF}(A \to B)$:*

$$\begin{array}{rl} (i) & [\![a]\!]_{A \to B} \in pr(K\Omega([\hat{A} \to \hat{B}])) \\ (ii) & \mathsf{CON}(a) \iff [\![a]\!]_{A \to B} \neq \varnothing \\ (iii) & \mathsf{T}(a) \iff \bot \notin [\![a]\!]_{A \to B}. \end{array}$$

Proof. (i) Let $a \in pr(A)$, $b \in pr(B)$. If $a \notin con(A)$,

$$[\![(a \to b)]\!]_{A \to B} = [\hat{A} \to \hat{B}] = 1_{K\Omega([\hat{A} \to \hat{B}])};$$

while if $a \in con(A)$, $b \notin con(B)$,

$$[\![(a \to b)]\!]_{A \to B} = \varnothing.$$

Otherwise, $a \in con(A)$ and $b \in con(B)$. Let $u = \uparrow(a)$, $v = \uparrow(b)$. Then $u \in K(\hat{A})$, $v \in K(\hat{B})$, and so

$$\begin{aligned} [\![(a \to b)]\!]_{A \to B} &= ([\![a]\!]_A, [\![b]\!]_B) \\ &= (\uparrow u, \uparrow v) \\ &= \uparrow [u, v], \end{aligned}$$

where $[u, v]$ is the step function in $[\hat{A} \to \hat{B}]$. Similarly, for $a_i \in cpr(A)$, $b_i \in cpr(B)$:

$$\begin{aligned}
[\![\bigwedge_{i \in I}(a_i \to b_i)]\!]_{A \to B} &= \bigcap_{i \in I} \uparrow [u_i, v_i] \\
&= \begin{cases} \uparrow(\bigsqcup_{i \in I}[u_i, v_i]) & \text{if } \triangle\{[u_i, v_i] : i \in I\} \\ \varnothing & \text{otherwise.} \end{cases}
\end{aligned}$$

(ii) Let $a = \bigwedge_{i \in I}(a_i \to b_i)$. We use the notation of (i). Suppose $\mathsf{CON}(a)$. Then for $i \in I$,

$$b_i \notin con(B) \implies a_i \notin con(A) \implies [\![(a_i \to b_i)]\!]_{A \to B} = 1_{K\Omega([\hat{A} \to \hat{B}])},$$

and so

$$\begin{aligned}
[\![a]\!]_{A \to B} &= [\![\bigwedge\{(a_j \to b_j) : a_j \in cpr(A), b_j \in cpr(B)\}]\!]_{A \to B} \\
&= \uparrow(\bigsqcup\{[u_j, v_j] : a_j \in cpr(A), b_j \in cpr(B)\}),
\end{aligned}$$

which is well-defined by 2.2.2. For the converse, suppose $\neg\mathsf{CON}(a)$. Then for some $J \subseteq I$, $\bigwedge_{j \in J} a_j \in con(A)$ and $\bigwedge_{j \in J} b_j \notin con(B)$. But then we have

$$[\![a]\!]_{A \to B} \subseteq [\![(\bigwedge_{j \in J} a_j \to \bigwedge_{j \in J} b_j)]\!]_{A \to B} = \varnothing.$$

(iii) With notation as in (ii),

$$\bot \notin [\![a]\!]_{A \to B} \iff \exists i \in I. \bot \notin [\![(a_i \to b_i)]\!]_{A \to B}.$$

Now if $a_i \notin con(A)$,

$$\bot \in 1_{K\Omega([\hat{A} \to \hat{B}])} = [\![(a_i \to b_i)]\!]_{A \to B};$$

while if $a_i \in con(A)$, $b_i \notin con(B)$, then

$$\bot \notin \varnothing = [\![(a_i \to b_i)]\!]_{A \to B}.$$

Finally, if $a_i \in con(A)$ and $b_i \in con(B)$, then $[\![(a_i \to b_i)]\!]_{A \to B} = \uparrow[u_i, v_i]$, and

$$\bot \notin [\![(a_i \to b_i)]\!]_{A \to B} \iff v_i \neq \bot \iff b_i \in t(B).$$

Thus $\bot \notin [\![(a_i \to b_i)]\!]_{A \to B} \iff a_i \in con(A) \;\&\; b_i \in t(B)$. ∎

As corollaries we have:

(iv) $\mathsf{CPNF}(\bigwedge_{i\in I}(a_i \to b_i)) \implies [\![\bigwedge_{i\in I}(a_i \to b_i)]\!]_{A\to B} = \uparrow(\bigsqcup_{i\in I}[u_i, v_i])$, where $\uparrow u_i = [\![a_i]\!]_A, \uparrow v_i = [\![b_i]\!]_B, i \in I$.

(v) $\#(a) \iff [\![a]\!]_{A\to B} = \varnothing$.

(vi) $a\!\downarrow \iff \bot \notin [\![a]\!]_{A\to B}$.

Proposition 3.4.4 (T2) $\forall a \in |A \to B|. \exists b \in \mathsf{CDNF}(A \to B). a =_{A\to B} b$.

Proof. Using the distributive lattice laws, a can be put in the form

$$\bigvee_{i\in I} \bigwedge_{j\in J_i} (a_{ij} \to b_{ij}).$$

By $(d1)$, each a_{ij} is equal to

$$\bigvee_{k\in K_{ij}} c_k, \quad (c_k \in pr(A), k \in K_{ij}),$$

and each b_{ij} is equal to

$$\bigvee_{l\in L_{ij}} d_l, \quad (d_l \in pr(B), l \in L_{ij}).$$

Moreover, we may assume that $c_k \in con(A)$ for all $k \in K_{ij}$, since otherwise

$$\bigvee_{k\in K_{ij}} c_k =_A \bigvee_{k'\in K_{ij}-\{k\}} c_{k'},$$

and so any inconsistent disjuncts can be deleted; and similarly for the d_l. Now

$$\begin{aligned}(\bigvee_{k\in K_{ij}} c_k \to \bigvee_{l\in L_{ij}} d_l) &=_{A\to B} \bigwedge_{k\in K_{ij}} (c_k \to \bigvee_{l\in L_{ij}} d_l) \quad \text{by } (\to - \vee - L)\\ &=_{A\to B} \bigwedge_{k\in K_{ij}} \bigvee_{l\in L_{ij}} (c_k \to d_l) \quad \text{by } (\to - \vee - R).\end{aligned}$$

Using the distributive lattice laws again, we obtain the required normal form. ∎

Proposition 3.4.5 (T3) $\forall a, b \in |A \to B|. a \leq_{A\to B} \Rightarrow [\![a]\!]_{A\to B} \subseteq [\![b]\!]_{A\to B}$.

PROOF. $[\![]\!]_{A\to B}$ preserves meets and joins by definition, and $(p1)$–$(p4)$ are valid in any distributive lattice. Moreover, given any spaces X, Y and subsets $U \subseteq X$, $V \subseteq Y$,

$$U' \subseteq U, V \subseteq V' \iff (U,V) \subseteq (U',V')$$

$$(U, \bigcap_{i\in I} V_i) = \bigcap_{i\in I}(U, V_i)$$

$$(\bigcup_{i\in I} U_i, V) = \bigcap_{i\in I}(U_i, V)$$

are simple set-theoretic calculations. The soundness of ($\to$-#) follows from Corollary (v) to Proposition 3.4.3. Finally, suppose $a \in cpr(A)$. Then $[\![a]\!]_A = \uparrow u$ with $u \in K(\hat{A})$, and

$$\begin{aligned}
[\![(a \to \bigvee_{i\in I} b_i)]\!]_{A\to B} &= (\uparrow u, \bigcup_{i\in I}[\![b_i]\!]_B) \\
&= \{f : f(u) \in \bigcup_{i\in I}[\![b_i]\!]_B\} \quad \text{by monotonicity} \\
&= \bigcup_{i\in I}\{f : f(u) \in [\![b_i]\!]_B\} \\
&= \bigcup_{i\in I}(\uparrow u, [\![b_i]\!]_B) \\
&= [\![\bigvee_{i\in I}(a \to b_i)]\!]_{A\to B}
\end{aligned}$$

and so $(\to - \vee - R)$ is sound. ∎

Proposition 3.4.6 (T4) *For* $\bigwedge_{i\in I}(a_i \to b_i)$, $\bigwedge_{j\in J}(a_j \to b_j)$ *in* $\mathsf{CPNF}(A \to B)$*:*

$$[\![\bigwedge_{i\in I}(a_i \to b_i)]\!]_{A\to B} \subseteq [\![\bigwedge_{j\in J}(a_j \to b_j)]\!]_{A\to B}$$

implies

$$\bigwedge_{i\in I}(a_i \to b_i) \leq_{A\to B} \bigwedge_{j\in J}(a_j \to b_j).$$

PROOF. By Corollary (iv) to Proposition 3.4.3,

$$[\![\bigwedge_{i\in I}(a_i \to b_i)]\!]_{A\to B} = \uparrow \bigsqcup_{i\in I}[u_i, v_i],$$

$$[\![\bigwedge_{j\in J}(a_j \to b_j)]\!]_{A\to B} = \uparrow \bigsqcup_{j\in J}[u_j, v_j],$$

where

$$\uparrow u_i = [\![a_i]\!]_A, \ldots \ etc.$$

Now,

$$[\![\bigwedge_{i\in I}(a_i \to b_i)]\!]_{A\to B} \subseteq [\![\bigwedge_{j\in J}(a_j \to b_j)]\!]_{A\to B}$$

$$\iff \bigsqcup_{j\in J}[u_j, v_j] \sqsubseteq \bigsqcup_{i\in I}[u_i, v_i]$$

$$\iff \forall j \in J.\, v_j \sqsubseteq \bigsqcup\{v_i : u_i \sqsubseteq u_j\}$$

$$\iff \forall j \in J.\, [\![\bigwedge\{b_i : [\![a_j]\!]_A \subseteq [\![a_i]\!]_A\}]\!]_B \subseteq [\![b_j]\!]_B$$

$$\iff \forall j \in J.\ \bigwedge\{b_i : a_j \leq_A a_i\} \leq_B b_j \quad (*).$$

Thus, for all $j \in J$:

$$\begin{array}{llll} \bigwedge_{i\in I}(a_i \to b_i) & \leq_{A\to B} & \bigwedge\{(a_i \to b_i) : a_j \leq_A a_i\} & \text{by (p3)} \\ & \leq_{A\to B} & \bigwedge\{(a_j \to b_i) : a_j \leq_A a_i\} & \text{by } (\to - \leq) \\ & =_{A\to B} & (a_j \to \bigwedge\{b_i : a_j \leq_A a_i\}) & \text{by } (\to - \wedge) \\ & \leq_{A\to B} & (a_j \to b_j) & \text{by } (*) \end{array}$$

and so by $(p2)$

$$\bigwedge_{i\in I}(a_i \to b_i) \leq_{A\to B} \bigwedge_{j\in J}(a_j \to b_j). \quad \blacksquare$$

Proposition 3.4.7 (T5) $\forall U \in K\Omega([\hat{A} \to \hat{B}]).\, \exists a \in |A \to B|.\, [\![a]\!]_{A\to B} = U.$

PROOF. Directly from Propositions 2.4.2 and 3.4.3. ∎

Proposition 3.4.8 (T6) *Given $A \trianglelefteq A'$, $B \trianglelefteq B'$, let $e_1 : \hat{A} \to \hat{A}'$, $e_2 : \hat{B} \to \hat{B}'$ be the corresponding embeddings. Then*

$$(A) \quad (e_1 \to e_2)^\dagger \circ [\![\cdot]\!]_{A \to B} = [\![\cdot]\!]_{A' \to B'}$$

$$(B) \quad (e_1 \to e_2) \circ \eta_{A \to B} = \eta_{A' \to B'} \circ \downarrow(\cdot).$$

PROOF. Firstly, we recall the definition of $e_1 \to e_2$:

$$(e_1 \to e_2)(f) = e_2 \circ f \circ e_1^R,$$

where e_1^R is the right adjoint of e_1, i.e. the corresponding projection. Now in fact we can eliminate the use of the projection in describing $(e_1 \to e_2)^\dagger$, since we have

$$(e_1 \to e_2)(\bigsqcup_{i \in I}[u_i, v_i]) = \bigsqcup_{i \in I}[e_1(u_i), e_2(v_i)].$$

Indeed,

$$\begin{aligned}
& (e_1 \to e_2)(\bigsqcup_{i \in I}[u_i, v_i])(d) \\
= \quad & e_2 \circ \bigsqcup_{i \in I}[u_i, v_i] \circ e_1^R(d) \\
= \quad & e_2(\bigsqcup_{i \in I}\{v_i : u_i \sqsubseteq e_1^R(d)\}) \\
= \quad & e_2(\bigsqcup_{i \in I}\{v_i : e_1(u_i) \sqsubseteq d\}) \\
= \quad & \bigsqcup_{i \in I}\{e_2(v_i) : e_1(u_i) \sqsubseteq d\} \\
& (e_2 \text{ preserves joins since it is a left adjoint}) \\
= \quad & (\bigsqcup_{i \in I}[e_1(u_i), e_2(v_i)])(d).
\end{aligned}$$

Now for (A), given

$$a =_{A \to B} \bigvee_{i \in I} \bigwedge_{j \in J_i} (a_{ij} \to b_{ij}) \in \mathsf{CDNF}(A \to B),$$

we calculate

$$
\begin{aligned}
(e_1 \to e_2)^\dagger [\![a]\!]_{A\to B} &= \bigcup_{i\in I}\bigcap_{j\in J_i}(e_1^\dagger[\![a_{ij}]\!]_A, e_2^\dagger[\![b_{ij}]\!]_B) \\
&= \bigcup_{i\in I}\bigcap_{j\in J_i}([\![a_{ij}]\!]_{A'}, [\![b_{ij}]\!]_{B'}) \quad \text{by 3.4.1} \\
&= [\![a]\!]_{A'\to B'}.
\end{aligned}
$$

Similarly for (B) we have:

$$
\begin{aligned}
& (e_1 \to e_2)\circ \eta_{A\to B}(x) \\
= \ & \bigsqcup\{[u,v] : \exists (a\to b)\in x.\, \uparrow u = [\![a]\!]_A \;\&\; \uparrow v = [\![b]\!]_B\} \\
= \ & \bigsqcup\{[u,v] : \exists (a\to b)\in x.\, \uparrow u = [\![a]\!]_{A'} \;\&\; \uparrow v = [\![b]\!]_{B'}\} \\
= \ & \eta_{A'\to B'}(\downarrow(x)). \quad \blacksquare
\end{aligned}
$$

To illustrate the uniformity in our treatment of all the type constructions, we shall deal with two more: the upper or Smyth powerdomain, and the coalesced sum.

Definition 3.4.9 The *upper powerdomain* $P_u(A)$.
(i) The generators:

$$G(P_u(A)) \equiv \{\Box a | a \in |A|$$

(ii) Metapredicates:

$$
\begin{aligned}
\mathsf{PNF}(P_u(A)) &\equiv \{\Box \bigvee_{i\in I} a_i : a_i \in pr(A), i\in I\} \\
\mathsf{CON}(t) & \\
\mathsf{CON}(\bigwedge_{i\in I}\Box\bigvee_{j\in J_i} a_{ij}) &\equiv \exists f\in \prod_{i\in I} J_i.\ \bigwedge_{i\in I} a_{i,f(i)} \in con(A) \\
\mathsf{T}(\bigwedge_{i\in I}\Box\bigvee_{j\in J_i} a_{ij}) &\equiv \exists i\in I.\, \forall j\in J_i.\, a_{ij}\in t(A) \\
\mathsf{CPNF}(\Box\bigvee_{i\in I} a_i) &\equiv \mathsf{CON}(\Box\bigvee_{i\in I} a_i) \;\&\; I\neq\varnothing \\
& \quad \& \,\forall i\in I.\, a_i\in con(A)
\end{aligned}
$$

(iii) Axioms in addition to $(p1)$ – $(p4)$:

$$(\Box - \leq) \quad \frac{a \leq b}{\Box a \leq \Box b}$$

$$(\Box - \wedge) \quad \Box \bigwedge_{i \in I} a_i = \bigwedge_{i \in I} \Box a_i$$

$$(\Box - 0) \quad \Box 0 = 0$$

(iv) The semantic function:

$$[\![\cdot]\!]_{P_u(A)} : |P_u(A)| \longrightarrow K\Omega(P_u(\hat{A}))$$

$$[\![\Box a]\!]_{P_u(A)} = \{S \in P_u(\hat{A}) : S \subseteq [\![a]\!]_A\}$$

(The further clauses are the standard ones described in the definition of function space.)

Proposition 3.4.10 (T1) *For all* $a, \{a_i\}_{i \in I} \in \mathsf{PNF}(P_u(A))$*:*

$$(i) \quad [\![a]\!]_{P_u(A)} \in pr(K\Omega(P_u(A)))$$

$$(ii) \quad \mathsf{CON}(\textstyle\bigwedge_{i \in I} a_i) \iff [\![\textstyle\bigwedge_{i \in I} a_i]\!]_{P_u(A)} \neq \varnothing$$

$$(iii) \quad \mathsf{T}(\textstyle\bigwedge_{i \in I} a_i) \iff \bot \notin [\![\textstyle\bigwedge_{i \in I} a_i]\!]_{P_u(A)}$$

Proof. (i). Let $\Box \bigvee_{i \in I} a_i \in \mathsf{PNF}(P_u(A))$. Then either $\bigvee_{i \in I} a_i \notin con(A)$, and

$$[\![\Box \bigvee_{i \in I} a_i]\!]_{P_u(A)} = \varnothing \in pr(K\Omega(P_u(A)));$$

or for some $X \subseteq_{\mathsf{f}} \mathcal{K}(\hat{A})$, $X \neq \varnothing$ and

$$[\![\bigvee_{i \in I} a_i]\!]_A = \uparrow_{\hat{A}} X.$$

In the latter case,

$$\begin{aligned} [\![\Box \bigvee_{i \in I} a_i]\!]_{P_u(A)} &= \{S \in P_u(\hat{A}) : S \subseteq [\![\bigvee_{i \in I} a_i]\!]_A\} \\ &= \{S \in P_u(\hat{A}) : \uparrow_{\hat{A}} X \sqsubseteq_u S\} \\ &= \uparrow_{P_u(\hat{A})}([\![\bigvee_{i \in I} a_i]\!]_A). \end{aligned}$$

(ii) Firstly,

$$[\![\bigwedge_{i\in I} \Box \bigvee_{j\in J_i} a_{ij}]\!]_{P_u(A)} = [\![\Box \bigvee_{f\in\prod_{i\in I} J_i} \bigwedge_{i\in I} a_{i,f(i)}]\!]_{P_u(A)},$$

by $(\Box - \wedge)$ (see the proof of (T3)) and distributivity. Now by (i),

$$[\![\Box \bigvee_{f\in\prod_{i\in I} J_i} \bigwedge_{i\in I} a_{i,f(i)}]\!]_{P_u(A)} \neq \varnothing$$

$$\iff [\![\bigvee_{f\in\prod_{i\in I} J_i} \bigwedge_{i\in I} a_{i,f(i)}]\!]_A \neq \varnothing$$

$$\iff \exists f \in \prod_{i\in I} J_i. \bigwedge_{i\in I} a_{i,f(i)} \in con(A).$$

(iii) This follows from the fact that

$$\bot \notin [\![\Box a]\!]_{P_u(A)} \iff \bot \notin [\![a]\!]_A. \ \blacksquare$$

Proposition 3.4.11 (T2) $\forall a \in |P_u(A)|. \exists b \in \mathsf{CDNF}(P_u(A)). a =_{P_u(A)} b.$

Proof. We can use the distributive lattice laws to put a in the form

$$\bigvee_{i\in I} \bigwedge_{j\in J_i} \Box a_{ij}.$$

By $(d1)$, each a_{ij} can be written as

$$\bigvee_{k\in K_{ij}} b_k,$$

where each $b_k \in cpr(A)$. We can now use $(\Box - \wedge)$ and the distributive laws to obtain an expression of the form

$$\bigvee_{i'\in I'} \Box \bigvee_{l\in L_{i'}} c_l,$$

where each $c_l \in cpr(A)$. Moreover disjuncts with $L_{i'} = \varnothing$ can be deleted using $(\Box - 0)$. This yields the required normal form. $\blacksquare$

Proposition 3.4.12 (T3) *For all* $a, b \in |P_u(A)|$*:*

$$a \leq_{P_u(A)} b \implies [\![a]\!]_{P_u(A)} \subseteq [\![b]\!]_{P_u(A)}.$$

Proof. Given $U \in K\Omega(\hat{A}))$, define

$$\Box U \;\equiv\; \{S \in P_u(\hat{A}) : S \subseteq U\}.$$

Then

$$U \subseteq V \implies \Box U \subseteq \Box V,$$

$$\Box \bigcap_{i \in I} U_i = \bigcap_{i \in I} \Box U_i$$

are simple set calculations, which validate $(\Box - \leq)$ and $(\Box - \wedge)$. $(\Box - 0)$ is valid because the empty set is excluded from $P_u(\hat{A})$. (In fact, dropping $(\Box - 0)$ exactly corresponds to retaining the empty set). ∎

Proposition 3.4.13 (T4) *For all* $\Box a, \Box b \in \mathsf{CPNF}(P_u(A))$*:*

$$[\![\Box a]\!]_{P_u(A)} \subseteq [\![\Box b]\!]_{P_u(A)} \implies \Box a \leq_{P_u(A)} \Box b.$$

Proof. Using the description of $[\![\Box a]\!]_{P_u(A)}$, $[\![\Box b]\!]_{P_u(A)}$ from the proof of Proposition 3.4.10(i),

$$\begin{aligned} & [\![\Box a]\!]_{P_u(A)} \subseteq [\![\Box b]\!]_{P_u(A)} \\ \implies\ & [\![a]\!]_A \subseteq [\![b]\!]_A \\ \implies\ & a \leq_A b \\ \implies\ & \Box a \leq_{P_u(A)} \Box b \quad (\Box - \leq). \end{aligned}$$ ∎

Proposition 3.4.14 (T6(A)) *Let* $A \trianglelefteq B$*, with* $e : \hat{A} \to \hat{B}$ *the corresponding projection. Then*

$$(P_u(e))^{\dagger} \circ [\![\cdot]\!]_{P_u(A)} = [\![\cdot]\!]_{P_u(B)}.$$

PROOF. From the proof of Proposition 3.4.10(i), for $a \in con(A)$:

$$(*) \quad [\![\Box a]\!]_{P_u(A)} = \uparrow_{P_u(A)} [\![a]\!]_{P_u(A)},$$

while for $a \in con(A)$ we have, directly from the definitions,

$$(**) \quad P_u(e)([\![a]\!]_A) = e^{\dagger}([\![a]\!]_A).$$

Now given $a \in |P_u(A)|$, by 3.4.11

$$a =_{P_u(A)} \bigvee_{i \in I} \Box a_i, \quad (a_i \in con(A), i \in I),$$

and we can calculate:

$$\begin{array}{rcll}
P_u(e)^{\dagger}([\![a]\!]_{P_u(A)}) & = & \bigcup_{i \in I} P_u(e)^{\dagger}([\![\Box a_i]\!]_{P_u(A)}) & \\
& = & \bigcup_{i \in I} P_u(e)^{\dagger}(\uparrow_{P_u(\hat{A})} [\![a_i]\!]_A) & (*) \\
& = & \bigcup_{i \in I} \uparrow_{P_u(\hat{B})} (P_u(e) [\![a_i]\!]_A) & \\
& = & \bigcup_{i \in I} \uparrow_{P_u(\hat{B})} (e^{\dagger} [\![a_i]\!]_A) & (**) \\
& = & \bigcup_{i \in I} \uparrow_{P_u(\hat{B})} ([\![a_i]\!]_B) & 3.4.1 \\
& = & \bigcup_{i \in I} [\![\Box a_i]\!]_{P_u(B)} & (*) \\
& = & [\![a]\!]_{P_u(B)}. & \blacksquare
\end{array}$$

Definition 3.4.15 The *coalesced sum.*
(i) The generators:

$$G(A \oplus B) \equiv \{(a \oplus f) : a \in |A|\} \cup \{(f \oplus b) : b \in |B|\}.$$

(ii) Metapredicates:

$$\mathsf{PNF}(A \oplus B) \equiv \{(a \oplus f) : a \in pr(A)\} \cup \{(f \oplus b) : b \in pr(B)\} \cup \{t\}$$

$$\mathsf{CON}(t)$$

$$\begin{array}{rcl}
\mathsf{CON}(\bigwedge_{i \in I} (a_i \oplus f) \wedge \bigwedge_{j \in J} (f \oplus b_j)) & \equiv & \neg(\bigwedge_{i \in I} a_i \in t(A) \;\&\; \bigwedge_{j \in J} b_j \in t(B)) \\
& & \& \bigwedge_{i \in I} a_i \in con(A) \\
& & \& \bigwedge_{j \in J} b_j \in con(B)
\end{array}$$

$$\mathsf{T}(\bigwedge_{i\in I}(a_i \oplus f) \wedge \bigwedge_{j\in J}(f \oplus b_j)) \;\equiv\; \exists i \in I.\, a_i \in t(A) \text{ or } \exists j \in J.\, b_j \in t(B)$$

$$\mathsf{CPNF}(a) \;\equiv\; \mathsf{CON}(a)$$

(iii) Axioms:

$$(\oplus\!-\leq) \quad \frac{a \leq b}{(a \oplus f) \leq (b \oplus f)} \qquad \frac{a \leq b}{(f \oplus a) \leq (f \oplus b)}$$

$$(\oplus - \wedge) \quad \bigwedge_{i\in I}(a_i \oplus f) = (\bigwedge_{i\in I} a_i \oplus f) \qquad \bigwedge_{i\in I}(f \oplus a_i) = (f \oplus \bigwedge_{i\in I} a_i)$$

$$(\oplus - \vee) \quad \bigvee_{i\in I}(a_i \oplus f) = (\bigvee_{i\in I} a_i \oplus f) \qquad \bigvee_{i\in I}(f \oplus a_i) = (f \oplus \bigvee_{i\in I} a_i)$$

$$(\oplus - \#) \quad a \leq f \qquad (\#(a))$$

(iv) Semantic function:

$$[\![\cdot]\!]_{A\oplus B} : |A \oplus B| \longrightarrow K\Omega(\hat{A} \oplus \hat{B})$$

$$\begin{aligned} [\![(a \oplus f)]\!]_{A\oplus B} &= \{< 0, d >: d \in [\![a]\!]_A, d \neq \bot\} \\ &\quad \cup \{x \in \hat{A} \oplus \hat{B} : \bot \in [\![a]\!]_A\} \end{aligned}$$

$$\begin{aligned} [\![(f \oplus b)]\!]_{A\oplus B} &= \{< 1, d >: d \in [\![b]\!]_B, d \neq \bot\} \\ &\quad \cup \{x \in \hat{A} \oplus \hat{B} : \bot \in [\![b]\!]_B\} \end{aligned}$$

Proposition 3.4.16 (T1) *For all* $c, \{c_i\}_{i\in I} \in \mathsf{PNF}(A \oplus B)$*:*

(i) $[\![c]\!]_{A\oplus B} \in pr(K\Omega(\hat{A} \oplus \hat{B}))$

(ii) $\mathsf{CON}(\bigwedge_{i\in I} c_i) \iff [\![\bigwedge_{i\in I} c_i]\!]_{A\oplus B} \neq \varnothing$

(iii) $\mathsf{T}(\bigwedge_{i\in I} c_i) \iff \bot \notin [\![\bigwedge_{i\in I} c_i]\!]_{A\oplus B}$.

PROOF. (i) If $c = (a \oplus f)$, $a \in pr(A)$, we can distinguish three cases:
(1): $a \notin con(A)$. In this case,

$$[\![c]\!]_{A \oplus B} = \varnothing.$$

(2): $[\![a]\!]_A = 1_{K\Omega(\hat{A})} = \uparrow(\bot)$. In this case,

$$[\![c]\!]_{A \oplus B} = \uparrow(\bot) \in pr(K\Omega(\hat{A} \oplus \hat{B})).$$

(3): $a \in con(A)$, $\bot \notin [\![a]\!]_A$. In this case, for some $u \in K(\hat{A})$, $u \neq \bot$, $[\![a]\!]_A = \uparrow u$. Then

$$\begin{aligned} [\![c]\!]_{A \oplus B} &= \{< 0, d >: u \sqsubseteq d\} \\ &= \uparrow_{\hat{A} \oplus \hat{B}}(< 0, u >). \end{aligned}$$

The case for $c = (f \oplus b)$ is similar.
(ii), (iii). Straightforward. ∎

Proposition 3.4.17 (T2) $\forall a \in |A \oplus B|. \exists b \in \mathsf{CDNF}(A \oplus B). a =_{A \oplus B} b.$

PROOF. We can use the distributive lattice laws to put a in the form

$$\bigvee_{i \in I} (\bigwedge_{j \in J_i} (a_{ij} \oplus f) \wedge \bigwedge_{k \in K_i} (f \oplus b_{ik})).$$

Moreover, we can write each a_{ij} as $\bigvee_{l \in L_{ij}} c_l$, b_{ik} as $\bigvee_{m \in M_{ik}} d_m$, with $c_l \in cpr(A)$, $d_m \in cpr(B)$. Using $(\oplus - \vee)$, we obtain

$$\bigvee_{i \in I'} (\bigwedge_{j \in J_i'} (a_{ij} \oplus f) \wedge \bigwedge_{k \in K_i'} (f \oplus b_{ik}))$$

with $a_{ij} \in cpr(A)$, $b_{ik} \in cpr(B)$. Now using $(\oplus - \wedge)$, we obtain

$$\bigvee_{i \in I'} ((\bigwedge_{j \in J_i'} a_{ij} \oplus f) \wedge (f \oplus \bigwedge_{k \in K_i'} b_{ik})).$$

For each $i \in I'$, if both

$$\bigwedge_{j \in J_i'} a_{ij} \in t(A)$$

and

$$\bigwedge_{k \in K_i'} b_{ik} \in t(B),$$

we may delete the i'th disjunct by $(\oplus - \#)$. If either

$$\bigwedge_{j \in J_i{}'} a_{ij} \notin con(A)$$

or

$$\bigwedge_{k \in K_i{}'} b_{ik} \notin con(B),$$

we can delete the i'th disjunct by $(\oplus - \vee)$. Otherwise, either

$$\bigwedge_{j \in J_i{}'} a_{ij} =_A 1_A$$

or

$$\bigwedge_{k \in K_i{}'} b_{ik} =_B 1_B,$$

and we can delete one of these conjuncts by $(\oplus - \wedge)$. In this way we obtain an expression of the form

$$\bigvee\{(a \oplus f)\} \vee \bigvee\{(f \oplus b)\},$$

with each $a \in cpr(A)$, $b \in cpr(B)$, as required. ∎

Proposition 3.4.18 (T4) *For all* $c, d \in \mathsf{CPNF}(A \oplus B)$*:*

$$[\![c]\!]_{A\oplus B} \subseteq [\![d]\!]_{A\oplus B} \implies c \leq_{A\oplus B} d.$$

Proof. Take $c = (a \oplus f)$. We consider two subcases.
(1): $d = (b \oplus f)$.

$$\begin{array}{rcl} [\![c]\!]_{A\oplus B} \subseteq [\![d]\!]_{A\oplus B} & \implies & [\![a]\!]_A \subseteq [\![b]\!]_A \\ & \implies & a \leq_A b \\ & \implies & (a \oplus f) \leq_{A\oplus B} (b \oplus f) \quad \text{by } (\oplus - \leq). \end{array}$$

(2): $d = (f \oplus b)$.

$$\begin{array}{rcl} [\![c]\!]_{A\oplus B} \subseteq [\![d]\!]_{A\oplus B} & \implies & \bot \in [\![b]\!]_B \\ & \implies & t \leq_B b \\ & \implies & c \leq_{A\oplus B} t \\ & & =_{A\oplus B} (f \oplus t) \quad (\oplus - \wedge) \\ & & \leq_{A\oplus B} (f \oplus b) \quad (\oplus - \leq). \end{array}$$

The case for $c = (f \oplus a)$ is similar. ∎

3.5 Logical Semantics of Types

We now build on the work of the previous sections to give a *logical semantics* for a language of type expressions, in which each type is interpreted as a propositional theory (domain prelocale).

Syntax of Type Expressions

We define a set of type expressions TExp by

$$\sigma \quad ::= \quad \mathrm{OP}(\sigma_1, \ldots \sigma_n) \ (\mathrm{OP} \in \Sigma_n) \mid t \mid \mathsf{rec}\, t.\sigma$$

where t ranges over a set of type variables TVar, σ over type expressions, and $\Sigma = \{\Sigma_n\}_{n \in \omega}$ is a ranked alphabet of type constructors. For each such constructor $\mathrm{OP} \in \Sigma_n$, we assume we have an operation $\mathrm{op}^{\mathcal{L}} : \mathbf{DPL1}^n \to \mathbf{DPL1}$ which satisfies properties (T1) – (T6) from the previous section with respect to a functor $\mathrm{op}^{\mathcal{D}} : \mathbf{SDom}^n \to \mathbf{SDom}$.

Logical Semantics of Type Expressions

We define a semantic function

$$\mathcal{L} : \mathsf{TExp} \longrightarrow \mathsf{LEnv} \longrightarrow \mathbf{DPL1}$$

where LEnv is the set of type environments

$$\mathsf{TVar} \longrightarrow \mathbf{DPL1}$$

as follows:

$$\begin{array}{rcl} \mathcal{L}[\![\mathrm{OP}(\sigma_1, \ldots, \sigma_n)]\!]\rho & = & \mathrm{op}^{\mathcal{L}}(\mathcal{L}[\![\sigma_1]\!]\rho, \ldots, \mathcal{L}[\![\sigma_n]\!]\rho) \\ \mathcal{L}[\![t]\!]\rho & = & \rho t \\ \mathcal{L}[\![\mathsf{rec}\, t.\sigma]\!]\rho & = & \mathsf{fix}(F) = \bigsqcup_{k \in \omega} F^k(\mathbf{1}), \end{array}$$

where $F : \mathbf{DPL1} \to \mathbf{DPL1}$ is defined by

$$F(A) = \mathcal{L}[\![\sigma]\!]\rho[t \mapsto A].$$

We write $\mathcal{LA}(\sigma)\rho$ for $\tilde{A}$, where $A = \mathcal{L}[\![\sigma]\!]\rho$.

Denotational Semantics of Type Expressions

Similarly to the logical semantics, we define

$$\mathcal{D} : \mathsf{TExp} \longrightarrow \mathsf{DEnv} \longrightarrow \mathbf{SDom}$$

where $\mathsf{DEnv} = \mathsf{TVar} \longrightarrow \mathbf{SDom}$. In this semantics, each $\mathrm{OP} \in \Sigma_n$ is interpreted by the corresponding functor

$$\mathrm{op}^{\mathcal{D}} : (\mathbf{SDom}^{\mathbf{E}})^n \longrightarrow \mathbf{SDom}^{\mathbf{E}}$$

and $\mathsf{rec}\,t.\sigma$ as the inititial fixed point of the endofunctor $\mathbf{SDom}^{\mathbf{E}} \longrightarrow \mathbf{SDom}^{\mathbf{E}}$ induced from $t \mapsto \sigma(t)$. See [Plo81, Chapter 5] and [SP82, Nie84].

Theorem 3.5.1 (Stone Duality) *Let $\rho_L \in \mathsf{LEnv}$, $\rho_D \in \mathsf{DEnv}$ satisfy:*

$$\forall t \in \mathsf{TVar}.\, K\Omega(\rho_D t) \;\cong\; \rho_L t.$$

Then for any type expression σ, $\mathcal{LA}[\![\sigma]\!]\rho_L$ is the Stone dual of $\mathcal{D}[\![\sigma]\!]\rho_D$, i.e.

$$(i) \quad \mathcal{D}[\![\sigma]\!]\rho_D \;\cong\; \mathsf{Spec}\,\mathcal{LA}[\![\sigma]\!]\rho_L$$

$$(ii) \quad K\Omega(\mathcal{D}[\![\sigma]\!]\rho_D) \;\cong\; \mathcal{LA}[\![\sigma]\!]\rho_L.$$

Proof. Firstly, note that the two conclusions of the Theorem are equivalent, since Scott domains are coherent spaces. Thus it suffices to prove (i).

It will be convenient to consider systems of simultaneous domain equations

$$\left.\begin{array}{rcl} \xi_1 & = & \sigma_1(\xi_1, \ldots, \xi_n) \\ & \vdots & \\ \xi_n & = & \sigma_n(\xi_1, \ldots, \xi_n) \end{array}\right\} \qquad (3.1)$$

where each σ_i is a type expression not containing any occurrences of rec. It is standard that any $\sigma \in \mathsf{TExp}$ is equivalent to a system of equations of this form, in the sense that the denotation of σ is isomorphic to a component of the solution of such a system. Thus what we shall show is that $\hat{A} \cong D$, where A is the solution of 3.1 in **DPL1** and D is the solution in **SDom**. To make this more precise, we need some definitions.

Firstly, we define a diagram Δ^D in $(\mathbf{SDom}^E)^n$ as follows:

$$\Delta^D = (D_n, f_n)_{n\in\omega}$$

where

$$\begin{array}{rcl} D_0 & = & (\mathbf{1}^{\mathcal{D}}, \ldots, \mathbf{1}^{\mathcal{D}}) \\ D_{k+1} & = & (\mathcal{D}[\![\sigma_1]\!]\rho^D[\vec{\xi} \mapsto D_k], \ldots, \mathcal{D}[\![\sigma_n]\!]\rho^D[\vec{\xi} \mapsto D_k]) \end{array}$$

and $f_k : D_k \to D_{k+1}$ is defined as follows: f_0 is the unique morphism given by initiality of D_0 in $(\mathbf{SDom}^E)^n$;

$$f_{k+1} = (\mathcal{D}_m[\![\sigma_1]\!]\rho^D_m[\vec{\xi} \mapsto f_n], \ldots, \mathcal{D}_m[\![\sigma_n]\!]\rho^D_m[\vec{\xi} \mapsto f_n])$$

where $\mathcal{D}_m$ gives the morphism part of the functor corresponding to σ, and $\rho^D_m t = \mathsf{id}_{\rho^D t}$. Now it is standard that the solution of 3.1 in **SDom** is given by

$$\lim_{\to} \Delta^D.$$

Similarly, we define a $\trianglelefteq$–chain $\{A_n\}$ in $\mathbf{DPL1}^n$ by

$$\begin{array}{rcl} A_0 & = & (\mathbf{1}^{\mathcal{L}}, \ldots, \mathbf{1}^{\mathcal{L}}) \\ A_{k+1} & = & (\mathcal{L}[\![\sigma_1]\!]\rho^L[\vec{\xi} \mapsto A_k], \ldots, \mathcal{L}[\![\sigma_n]\!]\rho^L[\vec{\xi} \mapsto A_k]) \end{array}$$

and we let Δ^L be the diagram $(\hat{A}_k, e_k)$ in $(\mathbf{SDom}^E)^n$, where $e_k : \hat{A}_k \to \hat{A}_{k+1}$ is the tuple of embeddings

$$e_{k,i} : \hat{A}_{k,i} \to \hat{A}_{k+1,i} \quad (1 \leq i \leq n)$$

induced by $A_{k,i} \trianglelefteq A_{k+1,i}$. Now the solution of 3.1 in **DPL1** is given by

$$A_\infty = \bigsqcup_k A_k = (\bigsqcup_k A_{k,1}, \ldots, \bigsqcup_k A_{k,n}).$$

It is easily verified that the cone $\mu : \Delta^L \to \hat{A}_\infty$ with μ_k the embedding induced by $A_k \trianglelefteq A_\infty$ is colimiting in $(\mathbf{SDom}^E)^n$. Thus our task reduces to proving

$$\lim_{\to} \Delta^L \cong \lim_{\to} \Delta^D,$$

for which it suffices to construct a natural isomorphism $\nu : \Delta^L \cong \Delta^D$.

We fix $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ as the system of equations under consideration. For each $\vec{\tau} = (\tau_1, \ldots, \tau_n)$ where each τ_i contains no occurrences of rec, and $k \in \omega$, we shall define:

- objects $D_{\vec{\tau},k}$ and morphisms

$$f_{\vec{\tau},k} : D_{\vec{\tau},k} \to D_{\vec{\tau},k+1}$$

in $(\mathbf{SDom}^E)^n$;

- objects $A_{\vec{\tau},k}$ in $\mathbf{DPL1}^n$ and morphisms

$$e_{\vec{\tau},k} : \hat{A}_{\vec{\tau},k} \to \hat{A}_{\vec{\tau},k+1}$$

- morphisms $\nu_{\vec{\tau},k} : \hat{A}_{\vec{\tau},k} \to D_{\vec{\tau},k}$.

$$D_{\vec{\tau},0} = (\mathbf{1}^{\mathcal{D}}, \ldots, \mathbf{1}^{\mathcal{D}}); \quad A_{\vec{\tau},0} = (\mathbf{1}^{\mathcal{L}}, \ldots, \mathbf{1}^{\mathcal{L}})$$

$$D_{\vec{\tau},k+1} = (\mathcal{D}[\![\tau_1]\!]\rho^D[\vec{\xi} \mapsto D_{\vec{\sigma},k}], \ldots, \mathcal{D}[\![\tau_n]\!]\rho^D[\vec{\xi} \mapsto D_{\vec{\sigma},k}])$$

$$A_{\vec{\tau},k+1} = (\mathcal{L}[\![\tau_1]\!]\rho^L[\vec{\xi} \mapsto A_{\vec{\sigma},k}], \ldots, \mathcal{L}[\![\tau_n]\!]\rho^L[\vec{\xi} \mapsto A_{\vec{\sigma},k}])$$

$f_{\vec{\tau},0}$ is the unique morphism given by initiality.

$$f_{\vec{\tau},k+1} = (\mathcal{D}_m[\![\tau_1]\!]\rho^D[\vec{\xi} \mapsto f_{\vec{\sigma},k}], \ldots, \mathcal{D}_m[\![\tau_n]\!]\rho^D[\vec{\xi} \mapsto f_{\vec{\sigma},k}])$$

$e_{\vec{\tau},k+1}$ is the embedding induced by

$$A_{\vec{\tau},k} \trianglelefteq A_{\vec{\tau},k+1}$$

which holds since $A_{\vec{\sigma},k} \trianglelefteq A_{\vec{\sigma},k+1}$ by the usual argument. $\nu_{\vec{\tau},0}$ is the unique isomorphism arising from $\hat{\mathbf{1}}^{\mathcal{L}} \cong \mathbf{1}^{\mathcal{D}}$.

$$\nu_{\vec{\tau},k+1} = (\nu_{\tau_1,k+1}, \ldots, \nu_{\tau_n,k+1}),$$

where $\nu_{\tau,k+1}$ is defined by induction on τ:

$$\nu_{\xi_i,k+1} = \nu_{\sigma_i,k}$$

$$\nu_{t,k+1} = \hat{\rho}^L t \cong \rho^D t,$$

the isomorphism given in the hypothesis of the theorem. For $\tau = \mathrm{OP}(\theta_1, \ldots, \theta_m)$,

$$\nu_{\tau,k+1} = \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+1}, \ldots, \nu_{\theta_m,k+1}) \circ \eta_{\tau,k+1},$$

where $\eta_{\tau,k+1} : \hat{A}_{\tau,k+1} \cong \mathrm{op}^{\mathcal{D}}(\hat{A}_{\theta_1,k+1}, \ldots, \hat{A}_{\theta_m,k+1})$ is the isomorphism given by property (T6)(B) for OP.

Note that

$$\Delta^D = (D_{\vec{\sigma},k}, f_{\vec{\sigma},k})_{k\in\omega},$$

$$\Delta^L = (\hat{A}_{\vec{\sigma},k}, e_{\vec{\sigma},k})_{k\in\omega},$$

and so, defining $\nu : \Delta^L \to \Delta^D$ by $\nu_k \equiv \nu_{\vec{\sigma},k}$, it remains to verify that for all k:

- ν_k is an isomorphism
- $\nu_{k+1} \circ e_k = f_k \circ \nu_k$.

We argue by induction on k. The basis follows from the fact that $\hat{\mathbf{1}}^{\mathcal{L}} \cong \mathbf{1}^{\mathcal{D}}$, and the initiality of $(\mathbf{1}^{\mathcal{D}}, \ldots, \mathbf{1}^{\mathcal{D}})$ in $(\mathbf{SDom}^E)^n$. For the inductive step, we assume:

(*i*) $\nu_k = \nu_{\vec{\sigma},k}$ is an isomorphism

(*ii*) $\nu_{k+1} \circ e_k = \nu_{\vec{\sigma},k+1} \circ e_{\vec{\sigma},k} = f_{\vec{\sigma},k} \circ \nu_{\vec{\sigma},k} = f_k \circ \nu_k$

and prove that for all τ with no occurrences of rec,

(*iii*) $\nu_{\tau,k+1}$ is an isomorphism

(*iv*) $\nu_{\tau,k+2} \circ e_{\tau,k+1} = f_{\tau,k+1} \circ \nu_{\tau,k+1}$

(where $(e_{\tau,k+1}, \ldots, e_{\tau,k+1}) = e_{(\tau,\ldots,\tau),k+1}$, and similarly for $f_{\tau,k+1}$). Taking $\tau = \sigma_i$, $1 \le i \le n$ in (*iii*) and (*iv*) then yields

(*v*) $\nu_{k+1} = \nu_{\vec{\sigma},k+1}$ is an isomorphism

and

$$(vi)\quad \nu_{k+2} \circ e_{k+1} = \nu_{\vec{\sigma},k+2} \circ e_{\vec{\sigma},k+1} = f_{\vec{\sigma},k+1} \circ \nu_{\vec{\sigma},k+1} = f_{k+1} \circ \nu_{k+1},$$

as required. We prove (*iii*) and (*iv*) by induction on τ.

Case 1: $\tau = \xi_i$. In this case, (iii) just says that $\nu_{\sigma_i,k}$ is an isomorphism, and (iv) that

$$\nu_{\sigma_i,k+1} \circ e_{\sigma_i,k} = f_{\sigma_i,k} \circ \nu_{\sigma_i,k},$$

and we can use our outer induction hypothesis on k.
Case 2: $\tau = t$. In this case, τ denotes a constant functor, and

$$f_{\tau,k+1} = id_{D_{\tau,k+1}},$$

$$e_{\tau,k+1} = id_{\hat{A}_{\tau,k+1}},$$

$$\nu_{\tau,k+1} = \nu_{\tau,k+2} = (\hat{\rho}^L t \cong \rho^D t),$$

so (iii) and (iv) hold trivially.
Case 3: $\tau = \mathrm{OP}(\theta_1, \ldots, \theta_m)$. Applying our inner induction hypothesis to each θ_i, we have

(vii) $\nu_{\theta_i,k+1}$ is an isomorphism

$(viii)$ $\nu_{\theta i,k+2} \circ e_{\theta i,k+1} = f_{\theta_i,k+1} \circ \nu_{\theta_i,k+1}$.

By definition,

$$\nu_{\tau,k+1} = \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+1}, \ldots, \nu_{\theta_m,k+1}) \circ \eta_{\tau,k+1}.$$

Since $\mathrm{op}^{\mathcal{D}}$ is a functor, by (vii) $\mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+1}, \ldots, \nu_{\theta_m,k+1})$ is an isomorphism; while $\eta_{\tau,k+1}$ is given as an isomorphism by (T6)(B). This proves (iii). Finally,

$$\begin{aligned}
& \nu_{\tau,k+2} \circ e_{\tau,k+1} \\
= \; & \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2}, \ldots, \nu_{\theta_m,k+2}) \circ \eta_{\tau,k+2} \circ e_{\tau,k+1} \\
= \; & \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2}, \ldots, \nu_{\theta_m,k+2}) \circ \mathrm{op}^{\mathcal{D}}(e_{\theta_1,k+1}, \ldots, e_{\theta_m,k+1}) \circ \eta_{\tau,k+1} \\
& \text{by (T6)(B)} \\
= \; & \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2} \circ e_{\theta_1,k+1}, \ldots, \nu_{\theta_m,k+2} \circ e_{\theta_m,k+1}) \circ \eta_{\tau,k+1} \\
= \; & \mathrm{op}^{\mathcal{D}}(f_{\theta_1,k+2} \circ \nu_{\theta_1,k+1}, \ldots, f_{\theta_m,k+2} \circ \nu_{\theta_m,k+1}) \circ \eta_{\tau,k+1} \\
& \text{by } (viii) \\
= \; & \mathrm{op}^{\mathcal{D}}(f_{\theta_1,k+2}, \ldots, f_{\theta_m,k+2}) \circ \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+1}, \ldots, \nu_{\theta_m,k+1}) \circ \eta_{\tau,k+1} \\
= \; & f_{\tau,k+2} \circ \nu_{\tau,k+1},
\end{aligned}$$

which proves (iv). ∎

We finish with an observation that will be useful in the next Chapter. In our definitions of the constructions $A \to B$ etc. in section 4, we used the "semantic" predicates pr, con, t at the argument types A, B. Now suppose we are forming a theory as the denotation of a type expression, e.g. $\mathcal{L}[\![\sigma \to \tau]\!]\rho$; the arguments are $A = [\![\sigma]\!]\rho$, $B = [\![\tau]\!]\rho$. Then it makes sense to use the *syntactic* predicates $\mathsf{PNF}(A)$, $\mathsf{CON}(A)$, $\mathsf{T}(A)$ etc. in our definition of

$$A \to B = \mathcal{L}[\![\sigma \to \tau]\!]\rho.$$

Using properties (T1), (T2) and (T8) for each type construction, it is straightforward to prove the

Observation 3.5.2 *For all σ, ρ the same theory is obtained as $\mathcal{L}[\![\sigma]\!]\rho$ whether syntactic or semantic predicates are used in each application of a type construction.* ∎

Chapter 4

Domain Theory In Logical Form

4.1 Introduction

In this Chapter we shall complete the core of our research programme, as set out in Chapter 1. We shall introduce a meta-language for denotational semantics, give it a logical interpretation *via* the localic side of Stone duality, and relate this logical interpretation to the standard denotational one by showing that they are Stone duals of each other.

Denotational semantics is always based, more or less explicitly, on a typed functional meta-language. The types are interpreted as topological spaces (usually domains in the sense of Scott [Sco81, Sco82], but sometimes metric spaces, as in [dBZ82, Niv81]), while the terms denote elements of or functions between these spaces. A *program logic* comprises an assertion language of formulas for expressing properties of programs, and an interface between these properties and the programs themselves. Two main types of interface can be identified [Pnu77]:

Endogenous logic In this style, formulas describe properties pertaining to the "world" of a single program. Notation:

$$P \models \phi$$

where P is a program and ϕ is a formula. Examples: temporal logic as used e.g. in [Pnu77]; Hennessy-Milner logic [HM85]; type inference [DM82].

Exogenous logic Here, programs are embedded in formulas as *modal operators.* Notation:

$$[P]\phi$$

where P is now a program denoting a function or relation. Examples: dynamic logic [Har79, Pra81], including as special cases Hoare logic [Hoa69], since "Hoare triples" $\{\phi\}P\{\psi\}$ can be represented by

$$\phi \rightarrow [P]\psi,$$

and Dijkstra's wlp-calculus [Dij76], since $wlp(P, \psi)$ can be represented as $[P]\psi$. (Total correctness assertions can also be catered for; see [Har79].)

Extensionally, formulas denote sets of points in our denotational domains, i.e. ϕ is a syntactic description of $\{x : x \text{ satisfies } \phi\}$. Then $P \models \phi$ can be interpreted as $x \in U$, where x is the point denoted by P, and U is the set denoted by ϕ. Similarly, $[M]\phi$ can be interpreted as $f^{-1}(U)$, where f is the function denoted by M (and elaborations of this when M denotes a relation or multifunction). In this way, we can give a topological interpretation of program logic.

But this is not all: duality cuts both ways. We can also use it to give a *logical interpretation of denotational semantics.* Rather than starting with the denotational domains as spaces of points, and then interpreting formulas as sets of points, we can give an axiomatic presentation of the topologies on our spaces, viewed as abstract lattices (logical theories), and then reconstruct the points from the properties they satisfy. In other words, we can present denotational semantics in axiomatic form, as a logic of programs. This has a number of attractions:

- It unifies semantics and program logic in a general and systematic setting.

- It extends the scope of program logic to the entire range of denotational semantics – higher-order functions, recursive types, powerdomains etc.

- The syntactic presentation of recursive types, powerdomains etc. makes these constructions more "visible" and easier to calculate with.
- The construction of "points", i.e. denotations of computational processes, from the properties they satisfy is very compatible with work currently being done in a mainly operational setting in concurrency [HM85, Win80] and elsewhere [BC85], and offers a promising approach to unification of this work with denotational semantics.

The setting we shall take for our work in this Chapter is **SDom**, the category of Scott domains. The significance of this as far as the meta-language is concerned is that we omit the Plotkin powerdomain construction. However, this construction will be treated, in the context of a particular domain equation, in Chapter 5. Our reason for not including the Plotkin powerdomain, and extending the duality to **SFP**, is that this creates some additional technical complications, though certainly not insuperable ones; lack of time and energy supervened. For further discussion, see Chapter 7.

The remainder of the Chapter is organised as follows. In section 2, we interpret the types of our denotational meta-language as propositional theories. We can then apply the results of Chapter 3 to show that each such theory is the Stone dual of the domain obtained as the denotation of the type in the standard interpretation. In section 3, we extend the meta-language to include typed terms, i.e. *functional programs.* We extend our logic to an axiomatisation of the satisfaction relation $P \models \phi$ (P a term, ϕ a formula of the logic introduced in section 2), and prove that this axiomatisation is sound and complete with respect to the spatial interpretation $x \in U$, where x is the point denoted by P, and U the open set denoted by ϕ. In section 4, we consider an alternative formulation of the meta-language, in which terms are formed at the morphism level rather than the element level; the comparison between these formulations extends the standard one between λ-calculus (element level) and cartesian closed categories (morphism level). We find a pleasing correspondence between the two known, but hitherto quite unrelated, dichotomies:

cartesian closed categories		exogenous logic
vs.	$\sim$	*vs.*
λ-calculus		endogenous logic.

Our axiomatisation of the morphism-level language comprises an extended and generalised *dynamic logic* [Pra81, Har79]. We prove a restricted Completeness Theorem for this axiomatisation, and show that the general validity problem for this logic is undecidable. Finally, in section 5 we indicate how the results of this Chapter pave the way for a whole class of applications, and set the scene for the two case studies to be described in Chapters 5 and 6.

4.2 Domains as Propositional Theories

We begin by introducing the first part of a meta-language for denotational semantics, the *type expressions*, with syntax

$$\sigma \quad ::= \quad \mathbf{1} \mid \sigma \times \tau \mid \sigma \rightarrow \tau \mid \sigma \oplus \tau \mid (\sigma)_\perp \mid P_u\sigma \mid P_l\sigma \mid t \mid \mathsf{rec}\, t.\sigma$$

where t ranges over type variables, and σ, τ over type expressions.

The standard way of interpreting these expressions is as objects of **SDom** (more generally as cpo's, but **SDom** is closed under all the above constructions as a subcategory of **CPO**). Thus for each type expression σ we define a domain $\mathcal{D}(\sigma) = (D(\sigma), \sqsubseteq_\sigma)$ in **SDom**; $\sigma \times \tau$ is interpreted as product, $\sigma \rightarrow \tau$ as function space, $\sigma \oplus \tau$ as coalesced sum, $(\sigma)_\perp$ as lifting, $P_u\sigma$ and $P_l\sigma$ as the upper and lower (or Smyth and Hoare) powerdomains, and $\mathrm{rec}\, t.\sigma$ as the solution of the domain equation

$$t = \sigma(t),$$

i.e. as the initial fixpoint of an endofunctor over **SDom**. Other constructions (e.g. strict function space, smash product) can be added to the list.

So far, all this is standard ([Plo81, SP82]). Now we begin our alternative approach. For each type expression σ, we shall define a propositional theory $\mathcal{L}(\sigma) = (L(\sigma), \leq_\sigma, =_\sigma)$, where:

- $L(\sigma)$ is a set of formulae
- $\leq_\sigma$, $=_\sigma$ are the relations of logical *entailment* and *equivalence* between formulae.

$\mathcal{L}(\sigma)$ is defined inductively via formation rules, axioms and inference rules in the usual way.

Formation Rules

- $t, f \in L(\sigma)$
- $\dfrac{\phi, \psi \in L(\sigma)}{\phi \wedge \psi, \phi \vee \psi \in L(\sigma)}$
- $\dfrac{\phi \in L(\sigma),\ \psi \in L(\tau)}{(\phi \times \psi) \in L(\sigma \times \tau),\ (\phi \rightarrow \psi) \in L(\sigma \rightarrow \tau)}$

- $$\frac{\phi \in L(\sigma),\ \psi \in L(\tau)}{(\phi \oplus f),\ (f \oplus \psi) \in L(\sigma \oplus \tau)}$$
- $$\frac{\phi \in L(\sigma)}{(\phi)_\perp \in L((\sigma)_\perp)}$$
- $$\frac{\phi \in L(\sigma)}{\Box\phi \in L(P_u\sigma),\ \Diamond\phi \in L(P_l\sigma)}$$
- $$\frac{\phi \in L(\sigma[\mathsf{rec}\, t.\sigma/t])}{\phi \in L(\mathsf{rec}\, t.\sigma)}$$

We should think of $(\phi \rightarrow \psi)$, $\Box\phi$ etc. as "constructors" or "generators", which build basic formulae at complex types from arbitrary formulae at simpler types. Note that no constructors are introduced for recursive types; we are taking advantage of the observation, familiar from work on information systems [LW84], that if we work with preorders it is easy to solve domain equations up to *identity*.

Examples

We define separated sum as a derived operation:

$$\sigma + \tau \equiv (\sigma)_\perp \oplus (\tau)_\perp$$

Also, we define the Sierpinski space (two-point domain):

$$\mathbb{O} \equiv (\mathbf{1})_\perp$$

Now we construct a number of familiar semantic domains:

name	expression	description
B	$\mathbf{1} + \mathbf{1}$	flat domain of booleans
N	$\mathsf{rec}\, t.\, \mathbb{O} \oplus t$	flat domain of natural numbers
LN	$\mathsf{rec}\, t.\, \mathbf{1} + t$	lazy natural numbers
List(N)	$\mathsf{rec}\, t.\, \mathbf{1} + (\mathsf{N} \times t)$	lazy lists of eager numbers
CBN	$\mathsf{rec}\, t.\, \mathsf{N} + (t \rightarrow t)$	call-by-name untyped λ-calculus

Now we define some formulas in these types, to suggest how the expected structure emerges from the formal definitions.

name	formula	type
$\star$	$(t)_\perp$	$\mathbb{O}$
true	$(\star \oplus f)$	B
false	$(f \oplus \star)$	B
$\overline{0}$	$(\star \oplus f)$	N
$\overline{1}$	$(f \oplus \overline{0})$	N
$\overline{n+1}$	$(f \oplus \overline{n})$	N
nil	$(\star \oplus f)$	List(N)
$\overline{0}$ :: nil	$(f \oplus (\overline{0} \times \mathsf{nil}))$	List(N)
$\overline{0} :: \perp$	$(f \oplus (\overline{0} \times t))$	List(N)
parallel or	$((\mathsf{true} \times t) \to \mathsf{true})$	
	$\wedge\ ((t \times \mathsf{true}) \to \mathsf{true})$	
	$\wedge\ ((\mathsf{false} \times \mathsf{false}) \to \mathsf{false})$	$(\mathsf{B} \times \mathsf{B}) \to \mathsf{B}$

Auxiliary Predicates

Before proceeding to the axiomatisation proper, we shall define some auxiliary predicates on formulas. These will be used as side-conditions on a number of axioms and rules (e.g. $(\rightarrow - \vee - R)$ below). Thus it is important that they are recursive predicates, defined syntactically on formulae. The main predicates we define are:

- $\mathsf{PNF}(\phi)$: ϕ is in *prime normal form*, defined by the condition that disjunctions only occur in ϕ immediately under $\Box$.

Then for ϕ in PNF, we shall define:

- $\mathsf{C}(\phi)$: ϕ is *consistent*, i.e. so that we have

$$\mathsf{C}(\phi) \iff \neg(\phi \leq f) \iff [\![\phi]\!] \neq \varnothing$$

 (where $[\![\cdot]\!]$ is the semantics to be introduced below).

- $\mathsf{T}(\phi)$: ϕ requires *termination*, i.e. so that we have

$$\mathsf{T}(\phi) \iff \neg(t \leq \phi) \iff \bot \notin [\![\phi]\!].$$

Of these, the idea of formal consistency, and its definition for function spaces, go back to [Kre59], and also play a major role in [Sco81, Sco82]. The other predicates, as syntactic conditions on expressions, are apparently new (and in the presence of the type constructions we are considering, specifically function space and coalesced sum, the definitions of C and T are *mutually recursive*).

$$\begin{array}{lcl}
\mathsf{C}(t) & \equiv & \mathsf{true} \\
\mathsf{C}(\bigwedge_{i\in I}(\phi_i \times \psi_i)) & \equiv & \mathsf{C}(\bigwedge_{i\in I}\phi_i) \,\&\, \mathsf{C}(\bigwedge_{i\in I}\psi_i) \\
\mathsf{C}(\bigwedge_{i\in I}(\phi_i \rightarrow \psi_i)) & \equiv & \forall J \subseteq I.\, \mathsf{C}(\bigwedge_{j\in J}\phi_j) \;\Rightarrow\; \mathsf{C}(\bigwedge_{j\in J}\psi_j) \\
\mathsf{C}(\bigwedge_{i\in I}(\phi_i \oplus f) & & \\
\;\wedge \bigwedge_{j\in J}(f \oplus \psi_j)) & \equiv & \neg(\mathsf{T}(\bigwedge_{i\in I}\phi_i) \,\&\, \mathsf{T}(\bigwedge_{j\in J}\psi_j)) \\
 & & \&\, \mathsf{C}(\bigwedge_{i\in I}\phi_i) \,\&\, \mathsf{C}(\bigwedge_{j\in J}\psi_j) \\
\mathsf{C}(\bigwedge_{i\in I}(\phi_i)_\bot) & \equiv & \mathsf{C}(\bigwedge_{i\in I}\phi_i) \\
\mathsf{C}(\bigwedge_{i\in I}\Diamond\phi_i) & \equiv & \forall i \in I.\, \mathsf{C}(\phi_i) \\
\mathsf{C}(\bigwedge_{i\in I}\Box\bigvee_{j\in J_i}\phi_{ij}) & \equiv & \exists f \in \prod_{i\in I} J_i.\, \mathsf{C}(\bigwedge_{i\in I}\phi_{if(i)})
\end{array}$$

$$\begin{array}{lcl}
\mathsf{T}(\bigwedge_{i\in I}\phi_i) & \equiv & \exists i \in I.\, \mathsf{T}(\phi) \\
\mathsf{T}(\phi \rightarrow \psi) & \equiv & \mathsf{C}(\phi) \,\&\, \mathsf{T}(\psi) \\
\mathsf{T}(\phi \times \psi) & \equiv & \mathsf{T}(\phi) \text{ or } \mathsf{T}(\psi) \\
\mathsf{T}(\phi \oplus f) & \equiv & \mathsf{T}(f \oplus \phi) \;\equiv\; \mathsf{T}(\phi) \\
\mathsf{T}((\phi)_\bot) & \equiv & \mathsf{true} \\
\mathsf{T}(\Diamond\phi) & \equiv & \mathsf{T}(\Box\phi) \equiv T(\phi).
\end{array}$$

Once we have defined C and T, we can introduce the following derived predicates:

$$\mathsf{CPNF}(\phi) \;\equiv\; \mathsf{PNF}(\phi) \text{ and for all sub-formulae } \psi \text{ of } \phi,$$

$$\begin{aligned}
&\mathsf{PNF}(\psi) \;\Rightarrow\; \mathsf{C}(\psi). \\
\mathsf{CDNF}(\phi) \;&\equiv\; \phi = \bigvee_{i \in I} \phi_i \;\&\; \forall i \in I.\, \mathsf{CPNF}(\phi_i) \\
\#(\phi) \;&\equiv\; \phi = \bigvee_{i \in I} \phi_i \;\&\; \forall i \in I.\, \mathsf{PNF}(\phi) \;\&\; \neg \mathsf{C}(\phi) \\
(\phi){\downarrow} \;&\equiv\; \phi = \bigvee_{i \in I} \phi_i \;\&\; \forall i \in I.\, \mathsf{PNF}(\phi) \;\&\; \mathsf{T}(\phi).
\end{aligned}$$

Now we turn to the axiomatization. The axioms of our logic are all "polymorphic" in character, i.e. they arise from the type constructions uniformly over the types to which the constructions are applied. Thus we omit type subscripts.

The axioms fall into two main groups.

Logical Axioms

These give each $\mathcal{L}(\sigma)$ the structure of a distributive lattice.

$$(\leq - \mathrm{ref}) \quad \phi \leq \phi \qquad (\leq - \mathrm{trans}) \quad \frac{\phi \leq \psi,\ \psi \leq \chi}{\phi \leq \chi}$$

$$(= - I) \quad \frac{\phi \leq \psi,\ \psi \leq \phi}{\phi = \psi} \qquad (= - E) \quad \frac{\phi = \psi}{\phi \leq \psi,\ \psi \leq \phi}$$

$$(t - I) \quad \phi \leq t \qquad (\wedge - I) \quad \frac{\phi \leq \psi_1,\ \phi \leq \psi_2}{\phi \leq \psi_1 \wedge \psi_2}$$

$$(\wedge - E - L) \quad \phi \wedge \psi \leq \phi \qquad (\wedge - E - R) \quad \phi \wedge \psi \leq \psi$$

$$(f - E) \quad f \leq \phi \qquad (\vee - I) \quad \frac{\phi_1 \leq \psi,\ \phi_2 \leq \psi}{\phi_1 \vee \phi_2 \leq \psi}$$

$$(\vee - E - L) \quad \phi \leq \phi \vee \psi \qquad (\vee - E - R) \quad \psi \leq \phi \vee \psi$$

$$(\wedge - \mathrm{dist}) \quad \phi \wedge (\psi \vee \chi) \leq (\phi \wedge \psi) \vee (\psi \wedge \chi)$$

Type-specific Axioms

These articulate each type construction, by showing how its generators interact with the logical structure.

$$(\times - \leq) \quad \frac{\phi \leq \phi', \ \psi \leq \psi'}{(\phi \times \psi) \leq (\phi' \times \psi')}$$

$$(\times - \wedge) \quad \bigwedge_{i \in I} (\phi_i \times \psi_i) = (\bigwedge_{i \in I} \phi_i \times \bigwedge_{i \in I} \psi_i)$$

$$(\times - \vee - L) \quad (\bigvee_{i \in I} \phi_i \times \psi) = \bigvee_{i \in I} (\phi \times \psi)$$

$$(\times - \vee - R) \quad (\phi \times \bigvee_{i \in I} \psi_i) = \bigvee_{i \in I} (\phi \times \psi_i)$$

$$(\rightarrow - \leq) \quad \frac{\phi' \leq \phi, \ \psi \leq \psi'}{(\phi \rightarrow \psi) \leq (\phi' \rightarrow \psi')}$$

$$(\rightarrow - \wedge) \quad (\phi \rightarrow \bigwedge_{i \in I} \psi_i) = \bigwedge_{i \in I} (\phi \rightarrow \psi_i)$$

$$(\rightarrow - \vee - L) \quad (\bigvee_{i \in I} \phi_i \rightarrow \psi) = \bigwedge_{i \in I} (\phi_i \rightarrow \psi)$$

$$(\rightarrow - \vee - R) \quad (\phi \rightarrow \bigvee_{i \in I} \psi_i) = \bigvee_{i \in I} (\phi \rightarrow \psi_i) \quad (\mathsf{CPNF}(\phi))$$

$$(\oplus - \leq) \quad \frac{\phi \leq \psi}{(\phi \oplus f) \leq (\psi \oplus f), \ (f \oplus \phi) \leq (f \oplus \psi)}$$

$$(\oplus - \wedge - L) \quad (\bigwedge_{i \in I} \phi_i \oplus f) = \bigwedge_{i \in I} (\phi_i \oplus f)$$

$$(\oplus - \wedge - R) \quad (f \oplus \bigwedge_{i \in I} \psi_i) = \bigwedge_{i \in I} (f \oplus \psi_i)$$

$$(\oplus - \vee - R) \quad (\bigvee_{i \in I} \phi_i \oplus f) = \bigvee_{i \in I} (\phi_i \oplus f)$$

$$(\oplus - \vee - L) \quad (f \oplus \bigvee_{i \in I} \psi_i) = \bigvee_{i \in I} (f \oplus \psi_i)$$

$$((\cdot)_\perp - \leq) \quad \frac{\phi \leq \psi}{(\phi)_\perp \leq (\psi)_\perp}$$

$$((\cdot)_\perp - \wedge) \quad (\phi \wedge \psi)_\perp = (\phi)_\perp \wedge (\psi)_\perp$$

$$((\cdot)_\perp - \vee) \quad (\bigvee_{i \in I} \phi_i)_\perp = \bigvee_{i \in I} (\phi_i)_\perp$$

$$(\Box - \leq) \quad \frac{\phi \leq \psi}{\Box\phi \leq \Box\psi}$$

$$(\Box - \wedge) \quad \Box \bigwedge_{i \in I} \phi_i = \bigwedge_{i \in I} \Box\phi_i$$

$$(\Box - f) \quad \Box f = f$$

$$(\Diamond - \leq) \quad \frac{\phi \leq \psi}{\Diamond\phi \leq \Diamond\psi}$$

$$(\Diamond - \vee) \quad \Diamond \bigvee_{i \in I} \phi_i = \bigvee_{i \in I} \Diamond\phi_i$$

$$(\Diamond - t) \quad \Diamond t = t$$

$$(\#) \quad \phi \leq f \quad (\#(\phi))$$

The axiom $(\Box - f)$ exemplifies the possibilities for fine-tuning in our approach. It corresponds exactly to the *omission* of the empty set from the upper powerdomain.

To make precise the sense in which this axiomatic presentation is equivalent to the usual denotational construction of domains we define, for each (closed) type expression σ, an interpretation function

$$[\![\cdot]\!]_\sigma : L(\sigma) \longrightarrow K\Omega(\mathcal{D}(\sigma))$$

by

$$
\begin{array}{lcl}
[\![\phi \wedge \psi]\!]_\sigma & = & [\![\phi]\!]_\sigma \cap [\![\psi]\!]_\sigma \\
[\![t]\!]_\sigma & = & D(\sigma) = 1_{K\Omega(\mathcal{D}(\sigma))} \\
[\![\phi \vee \psi]\!]_\sigma & = & [\![\phi]\!]_\sigma \cup [\![\psi]\!]_\sigma \\
[\![f]\!]_\sigma & = & \varnothing = 0_{K\Omega(\mathcal{D}(\sigma))} \\
[\![(\phi \times \psi)]\!]_{\sigma\times\tau} & = & \{<u, v> : u \in [\![\phi]\!]_\sigma,\ v \in [\![\psi]\!]_\tau\} \\
[\![(\phi \rightarrow \psi)]\!]_{\sigma\rightarrow\tau} & = & \{f \in D(\sigma \rightarrow \tau) : f([\![\phi]\!]_\sigma) \subseteq [\![\psi]\!]_\tau\} \\
[\![(\phi \oplus f)]\!]_{\sigma\oplus\tau} & = & \{<0, u> : u \in [\![\phi]\!]_\sigma - \{\bot_\sigma\}\} \\
 & & \cup \{\bot_{\sigma\oplus\tau} : \bot_\sigma \in [\![\phi]\!]_\sigma\} \\
[\![(f \oplus \psi)]\!]_{\sigma\oplus\tau} & = & \{<1, v> : v \in [\![\psi]\!]_\tau - \{\bot_\tau\}\} \\
 & & \cup \{\bot_{\sigma\oplus\tau} : \bot_\tau \in [\![\psi]\!]_\tau\} \\
[\![(\phi)_\bot]\!]_{(\sigma)_\bot} & = & \{<0, u> : u \in [\![\phi]\!]_\sigma\} \\
[\![\Box\phi]\!]_{P_u\sigma} & = & \{S \in D(P_u\sigma) : S \subseteq [\![\phi]\!]_\sigma\} \\
[\![\Diamond\phi]\!]_{P_l\sigma} & = & \{S \in D(P_l\sigma) : S \cap [\![\phi]\!]_\sigma \neq \varnothing\} \\
[\![\phi]\!]_{\mathsf{rec}\, t.\, \sigma} & = & \{\alpha_\sigma(u) : u \in [\![\phi]\!]_{\sigma[\mathsf{rec}\, t.\, \sigma/t]}\}
\end{array}
$$

where $\alpha_\sigma : \mathcal{D}(\sigma[\mathsf{rec}\, t.\, \sigma/t]) \cong \mathcal{D}(\mathsf{rec}\, t.\, \sigma)$ is the isomorphism arising from the initial solution to the domain equation $t = \sigma(t)$.

Then for $\phi, \psi \in L(\sigma)$, we define

$$\mathcal{D}(\sigma) \models \phi \leq \psi \ \equiv \ [\![\phi]\!]_\sigma \subseteq [\![\psi]\!]_\sigma.$$

We now use the results of Chapter 3 to establish some fundamental properties of our system of "Domain Logic".

Firstly, we note that operations on prelocales in the style of Chapter 3 can be distilled from our definitions for product, lifting and Hoare powerdomain. The reader will find no difficulty in carrying out the same programme for these constructions as that shown for function space, Smyth powerdomain and coalesced sum in Chapter 3. Now using 3.5.2, we see that, for each closed σ and any $\rho \in \mathsf{LEnv}$:

$$\mathcal{L}[\![\sigma]\!]\rho = \mathcal{L}(\sigma).$$

The following results are then immediate consequences of our work in Chapter 3.

Notation. $\mathsf{PNF}(\sigma) \equiv \{\phi \in L(\sigma) : \mathsf{PNF}(\phi)\}$, and similarly for $\mathsf{CPNF}(\sigma)$, $\mathsf{CDNF}(\sigma)$.

Proposition 4.2.1 *For all* $\phi \in \mathsf{PNF}(\sigma)$*:*

$$
\begin{array}{rl}
(i) & [\![\phi]\!]_\sigma \in \mathsf{pr}(K\Omega(\mathcal{D}(\sigma))) \\
(ii) & \mathsf{C}(\phi) \iff [\![\phi]\!]_\sigma \neq \varnothing \\
(iii) & \mathsf{T}(\phi) \iff \bot_\sigma \notin [\![\phi]\!].
\end{array}
$$

Lemma 4.2.2 (Normal Forms) *For all* $\phi \in L(\sigma)$*, for some* $\psi \in \mathsf{CDNF}(\sigma)$*:*

$$\mathcal{L}(\sigma) \vdash \phi = \psi.$$

Now we define a relation

$$\leftrightsquigarrow \subseteq \mathsf{CPNF}(\sigma) \times K(\mathcal{D}(\sigma)) :$$

$$\phi \leftrightsquigarrow u \equiv [\![\phi]\!]_\sigma = \uparrow u.$$

Proposition 4.2.3 $\leftrightsquigarrow$ *is a surjective total function.*

Now we come to the main results of the section:

Theorem 4.2.4 (Soundness and Completeness) *For all* $\phi, \psi \in L(\sigma)$*:*

$$\mathcal{L}(\sigma) \vdash \phi \leq \psi \iff \mathcal{D}(\sigma) \models \phi \leq \psi.$$

Now we define

$$\mathcal{LA}(\sigma) \equiv (L(\sigma)/{=_\sigma}, \leq_\sigma /{=_\sigma}),$$

the *Lindenbaum algebra* of $\mathcal{L}(\sigma)$.

Theorem 4.2.5 (Stone Duality) $\mathcal{LA}(\sigma)$ *is the Stone dual of* $\mathcal{D}(\sigma)$*, i.e.*

$$
\begin{array}{rl}
(i) & \mathcal{D}(\sigma) \cong \mathsf{Spec}\,\mathcal{LA}(\sigma) \\
(ii) & K\Omega(\mathcal{D}(\sigma)) \cong \mathcal{LA}(\sigma).
\end{array}
$$

4.3 Programs as Elements: Endogenous Logic

We extend our meta-language for denotational semantics to include typed terms.

Syntax

For each type σ, we have a set of variables

$$\mathsf{Var}(\sigma) = \{x^\sigma, y^\sigma, z^\sigma, \ldots\}.$$

We give the term formation rules *via* an inference system for assertions of the form $M : \sigma$, i.e. "M is a term of type σ".

$$(\mathsf{Var}) \quad x^\sigma : \sigma$$

$$(\mathbf{1} - I) \quad \star : \mathbf{1}$$

$$(\times - I) \ \frac{M : \sigma, \quad N : \tau}{(M, N) : \sigma \times \tau} \qquad (\times - E) \ \frac{M : \sigma \times \tau, \quad N : \upsilon}{\mathsf{let}\ M\ \mathsf{be}\ (x^\sigma, y^\tau).\, N : \upsilon}$$

$$(\rightarrow - I) \ \frac{M : \tau}{\lambda x^\sigma . M : \sigma \rightarrow \tau} \qquad (\rightarrow - E) \ \frac{M : \sigma \rightarrow \tau, \quad N : \sigma}{MN : \tau}$$

$$(\oplus - I - L) \ \frac{M : \sigma}{\imath_{\sigma\tau}(M) : \sigma \oplus \tau} \qquad (\oplus - I - R) \ \frac{N : \tau}{\jmath_{\sigma\tau}(M) : \sigma \oplus \tau}$$

$$(\oplus - E) \ \frac{M : \sigma \oplus \tau, \quad N_1, N_2 : \upsilon}{\mathsf{cases}\ M\ \mathsf{of}\ \imath(x^\sigma).\, N_1\ \mathsf{else}\ \jmath(y^\tau).\, N_2 : \upsilon}$$

$$((\cdot)_\perp - I) \ \frac{M : \sigma}{\mathsf{up}(M) : (\sigma)_\perp} \qquad ((\cdot)_\perp - E) \ \frac{M : (\sigma)_\perp, \quad N : \tau}{\mathsf{lift}\ M\ \mathsf{to}\ \mathsf{up}(x^\sigma).\, N : \tau}$$

$$(\Diamond - I) \ \frac{M : \sigma}{\{\!|M|\!\}_l : P_l\sigma} \qquad (\Box - I) \ \frac{M : \sigma}{\{\!|M|\!\}_u : P_u\sigma}$$

$$(\Diamond - E) \ \frac{M : P_l\sigma, \quad N : P_l\tau}{\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x^\sigma|\!\}_l.\, N : P_l\tau}$$

$$(\Box - E) \ \frac{M : P_u\sigma, \quad N : P_u\tau}{\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x^\sigma|\!\}_u.\, N : P_u\tau}$$

$$(\Diamond - +)\ \frac{M, N : P_l\sigma}{M \uplus_l N : P_l\sigma} \qquad (\Box - +)\ \frac{M, N : P_u\sigma}{M \uplus_u N : P_u\sigma}$$

$$(\Diamond - \otimes)\ \frac{M : P_l\sigma, \quad N : P_l\tau}{M \otimes_l N : P_l(\sigma \times \tau)} \qquad (\Box - \otimes)\ \frac{M : P_u\sigma, \quad N : P_u\tau}{M \otimes_u N : P_u(\sigma \times \tau)}$$

$$(\mathsf{rec} - I)\ \frac{M : \sigma[\mathsf{rec}\, t.\, \sigma/t]}{\mathsf{fold}_{t,\sigma}(M) : \mathsf{rec}\, t.\, \sigma} \qquad (\mathsf{rec} - E)\ \frac{M : \mathsf{rec}\, t.\, \sigma}{\mathsf{unfold}_{t,\sigma}(M) : \sigma[\mathsf{rec}\, t.\, \sigma/t]}$$

$$(\mu - I)\ \frac{M : \sigma}{\mu x^\sigma .\, M : \sigma}$$

We write $\Lambda(\sigma)$ for the set of terms of type σ. Note the systematic presentation of these constructs as *introduction* and *elimination* rules for each of the type constructions, following ideas of Martin-Löf [Mar83] and Plotkin [Plo85]. Note that λ, let, cases, lift, extend, μ are all *variable binding* operations in the obvious way. Also, note that $\{\!|.|\!\}$, extend arise from the adjunction defining the powerdomain construction; $\uplus$ is the operation of the free algebras for this adjunction; while $\otimes$ is the universal map for the tensor product with respect to this operation [HP79].

We now introduce an endogenous program logic with assertions of the form

$$M, \Gamma \vdash \phi$$

where $M : \sigma$, $\phi \in L(\sigma)$, and $\Gamma \in \prod_\sigma \{\mathsf{Var}(\sigma) \to L(\sigma)\}$ gives *assumptions* on the free variables of M.

Notation

$$\Gamma \leq \Delta \equiv \forall x \in \mathsf{Var}.\, \mathcal{L} \vdash \Gamma x \leq \Delta x.$$

For the remainder of this Chapter, we shall omit type subscripts and superscripts "whenever we think we can get away with it", in the delightful formulation of Barr and Wells [BW84, p. 1].

Axiomatisation

$$(\vdash - \wedge)\ \frac{\{M, \Gamma \vdash \phi_i\}_{i \in I}}{M, \Gamma \vdash \bigwedge_{i \in I} \phi_i} \qquad (\vdash - \vee)\ \frac{\{M, \Gamma[x \mapsto \phi_i] \vdash \psi\}_{i \in I}}{M, \Gamma[x \mapsto \bigvee_{i \in I} \phi_i] \vdash \psi}$$

$$(\vdash - \leq)\ \frac{\Gamma \leq \Delta \quad M, \Delta \vdash \phi \quad \phi \leq \psi}{M, \Gamma \vdash \psi} \qquad x, \Gamma[x \mapsto \phi] \vdash \phi$$

$$\frac{M, \Gamma \vdash \phi \quad N, \Gamma \vdash \psi}{(M, N), \Gamma \vdash (\phi \times \psi)} \qquad \frac{M, \Gamma \vdash (\phi \times \psi) \quad N, \Gamma[x \mapsto \phi, y \mapsto \psi] \vdash \theta}{\mathsf{let}\ M\ \mathsf{be}\ (x, y).\, N, \Gamma \vdash \theta}$$

$$\frac{M, \Gamma[x \mapsto \phi] \vdash \psi}{\lambda x. M, \Gamma \vdash (\phi \to \psi)} \qquad \frac{M, \Gamma \vdash (\phi \to \psi) \quad N, \Gamma \vdash \phi}{MN, \Gamma \vdash \psi}$$

$$\frac{M, \Gamma \vdash \phi}{\imath(M), \Gamma \vdash (\phi \oplus f)} \qquad \frac{M : (\phi \oplus f) \quad (\phi\downarrow) \quad N_1, \Gamma[x \mapsto \phi] \vdash \theta}{\mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \vdash \theta}$$

$$\frac{N, \Gamma \vdash \psi}{\jmath(N), \Gamma \vdash (f \oplus \psi)} \qquad \frac{M : (f \oplus \psi) \quad (\psi\downarrow) \quad N_2, \Gamma[y \mapsto \psi] \vdash \theta}{\mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \vdash \theta}$$

$$\frac{M, \Gamma \vdash \phi}{\mathsf{up}(M), \Gamma \vdash (\phi)_\perp} \qquad \frac{M, \Gamma \vdash (\phi)_\perp \quad N, \Gamma[x \mapsto \phi] \vdash \psi}{\mathsf{lift}\ M\ \mathsf{to}\ \mathsf{up}(x).\, N, \Gamma \vdash \psi}$$

$$\frac{M, \Gamma \vdash \phi}{\{\!|M|\!\}_l, \Gamma \vdash \Diamond\phi} \qquad \frac{M, \Gamma \vdash \phi}{\{\!|M|\!\}_u, \Gamma \vdash \Box\phi}$$

$$\frac{M, \Gamma \vdash \Diamond\phi \quad N, \Gamma[x \mapsto \phi] \vdash \Diamond\psi}{\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_l.\, N, \Gamma \vdash \Diamond\psi} \qquad \frac{M, \Gamma \vdash \Box\phi \quad N, \Gamma[x \mapsto \phi] \vdash \Box\psi}{\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_u.\, N, \Gamma \vdash \Box\psi}$$

$$\frac{M, \Gamma \vdash \Diamond\phi}{M \uplus_l N, \Gamma \vdash \Diamond\phi} \qquad \frac{N, \Gamma \vdash \Diamond\psi}{M \uplus_l N, \Gamma \vdash \Diamond\psi} \qquad \frac{M, \Gamma \vdash \Box\phi \quad N, \Gamma \vdash \Box\phi}{M \uplus_u N, \Gamma \vdash \Box\phi}$$

$$\frac{M, \Gamma \vdash \Diamond\phi \quad N, \Gamma \vdash \Diamond\psi}{M \otimes_l N, \Gamma \vdash \Diamond(\phi \times \psi)} \qquad \frac{M, \Gamma \vdash \Box\phi \quad N, \Gamma \vdash \Box\psi}{M \otimes_u N, \Gamma \vdash \Box(\phi \times \psi)}$$

$$\frac{M, \Gamma \vdash \phi}{\mathsf{fold}(M), \Gamma \vdash \phi} \qquad \frac{M, \Gamma \vdash \phi}{\mathsf{unfold}(M), \Gamma \vdash \phi}$$

$$\frac{\mu x.\, M, \Gamma \vdash \phi \quad M, \Gamma[x \mapsto \phi] \vdash \psi}{\mu x.\, M, \Gamma \vdash \psi}$$

Note that there is one inference rule for $\vdash$ per formation rule in our syntax. Thus we can refer *e.g.* to rule $(\vdash - \times - E)$ without ambiguity. Note the role of the convergence predicate $(\cdot)\downarrow$ in $(\vdash - \oplus - E)$; it plays a similar role in the elimination rules for the other "strict" constructions of smash product [Plo81, Chapter 3 p. 1] and strict function space [Plo81, Chapter 1 p. 11], which we do not cover here.

Semantics

Following standard ideas [Plo81, SP82, Plo76], we now give a denotational semantics for this meta-language, in the form of a map

$$[\![\cdot]\!]_\sigma : \Lambda(\sigma) \longrightarrow \mathsf{Env} \longrightarrow \mathcal{D}(\sigma)$$

where $\mathsf{Env} \equiv \prod_\sigma \{\mathsf{Var}(\sigma) \to \mathcal{D}(\sigma)\}$ is the set of *environments*.

$$
\begin{array}{lcl}
[\![x]\!]\rho & = & \rho x \\
[\![(M,N)]\!]\rho & = & <[\![M]\!]\rho, [\![N]\!]\rho> \\
[\![\mathsf{let}\ M\ \mathsf{be}\ (x,y).\,N]\!]\rho & = & [\![N]\!]\rho[x \mapsto d, y \mapsto e] \\
 & & \text{where} \\
 & & <d,e> = [\![M]\!]\rho \\
[\![\imath(M)]\!]\rho & = & \begin{cases} <0,[\![M]\!]\rho>, & [\![M]\!]\rho \neq \bot \\ \bot & [\![M]\!]\rho = \bot \end{cases} \\
[\![\jmath(N)]\!]\rho & = & \begin{cases} <1,[\![N]\!]\rho>, & [\![N]\!]\rho \neq \bot \\ \bot & [\![N]\!]\rho = \bot \end{cases} \\
[\![\mathsf{cases}\ M\ \mathsf{of} & & \\
\imath(x).\,N_1\ \mathsf{else}\ \jmath(y).\,N_2]\!]\rho & = & \begin{cases} [\![N_1]\!]\rho[x \mapsto d], & [\![M]\!]\rho = <0,d> \\ [\![N_2]\!]\rho[x \mapsto e], & [\![M]\!]\rho = <1,e> \\ \bot, & [\![M]\!]\rho = \bot \end{cases} \\
[\![\mathsf{up}(M)]\!]\rho & = & <0,[\![M]\!]\rho> \\
[\![\mathsf{lift}\ M\ \mathsf{to}\ \mathsf{up}(x).\,N]\!]\rho & = & \begin{cases} [\![N]\!]\rho[x \mapsto d], & [\![M]\!]\rho = <0,d> \\ \bot, & [\![M]\!]\rho = \bot \end{cases} \\
[\![\{\!|M|\!\}_l]\!]\rho & = & \downarrow([\![M]\!]\rho) \\
[\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_l.\,N]\!]\rho & = & \bigcup\{[\![N]\!]\rho[x \mapsto d] : d \in [\![M]\!]\rho\} \\
[\![M \uplus_l N]\!]\rho & = & ([\![M]\!]\rho) \cup ([\![N]\!]\rho) \\
[\![M \otimes_l N]\!]\rho & = & ([\![M]\!]\rho) \times ([\![N]\!]\rho)
\end{array}
$$

$$
\begin{array}{lcl}
[\![\{\!|M|\!\}_u]\!]\rho & = & \uparrow([\![M]\!]\rho) \\
[\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_u.\, N]\!]\rho & = & \bigcup\{[\![N]\!]\rho[x \mapsto d] : d \in [\![M]\!]\rho\} \\
[\![M \uplus_u N]\!]\rho & = & ([\![M]\!]\rho) \cup ([\![N]\!]\rho) \\
[\![M \otimes_u N]\!]\rho & = & ([\![M]\!]\rho) \times ([\![N]\!]\rho) \\
[\![\mathsf{fold}(M)]\!]\rho & = & \alpha([\![M]\!]\rho) \\
[\![\mathsf{unfold}(M)]\!]\rho & = & \alpha^{-1}([\![M]\!]\rho) \\
[\![\mu x.\, M]\!]\rho & = & \bigsqcup_{k\in\omega} d_k \\
 & & \text{where} \\
 & & d_0 = \bot, \quad d_{k+1} = [\![M]\!]\rho[x \mapsto d_k]
\end{array}
$$

Here α is the initial algebra isomorphism as in Section 2 page 78. We can use this semantics to define a notion of validity for assertions:

$$M, \Gamma \models \phi \ \equiv\ \forall \rho \in \mathsf{Env}.\, \rho \models \Gamma \Rightarrow [\![M]\!]_\sigma \rho \models \phi$$

where

$$\rho \models \Gamma \ \equiv\ \forall x \in \mathsf{Var}.\, \rho x \models \Gamma x$$

and for $d \in D(\sigma)$, $\phi \in L(\sigma)$:

$$d \models \phi \ \equiv\ d \in [\![\phi]\!]_\sigma.$$

We can now state the main result of this section:

Theorem 4.3.1 *The Endogenous logic is sound and complete:*

$$\forall M, \Gamma, \phi.\, M, \Gamma \vdash \phi \iff M, \Gamma \models \phi.$$

We can state this result more sharply in terms of Stone Duality: it says that

$$\eta_\sigma^{-1}(\{[\phi]_{=_\sigma} : M, \Gamma \vdash \phi\}) = [\![M]\!]_\sigma \rho,$$

where

$$\eta_\sigma : \mathcal{D}(\sigma) \ \cong\ \mathsf{Spec}\, \mathcal{LA}(\sigma)$$

is the component of the natural isomorphism arising from Theorem 4.2.5; i.e. that we recover the point of $\mathcal{D}(\sigma)$ given by the denotational semantics of M from the properties we can prove to hold of M in our logic.

We now turn to the proof of Theorem 4.3.1. Our strategy is analogous to that of Chapter 3; we get Completeness *via* Prime Completeness. Firstly, we have:

Theorem 4.3.2 (Soundness) *For all M, Γ, ϕ:*

$$M, \Gamma \vdash \phi \implies M, \Gamma \models \phi.$$

Proof. By a routine induction on the length of proofs in the endogenous logic. We give two cases for illustration.
1. Suppose the last step in the proof is an application of $(\vdash - \rightarrow - I)$:

$$\frac{M, \Gamma[x \mapsto \phi] \vdash \psi}{\lambda x.M, \Gamma \vdash (\phi \rightarrow \psi)}$$

By induction hypothesis, $M, \Gamma[x \mapsto \phi] \models \psi$, i.e for all $\rho \models \Gamma$, $d \in \mathcal{D}(\sigma)$,

$$d \in [\![\phi]\!] \implies [\![M]\!]\rho[x \mapsto d] \in [\![\psi]\!],$$

which implies

$$\lambda x.M, \Gamma \models (\phi \rightarrow \psi).$$

2. Next we consider $(\vdash - \Box - E)$:

$$\frac{M, \Gamma \vdash \Box\phi \quad N, \Gamma[x \mapsto \phi] \vdash \Box\psi}{\textsf{over } M \textsf{ extend } \{\!|x|\!\}_u.\, N, \Gamma \vdash \Box\psi}$$

By induction hypothesis, $M, \Gamma \models \Box\phi$ and $N, \Gamma[x \mapsto \phi] \models \Box\psi$. Hence for $\rho \models \Gamma$, $[\![M]\!]\rho \subseteq [\![\phi]\!]$, and for $d \in \mathcal{D}(\sigma)$,

$$d \in [\![\phi]\!] \implies [\![N]\!]\rho[x \mapsto d] \subseteq [\![\psi]\!].$$

Thus

$$\begin{aligned} & \textstyle\bigcup_{d \in [\![M]\!]\rho} [\![N]\!]\rho[x \mapsto d] \subseteq [\![\psi]\!] \\ \implies & [\![\textsf{over } M \textsf{ extend } \{\!|x|\!\}_u.\, N]\!]\rho \subseteq [\![\psi]\!] \\ \implies & \textsf{over } M \textsf{ extend } \{\!|x|\!\}_u.\, N, \Gamma \models \Box\psi. \quad \blacksquare \end{aligned}$$

Next, we shall need a technical lemma which describes our program constructs under the denotational semantics.

Lemma 4.3.3 *For* $u \in \mathcal{K}(\mathcal{D}(\sigma))$, $v \in \mathcal{K}(\mathcal{D}(\tau))$, $w \in \mathcal{K}(\mathcal{D}(\upsilon))$, $X \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\sigma)))$, $Y \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\tau)))$, $Z \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\sigma \times \tau)))$, $w_1 \in \mathcal{K}(\mathcal{D}(\mathsf{rec}\ t.\,\sigma))$, $w_2 \in \mathcal{K}(\mathcal{D}(\sigma[\mathsf{rec}\ t.\,\sigma/t]))$*:*

$$
\begin{array}{rl}
(i) & (u,v) \sqsubseteq [\![(M,N)]\!]\rho \;\Leftrightarrow\; u \sqsubseteq [\![M]\!]\rho \,\&\, v \sqsubseteq [\![N]\!]\rho \\
(ii) & w \sqsubseteq [\![\mathsf{let}\ M\ \mathsf{be}\ (x,y).\,N]\!]\rho \;\Leftrightarrow\; \exists u,v. \\
 & (u,v) \sqsubseteq [\![M]\!]\rho \,\&\, w \sqsubseteq [\![N]\!]\rho[x \mapsto u, y \mapsto v] \\
(iii) & [u,v] \sqsubseteq [\![\lambda x.M]\!]\rho \;\Leftrightarrow\; v \sqsubseteq [\![M]\!]\rho[x \mapsto u] \\
(iv) & v \sqsubseteq [\![MN]\!]\rho \;\Leftrightarrow\; \exists u.[u,v] \sqsubseteq [\![M]\!]\rho \,\&\, u \sqsubseteq [\![N]\!]\rho \\
(v) & <0,u> \sqsubseteq [\![\imath(M)]\!]\rho \;\Leftrightarrow\; u \sqsubseteq [\![M]\!]\rho \\
 & <1,v> \sqsubseteq [\![\jmath(N)]\!]\rho \;\Leftrightarrow\; v \sqsubseteq [\![N]\!]\rho \\
(vi) & w \neq \bot \;\Longrightarrow\; w \sqsubseteq [\![\mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\,N_1\ \mathsf{else}\ \jmath(y).\,N_2]\!]\rho \;\Leftrightarrow \\
 & \exists u \neq \bot.\, <0,u> \sqsubseteq [\![M]\!]\rho \,\&\, w \sqsubseteq [\![N_1]\!]\rho[x \mapsto u] \\
 & or \\
 & \exists v \neq \bot.\, <1,v> \sqsubseteq [\![M]\!]\rho \,\&\, w \sqsubseteq [\![N_2]\!]\rho[x \mapsto v] \\
(vii) & <0,u> \sqsubseteq [\![\mathsf{up}(M)]\!]\rho \;\Leftrightarrow\; u \sqsubseteq [\![M]\!]\rho \\
(viii) & v \neq \bot \;\Longrightarrow\; v \sqsubseteq [\![\mathsf{lift}\ M\ \mathsf{to}\ \mathsf{up}(x).\,N]\!]\rho \;\Leftrightarrow \\
 & \exists u.\, <0,u> \sqsubseteq [\![M]\!]\rho \,\&\, v \sqsubseteq [\![N]\!]\rho[x \mapsto u] \\
(ix) & \downarrow X \sqsubseteq [\![\{\!|M|\!\}_l]\!]\rho \;\Leftrightarrow\; \forall x \in X.\, x \sqsubseteq [\![M]\!]\rho \\
(x) & \downarrow Y \sqsubseteq [\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_l.\,N]\!]\rho \;\Leftrightarrow\; \exists X.\downarrow X \sqsubseteq [\![M]\!]\rho \\
 & \&\downarrow Y \sqsubseteq \bigcup_{u \in X} [\![N]\!]\rho[x \mapsto u] \\
(xi) & \downarrow X \sqsubseteq [\![M \uplus_l N]\!]\rho \;\Leftrightarrow\; \downarrow X \sqsubseteq [\![M]\!]\rho \ or \downarrow X \sqsubseteq [\![N]\!]\rho \\
(xii) & \downarrow Z \sqsubseteq [\![M \otimes_l N]\!]\rho \;\Leftrightarrow\; \exists X,Y.\downarrow Z \sqsubseteq \downarrow X \otimes_l \downarrow Y \\
 & \&\downarrow X \sqsubseteq [\![M]\!]\rho \,\&\downarrow Y \sqsubseteq [\![N]\!]\rho \\
(xiii) & \uparrow X \sqsubseteq [\![\{\!|M|\!\}_u]\!]\rho \;\Leftrightarrow\; \exists x \in X.\, x \sqsubseteq [\![M]\!]\rho
\end{array}
$$

$$(xiv)\quad \uparrow Y \sqsubseteq [\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_u.\, N]\!]\rho \;\Leftrightarrow\; \exists X.\, \uparrow X \sqsubseteq [\![M]\!]\rho$$
$$\& \uparrow Y \sqsubseteq \textstyle\bigcup_{u \in X} [\![N]\!]\rho[x \mapsto u]$$
$$(xv)\quad \uparrow X \sqsubseteq [\![M \uplus_u N]\!]\rho \;\Leftrightarrow\; \uparrow X \sqsubseteq [\![M]\!]\rho \;\&\; \uparrow X \sqsubseteq [\![N]\!]\rho$$
$$(xvi)\quad \uparrow Z \sqsubseteq [\![M \otimes_u N]\!]\rho \;\Leftrightarrow\; \exists X, Y.\, \uparrow Z \sqsubseteq \uparrow X \otimes_u \uparrow Y$$
$$\& \uparrow X \sqsubseteq [\![M]\!]\rho \;\&\; \uparrow Y \sqsubseteq [\![N]\!]\rho$$
$$(xvii)\quad w_1 \sqsubseteq [\![\mathsf{fold}(M)]\!]\rho \;\Leftrightarrow\; \alpha^{-1}(w_1) \sqsubseteq [\![M]\!]\rho$$
$$(xviii)\quad w_2 \sqsubseteq [\![\mathsf{unfold}(M)]\!]\rho \;\Leftrightarrow\; \alpha(w_2) \sqsubseteq [\![M]\!]\rho$$
$$(xix)\quad u \sqsubseteq [\![\mu x.M]\!]\rho \;\Leftrightarrow\; \exists k \in \omega,\, u_0, \ldots, u_k.\, u_0 = \bot \;\&\; u_k = u$$
$$\& \forall i : 0 \leq i < k.\, u_{i+1} \sqsubseteq [\![M]\!]\rho[x \mapsto u_i]$$

PROOF. The content of this Lemma is all quite standard, at least in the folklore. It amounts to a description of the combinators underlying the denotational semantics of terms as *approximable mappings*. Most of it can be found, couched in the language of information systems, in [Sco82], and for neighbourhood systems in [Sco81]. We shall just give a couple of the less familiar cases for illustration.
(xii).

- $\downarrow Z \sqsubseteq [\![M \otimes_l N]\!]\rho$

$\Leftrightarrow$ $\downarrow Z \subseteq \bigsqcup\{\downarrow X \otimes_l \downarrow Y : \downarrow X \sqsubseteq [\![M]\!]\rho \;\&\; \downarrow Y \sqsubseteq [\![N]\!]\rho\}$

since $\otimes_l$ is continuous

$\Leftrightarrow$ $\exists X, Y.\, \downarrow Z \sqsubseteq \downarrow X \otimes_l \downarrow Y \;\&\; \downarrow X \sqsubseteq [\![M]\!]\rho \;\&\; \downarrow Y \sqsubseteq [\![N]\!]\rho$

since $\downarrow Z$ is finite.

(xiv).

- $\uparrow Y \sqsubseteq [\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_u.\, N]\!]\rho$

$\Leftrightarrow$ $\uparrow Y \sqsubseteq \bigsqcup_{\uparrow X \sqsubseteq [\![M]\!]\rho} \bigcup\{[\![N]\!]\rho[x \mapsto u] : u \in \uparrow X\}$

since extend is continuous

$\Leftrightarrow$ $\exists X.\, \uparrow X \sqsubseteq [\![M]\!]\rho \;\&\; \uparrow Y \sqsubseteq \bigcup_{u \in \uparrow X} [\![N]\!]\rho[x \mapsto u]$

since $\uparrow Y$ is finite. The argument is completed by observing that

$$\bigcup_{u \in \uparrow X} [\![N]\!]\rho[x \mapsto u] = \bigcup_{u \in X} [\![N]\!]\rho[x \mapsto u]. \quad \blacksquare$$

Now for Prime Completeness.

Notation. $\mathsf{CPNF}(\Gamma) \equiv \forall x \in \mathsf{Var}.\, \mathsf{CPNF}(\Gamma x)$.

Theorem 4.3.4 (Prime Completeness) $\mathsf{CPNF}(\Gamma)$ *and* $\mathsf{CPNF}(\phi)$ *imply that*

$$M, \Gamma \models \phi \implies M, \Gamma \vdash \phi$$

PROOF. We begin by establishing some useful notation. Given Γ with $\mathsf{CPNF}(\Gamma)$, we define an environment ρ_Γ by:

$$\forall x \in \mathsf{Var}.\, \Gamma x \leftrightsquigarrow \rho_\Gamma x.$$

This is well-defined by Proposition 4.2.3. Similarly, let $\phi \leftrightsquigarrow u$. Now we have:

$$M, \Gamma \models \phi \iff u \sqsubseteq [\![M]\!]\rho_\Gamma. \tag{4.1}$$

The proof proceeds by induction on M. As the various cases all share a common pattern, we shall only give a selection of the more interesting for illustration.

Abstraction. We argue by induction on ϕ. The inductive case, which can only be a conjunction, since ϕ is in CPNF, is trivial. We are left with the case for a generator $(\phi \rightarrow \psi)$, where ϕ, ψ are in CPNF. Let $\phi \leftrightsquigarrow u$, $\psi \leftrightsquigarrow v$. Then

$\bullet$	$\lambda x.M, \Gamma \models (\phi \rightarrow \psi)$	
$\Rightarrow$	$[u, v] \sqsubseteq [\![\lambda x.M]\!]\rho_\Gamma$	4.1
$\Rightarrow$	$v \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto u]$	4.3.3(iii)
$\Rightarrow$	$M, \Gamma[x \mapsto \phi] \models \psi$	4.1
$\Rightarrow$	$M, \Gamma[x \mapsto \phi] \vdash \psi$	ind. hyp.
$\Rightarrow$	$\lambda x.M, \Gamma \vdash (\phi \rightarrow \psi)$	$(\vdash - \rightarrow - I)$

Application.

$$\begin{array}{lll}
\bullet & MN, \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![MN]\!]\rho_\Gamma & 4.1 \\
\Rightarrow & \exists v.\, [v, u] \sqsubseteq [\![M]\!]\rho \,\&\, v \sqsubseteq [\![N]\!]\rho & 4.3.3(\text{iv}) \\
\Rightarrow & M, \Gamma \models (\psi \rightarrow \phi) \;\&\; N, \Gamma \models \psi & 4.1 \\
 & \text{where } \psi \leftrightsquigarrow v & \\
\Rightarrow & M, \Gamma \vdash (\psi \rightarrow \phi) \;\&\; N, \Gamma \vdash \psi & \text{ind. hyp.} \\
\Rightarrow & MN, \Gamma \vdash \phi & (\vdash - \rightarrow - E).
\end{array}$$

Case expression.

$$\begin{array}{lll}
 & \mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \models \phi & \\
\Leftrightarrow & u \sqsubseteq [\![\mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2]\!]\rho_\Gamma & 4.1.
\end{array}$$

If $u = \bot$, then $\mathcal{L} \vdash t \leq \phi$, and the required conclusion follows by $(\vdash - \wedge)$ and $(\vdash - \leq)$. Otherwise, by 4.3.3(vi), either

$$(i)\ \ \exists u_1 \neq \bot.\, <0, u_1> \sqsubseteq [\![M]\!]\rho_\Gamma \,\&\, u \sqsubseteq [\![N_1]\!]\rho_\Gamma[x \mapsto u_1]$$

or

$$(ii)\ \ \exists u_2 \neq \bot.\, <1, u_2> \sqsubseteq [\![M]\!]\rho_\Gamma \,\&\, u \sqsubseteq [\![N_2]\!]\rho_\Gamma[x \mapsto u_2].$$

We shall consider sub-case (i); (ii) is entirely similar. Let $\phi_1 \leftrightsquigarrow u_1$. Then

$$\begin{array}{lll}
\bullet & <0, u_1> \sqsubseteq [\![M]\!]\rho_\Gamma \,\&\, u \sqsubseteq [\![N_1]\!]\rho_\Gamma[x \mapsto u_1] & \\
\Rightarrow & M, \Gamma \models (\phi_1 \oplus f) \;\&\; N_1, \Gamma[x \mapsto \phi_1] \models \phi & 4.1 \\
\Rightarrow & M, \Gamma \vdash (\phi_1 \oplus f) \;\&\; N_1, \Gamma[x \mapsto \phi_1] \vdash \phi & \text{ind. hyp.} \\
\Rightarrow & \mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \vdash \phi & \text{by } (\vdash - \oplus - E) \\
 & \text{since } u_1 \neq \bot \text{ implies } \phi_1\!\downarrow \text{ by 4.2.1.} &
\end{array}$$

Tensor product. We write $\phi \in \mathsf{CPNF}(P_u(\sigma \times \tau))$ as $\Box \bigvee_{i \in I}(\phi \times \psi)$, and define $Z = \uparrow\{(u_i, v_i) : i \in I\}$, where

$$\phi_i \leftrightsquigarrow u_i, \quad \psi_i \leftrightsquigarrow v_i \quad (i \in I).$$

Now

- $M \otimes_u N, \Gamma \models \Box \bigvee_{i \in I}(\phi \times \psi)$

$\Rightarrow$ $Z \sqsubseteq [\![M \otimes_u N]\!]\rho_\Gamma$ 4.1

$\Rightarrow$ $\exists X, Y.\, \uparrow X \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; \uparrow Y \sqsubseteq [\![N]\!]\rho_\Gamma$

$\& \uparrow Z \sqsubseteq \uparrow X \otimes_u \uparrow Y = \uparrow(X \times Y)$ 4.3.3(xvi)

Let $X = \{u_k\}_{k \in K}$, $Y = \{v_l\}_{l \in L}$, and define

$\phi_k \leftrightsquigarrow u_k \;\; (k \in K), \quad \psi_l \leftrightsquigarrow v_l \;\; (l \in L).$

Now

- $\uparrow X \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; \uparrow Y \sqsubseteq [\![N]\!]\rho_\Gamma$

$\Rightarrow$ $M, \Gamma \models \Box \bigvee_{k \in K} \phi_k \;\;\&\;\; N, \Gamma \models \Box \bigvee_{l \in L} \psi_l$ 4.1

$\Rightarrow$ $M, \Gamma \vdash \Box \bigvee_{k \in K} \phi_k \;\;\&\;\; N, \Gamma \vdash \Box \bigvee_{l \in L} \psi_l$ ind. hyp.

$\Rightarrow$ $M \otimes_u N, \Gamma \vdash \Box(\bigvee_{k \in K} \phi_k \times \bigvee_{l \in L} \psi_l)$ $(\vdash - \Box - \otimes)$.

Finally,

$$\begin{aligned} \mathcal{L} \vdash (\textstyle\bigvee_{k \in K} \phi_k \times \bigvee_{l \in L} \psi_l) &= \textstyle\bigvee_{(k,l) \in K \times L}(\phi_k \times \psi_l) \quad (\times - \vee) \\ &\leq \textstyle\bigvee_{i \in I}(\phi_i \times \psi_i) \end{aligned}$$

since $Z \sqsubseteq \uparrow X \otimes_u \uparrow Y$ implies

$\forall k, l.\, \exists i.\, \mathcal{L} \vdash (\phi_k \times \psi_l) \leq (\phi_i \times \psi_i).$

Hence by $(\vdash - \leq)$,

$$M \otimes_u N, \Gamma \vdash \Box \bigvee_{i \in I}(\phi_i \times \psi_i).$$

Extension. As in the case for abstraction, it suffices to consider the case when ϕ is a generator $\Box \bigvee_{i \in I} \phi_i$. We define $Y = \{u_i\}_{i \in I}$, where $\phi_i \leftrightsquigarrow u_i$, $(i \in I)$. Now

- $\textsf{over } M \textsf{ extend } \{\!|x|\!\}_u.\, N, \Gamma \models \Box \bigvee_{i \in I} \phi_i$

$\Rightarrow$ $\uparrow Y \sqsubseteq [\![\textsf{over } M \textsf{ extend } \{\!|x|\!\}_u.\, N]\!]\rho_\Gamma$ 4.1

$\Rightarrow$ $\exists X.\, \uparrow X \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; \uparrow Y \sqsubseteq \bigcup_{u \in X} [\![N]\!]\rho_\Gamma[x \mapsto u]$ 4.3.3(xiv)

$\Rightarrow$ $\exists X.\, \uparrow X \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; \forall u \in X.\, \uparrow Y \sqsubseteq [\![N]\!]\rho_\Gamma[x \mapsto u]$

Let $X = \{v_j\}_{j \in J}$, $\psi_j \leftrightsquigarrow v_j$, $(j \in J)$. Then

$$
\begin{array}{lll}
\bullet & \uparrow X \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; \forall u \in X.\, \uparrow Y \sqsubseteq [\![N]\!]\rho_\Gamma[x \mapsto u] & \\
\Rightarrow & M, \Gamma \models \Box \bigvee_{j \in J} \psi_j \;\&\; \forall j \in J.\, N, \Gamma[x \mapsto \psi_j] \models \phi & 4.1 \\
\Rightarrow & M, \Gamma \vdash \Box \bigvee_{j \in J} \psi_j \;\&\; \forall j \in J.\, N, \Gamma[x \mapsto \psi_j] \vdash \phi & \text{ind. hyp.} \\
\Rightarrow & M, \Gamma \vdash \Box \bigvee_{j \in J} \psi_j \;\&\; N, \Gamma[x \mapsto \bigvee_{j \in J} \psi_j] \vdash \phi & (\vdash - \vee) \\
\Rightarrow & \mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}_u.\, N, \Gamma \vdash \phi & (\vdash - \Box - E)
\end{array}
$$

Recursive types. Firstly, we note that for $\phi \in \mathcal{L}(\mathsf{rec}\, t.\, \sigma)$,

$$\phi \leftrightsquigarrow u \quad \Leftrightarrow \quad \phi \leftrightsquigarrow \alpha^{-1}(u),$$

since $\mathcal{L}(\mathsf{rec}\, t.\, \sigma) = \mathcal{L}(\sigma[\mathsf{rec}\, t.\, \sigma/t])$. Now,

$$
\begin{array}{lll}
\bullet & \mathsf{fold}(M), \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![\mathsf{fold}(M)]\!]\rho_\Gamma & 4.1 \\
\Rightarrow & \alpha^{-1}(u) \sqsubseteq [\![M]\!]\rho_\Gamma & 4.3.3(\text{xvii}) \\
\Rightarrow & M, \Gamma \models \phi & 4.1 \\
\Rightarrow & M, \Gamma \vdash \phi & \text{ind. hyp.} \\
\Rightarrow & \mathsf{fold}(M), \Gamma \vdash \phi & (\vdash - \mathsf{rec} - I)
\end{array}
$$

Recursion.

$$
\begin{array}{lll}
\bullet & \mu x.M, \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![\mu x.M]\!]\rho_\Gamma & 4.1 \\
\Rightarrow & \exists k \in \omega, u_0, \ldots, u_k.\, u_0 = \bot \;\&\; u_k = u & \\
 & \;\&\; \forall i : 0 \leq i < k.\, u_{i+1} \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto u_i] & 4.3.3(\text{xix}).
\end{array}
$$

Let $\|u\|$ be the least such k (as a function of u for $u \sqsubseteq [\![\mu x.M]\!]\rho_\Gamma$, keeping $\mu x.M$, Γ fixed). We complete the proof for this case by induction on $\|u\|$, with $\phi \leftrightsquigarrow u$.

Basis:

$$\|u\| = 0 \Rightarrow u = \bot \Rightarrow \vdash t \leq \phi \Rightarrow \mu x.M, \Gamma \vdash \phi,$$

by $(\vdash - \wedge)$ and $(\vdash - \leq)$.

Induction step: $\|u\| = k+1$. Then by definition of $\|u\|$, for some v:

$$u \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto v] \ \&\ \|v\| = k.$$

Let $\psi \leftrightsquigarrow v$. Then

$$\begin{array}{lll} \bullet & u \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto v] \ \&\ \|v\| = k & \\ \Rightarrow & M, \Gamma[x \mapsto \psi] \models \phi & 4.1 \\ & \text{and } \mu x.M, \Gamma \vdash \psi & \text{inner ind. hyp.} \\ \Rightarrow & M, \Gamma[x \mapsto \psi] \vdash \phi \ \&\ \mu x.M, \Gamma \vdash \psi & \text{outer ind. hyp.} \\ \Rightarrow & \mu x.M, \Gamma \vdash \phi & (\vdash - \mu - I). \ \blacksquare \end{array}$$

Finally, we can prove Theorem 4.3.1. One half is Theorem 4.3.2. For the converse, suppose $M, \Gamma \models \phi$. We can assume that $\Gamma x \neq f$[1] for all $x \in \mathsf{Var}$, since otherwise we could apply $(\vdash - \vee)$ to obtain $M, \Gamma \vdash \phi$. Let $V = \mathsf{FV}(M)$, the *free variables* of M. (We omit the formal definition, which should be obvious). We define Γ_V by

$$\Gamma_V x = \begin{cases} \Gamma x, & x \in V \\ t & \text{otherwise.} \end{cases}$$

Then by standard arguments we have:

$$M, \Gamma \models \phi \ \Leftrightarrow\ M, \Gamma_V \models \phi \qquad (4.2)$$
$$M, \Gamma \vdash \phi \ \Leftrightarrow\ M, \Gamma_V \vdash \phi \qquad (4.3)$$

Now by Lemma 4.2.2, we have

$$\mathcal{L} \vdash \phi = \bigvee_{i \in I} \phi_i,$$

and for all $x \in V$,

$$\mathcal{L} \vdash \Gamma x = \bigvee_{j \in J_x} \psi_j,$$

[1] meaning $[\![\Gamma x]\!] \neq \varnothing$, or, equivalently by Theorem 4.2.5, $\mathcal{L} \nvdash \Gamma x = f$

with each ϕ_i, ψ_j in CPNF. Moreover, our assumption that $\Gamma x \neq f$ for all x implies that $J_x \neq \varnothing$ for all $x \in V$. Given $f \in \prod_{x \in V} J_x$ (i.e. a *choice function* selecting one of the disjuncts ψ_{fx}, $fx \in J_x$, for each $x \in V$), we define Γ_f by:

$$\Gamma_f\, x = \begin{cases} \psi_{fx}, & x \in V \\ t & \text{otherwise.} \end{cases}$$

Then

$$\begin{array}{lll}
\bullet & M, \Gamma \models \phi & \\
\Rightarrow & M, \Gamma_V \models \phi & 4.2 \\
\Rightarrow & \forall f \in \prod_{x \in V} J_x.\, M, \Gamma_f \models \bigvee_{i \in I} \phi_i & (\vdash - \leq), \text{Soundness} \\
\Rightarrow & \forall f \in \prod_{x \in V} J_x.\, \exists i \in I.\, M, \Gamma_f \models \phi_i & \\
\Rightarrow & \forall f \in \prod_{x \in V} J_x.\, \exists i \in I.\, M, \Gamma_f \vdash \phi_i & \text{Prime Completeness} \\
\Rightarrow & \forall f \in \prod_{x \in V} J_x.\, M, \Gamma_f \vdash \phi & (\vdash - \leq) \\
\Rightarrow & M, \Gamma_V \vdash \phi & (\vdash - \vee) \\
\Rightarrow & M, \Gamma \vdash \phi & 4.3 \quad \blacksquare
\end{array}$$

4.4 Programs as Morphisms: Exogenous Logic

We now introduce a second extension of our denotational meta-language, which provides a syntax of terms denoting *morphisms between*, rather than elements of, domains. This is an extended version of the algebraic meta-language for cartesian closed categories [Poi86, LS86], just as the language of the previous section was an extended typed λ-calculus. Terms are sorted on *morphism types* (σ, τ), with notation $f : (\sigma, \tau)$. We shall give the formation rules in "polymorphic" style, with type subscripts omitted.

Syntax of morphism terms

- $\mathsf{id} : (\sigma, \sigma)$
- $\dfrac{f : (\sigma, \tau) \quad g : (\tau, \upsilon)}{f; g : (\sigma, \upsilon)}$
- $1 : (\sigma, \mathbf{1})$
- $\dfrac{f : (\upsilon, \sigma) \quad g : (\upsilon, \tau)}{<f, g> : (\upsilon, \sigma \times \tau)}$
- $\mathsf{p} : (\sigma \times \tau, \sigma)$
- $\mathsf{q} : (\sigma \times \tau, \tau)$
- $\dfrac{f : (\sigma \times \tau, \upsilon)}{\Lambda(f) : (\sigma, \tau \to \upsilon)}$
- $\mathsf{Ap} : ((\sigma \to \tau) \times \sigma, \tau)$
- $\mathsf{l} : (\sigma, \sigma \oplus \tau)$
- $\mathsf{r} : (\tau, \sigma \oplus \tau)$
- $\dfrac{f : (\sigma, \upsilon) \quad g : \tau, \upsilon)}{[f, g] : (\sigma \oplus \tau, \upsilon)}$
- $\mathsf{up} : (\sigma, (\sigma)_\bot)$
- $\dfrac{f : (\sigma, \tau)}{\mathsf{lift}(f) : ((\sigma)_\bot, \tau)}$
- $\dfrac{f : (\sigma, \tau)}{\mathsf{strict}(f) : (\sigma, \tau)}$
- $\{\!|\cdot|\!\}_l : (\sigma, P_l\sigma)$
- $\{\!|\cdot|\!\}_u : (\sigma, P_u\sigma)$
- $\dfrac{f : (\sigma, P_l\tau)}{f_l^\dagger : (P_l\sigma, P_l\tau)}$
- $\dfrac{f : (\sigma, P_u\tau)}{f_u^\dagger : (P_u\sigma, P_u\tau)}$
- $+_l : (P_l\sigma \times P_l\sigma, P_l\sigma)$
- $+_u : (P_u\sigma \times P_u\sigma, P_u\sigma)$
- $\otimes_l : (P_l\sigma \times P_l\tau, P_l(\sigma \times \tau))$
- $\otimes_u : (P_u\sigma \times P_u\tau, P_u(\sigma \times \tau))$
- $\mathsf{fold} : (\sigma[\mathsf{rec}\, t.\, \sigma/t], \mathsf{rec}\, t.\, \sigma)$
- $\mathsf{unfold} : (\mathsf{rec}\, t.\, \sigma, \sigma[\mathsf{rec}\, t.\, \sigma/t])$
- $\mathsf{Y} : (\sigma \to \sigma, \sigma)$

We now form an exogenous logic $\mathcal{DDL}$ (for *dynamic domain logic*, because of the evident analogy with dynamic logic [Pra81, Har79]). $\mathcal{DDL}$ is an extension of $\mathcal{L}$, the basic domain logic described in Section 2.

Formation Rules

We define the set of formulas DDL(σ) for each type σ.

- $L(\sigma) \subseteq \mathrm{DDL}(\sigma)$
- $\dfrac{f : (\sigma, \tau) \quad \psi \in \mathrm{DDL}(\tau)}{[f]\psi \in \mathrm{DDL}(\sigma)}$
- $t, f \in \mathrm{DDL}(\sigma)$
- $\dfrac{\phi, \psi \in \mathrm{DDL}(\sigma)}{\phi \wedge \psi, \phi \vee \psi \in \mathrm{DDL}(\sigma)}$

Axiomatization

The following axioms and rules are added to those of $\mathcal{L}$.

- $\dfrac{\phi \leq \psi}{[f]\phi \leq [f]\psi}$
- $[f] \bigwedge_{i \in I} \phi_i = \bigwedge_{i \in I} [f]\phi_i$
- $[f] \bigvee_{i \in I} \phi_i = \bigvee_{i \in I} [f]\phi_i$
- $[\mathsf{id}]\phi = \phi$
- $[f ; g]\phi = [f][g]\phi$
- $[<f, g>](\phi \times \psi) = [f]\phi \wedge [g]\psi$
- $[\mathsf{p}]\phi = (\phi \times t)$
- $[\mathsf{q}]\psi = (t \times \psi)$
- $\dfrac{(\phi \times \psi) \leq [f]\theta}{\phi \leq [\Lambda(f)](\psi \to \theta)}$
- $(\phi \to \psi) \times \phi \leq [\mathsf{Ap}]\psi$
- $[\mathsf{l}](\phi \oplus f) = \phi$
- $[\mathsf{l}](f \oplus \psi) = f \quad (\psi\downarrow)$
- $[\mathsf{r}](\phi \oplus f) = f \quad (\phi\downarrow)$
- $[\mathsf{r}](f \oplus \psi) = \psi$
- $[[f, g]]\phi = ([\mathsf{strict}(f)]\phi \oplus f) \vee (f \oplus [\mathsf{strict}(g)]\phi)$
- $\dfrac{\phi \leq [f]\psi}{\phi \leq [\mathsf{strict}(f)]\psi} \quad (\phi\downarrow)$
- $[\mathsf{up}](\phi)_\perp = \phi$
- $[\mathsf{lift}(f)]\phi = ([f]\phi)_\perp \quad (\phi\downarrow)$
- $[\{\!|\cdot|\!\}_l]\Diamond\phi = \phi$
- $[\{\!|\cdot|\!\}_u]\Box\phi = \phi$
- $\dfrac{\phi \leq [f]\Diamond\psi}{\Diamond\phi \leq [f_l^\dagger]\Diamond\psi}$
- $\dfrac{\phi \leq [f]\Box\psi}{\Box\phi \leq [f_u^\dagger]\Box\psi}$

- $[+_l]\Diamond\phi = (\Diamond\phi \times t) \vee (t \times \Diamond\phi)$
- $[+_u]\Box\phi = (\Box\phi \times \Box\phi)$
- $[\otimes_l]\Diamond(\phi \times \psi) = (\Diamond\phi \times \Diamond\psi)$
- $[\otimes_u]\Box(\phi \times \psi) = (\Box\phi \times \Box\psi)$
- $[\mathsf{fold}]\phi = \phi$
- $[\mathsf{unfold}]\phi = \phi$
- $\dfrac{\phi \leq [\mathsf{Y}]\psi}{\phi \wedge (\psi \rightarrow \theta) \leq [\mathsf{Y}]\theta}$

At this point, we could proceed to give a direct treatment of the semantics and meta-theory of $\mathcal{DDL}$, just as we did for the endogenous logic in Section 3. This would ignore the salient fact that our morphism term language and the typed λ-calculus presented in Section 3 are essentially *equivalent*. Instead, we shall give a translation of morphism terms into λ-terms. The idea is that a morphism term $f : (\sigma, \tau)$ is translated into a λ-term $(f)^\circ : \sigma \rightarrow \tau$.

Translation

$$\begin{array}{rcl}
(\mathsf{id})^\circ & = & \lambda x.x \\
(f;g)^\circ & = & \lambda x.(g)^\circ((f)^\circ x) \\
(1)^\circ & = & \lambda x.\star \\
(<f,g>)^\circ & = & \lambda x.((f)^\circ x, (g)^\circ x) \\
(\mathsf{p})^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\, x \\
(\mathsf{q})^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\, y \\
(\Lambda(f))^\circ & = & \lambda x.\lambda y.(f)^\circ(x,y) \\
(\mathsf{Ap})^\circ & = & \lambda f.\lambda x.fx \\
(\mathsf{l})^\circ & = & \lambda x.\imath(x) \\
(\mathsf{r})^\circ & = & \lambda y.\jmath(y) \\
([f,g])^\circ & = & \lambda z.\mathsf{cases}\ z\ \mathsf{of}\ \imath(x).\,(f)^\circ x\ \mathsf{else}\ \jmath(y).\,(g)^\circ y \\
(\mathsf{strict}(f))^\circ & = & \lambda z.\mathsf{cases}\ \imath((f)^\circ x)\ \mathsf{of}\ \imath(x).\,(f)^\circ x\ \mathsf{else}\ \jmath(y).\,y \\
(\mathsf{up})^\circ & = & \lambda x.\mathsf{up}(x) \\
(\mathsf{lift}(f))^\circ & = & \lambda y.\mathsf{lift}\ y\ \mathsf{to}\ \mathsf{up}(x).\,(f)^\circ x
\end{array}$$

$$
\begin{aligned}
(\{\!|\cdot|\!\}_l)^\circ &= \lambda x.\{\!|x|\!\}_l \\
(\{\!|\cdot|\!\}_u)^\circ &= \lambda x.\{\!|x|\!\}_u \\
(f_l^\dagger)^\circ &= \lambda z.\mathsf{over}\ z\ \mathsf{extend}\ \{\!|x|\!\}_l.\,(f)^\circ x \\
(f_u^\dagger)^\circ &= \lambda z.\mathsf{over}\ z\ \mathsf{extend}\ \{\!|x|\!\}_u.\,(f)^\circ x \\
(+_l)^\circ &= \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \uplus_l y \\
(+_u)^\circ &= \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \uplus_u y \\
(\otimes_l)^\circ &= \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \otimes_l y \\
(\otimes_u)^\circ &= \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \otimes_u y \\
(\mathsf{fold})^\circ &= \lambda x.\mathsf{fold}(x) \\
(\mathsf{unfold})^\circ &= \lambda x.\mathsf{unfold}(x) \\
(\mathsf{Y})^\circ &= \lambda f.\mu x.fx
\end{aligned}
$$

Semantics

Let $\mathcal{M}(\sigma, \tau)$ be the set of morphism terms of sort (σ, τ). Since

$$\mathbf{SDom}(\mathcal{D}(\sigma), \mathcal{D}(\tau)) \cong \mathcal{D}(\sigma \to \tau)$$

by cartesian closure, we can get a semantics

$$[\![\cdot]\!]_{\sigma\tau} : \mathcal{M}(\sigma, \tau) \longrightarrow \mathbf{SDom}(\mathcal{D}(\sigma), \mathcal{D}(\tau))$$

for morphism terms from the above translation. We use this to extend our semantics for $\mathcal{L}$ from Section 2 to $\mathcal{DDL}$:

$$[\![[f]\phi]\!] = ([\![f]\!])^{-1}([\![\phi]\!])$$

(the other clauses being handled in the obvious way). Note that the denotations of formulas in $\mathcal{DDL}$ are still *open* sets (continuity!), but need no longer be compact-open, since compactness is not preserved under inverse image in general.

This semantics yields a notion of validity for $\mathcal{DDL}$ assertions:

$$\models \phi \leq \psi \quad \equiv \quad [\![\phi]\!] \subseteq [\![\psi]\!].$$

Theorem 4.4.1 *$\mathcal{DDL}$ is sound:*

$$\mathcal{DDL} \vdash \phi \leq \psi \implies \models \phi \leq \psi$$

PROOF. The usual routine induction on the length of proofs. We give a few cases for illustration.

Left injection.

$$(i)\ [\![[\mathsf{l}](\phi \oplus f)]\!] = ([\![\mathsf{l}]\!])^{-1}([\![(\phi \oplus f)]\!]) = \{d : <0, d> \in [\![(\phi \oplus f)]\!]\} \cup \{\bot : \bot \in [\![(\phi \oplus f)]\!]\} = [\![\phi]\!].$$

$$(ii)\ \psi\downarrow \Rightarrow \bot \notin [\![\psi]\!] \Rightarrow ([\![\mathsf{l}]\!])^{-1}([\![(f \oplus \psi)]\!]) = \varnothing.$$

Strictification. Note that

$$[\![\mathsf{strict}(f)]\!]d = \begin{cases} \bot, & d = \bot \\ fd & \text{otherwise} \end{cases}$$

Now,

$$\phi\downarrow \Rightarrow \bot \notin [\![\phi]\!] \Rightarrow \forall d \in [\![\phi]\!].\ [\![\mathsf{strict}(f)]\!]d = fd,$$

which implies

$$[\![\phi]\!] \subseteq [\![[f]\psi]\!] \Leftrightarrow [\![\phi]\!] \subseteq [\![[\mathsf{strict}(f)]\psi]\!].$$

Union.

$$\begin{aligned} (i)\ [\![[+_l]\Diamond\phi]\!] &= \{(X, Y) : (X \cup Y) \cap [\![\phi]\!] \neq \varnothing\} \\ &= \{(X, Y) : X \cap [\![\phi]\!] \neq \varnothing \text{ or } Y \cap [\![\phi]\!] \neq \varnothing\} \\ &= \{(X, Z) : X \cap [\![\phi]\!] \neq \varnothing\} \\ &\quad \cup \{(Z, Y) : Y \cap [\![\phi]\!] \neq \varnothing\} \\ &= [\![(\Diamond\phi \times t) \vee (t \times \Diamond\phi)]\!] \end{aligned}$$

$$\begin{aligned} (ii)\ [\![[+_u]\Diamond\phi]\!] &= \{(X, Y) : X \cup Y \subseteq [\![\phi]\!]\} \\ &= \{(X, Y) : X \subseteq [\![\phi]\!]\ \&\ Y \subseteq [\![\phi]\!]\} \\ &= [\![(\Box\phi \times \Box\phi)]\!]. \end{aligned}$$

Recursion.

- $[\![\phi]\!] \subseteq [\![[\mathsf{Y}]\psi]\!]$

$\Rightarrow$ $\forall f \in [\![\phi]\!].\, \mathsf{Y}f \in [\![\psi]\!]$

$\Rightarrow$ $\forall f \in [\![\phi]\!] \cap [\![(\psi \to \theta)]\!].\, \mathsf{Y}f = f(\mathsf{Y}f) \in [\![\theta]\!].$ ∎

Next, we turn to what can be proved in the way of completeness. A *Hoare triple* in $\mathcal{DDL}$ is a formula $\phi \leq [f]\psi$ such that ϕ and ψ are formulas of $\mathcal{L}$, i.e. do not contain any program modalities.

Theorem 4.4.2 (Completeness For Hoare Triples) *Let $\phi \leq [f]\psi$ be a Hoare triple. Then*

$$\mathcal{DDL} \vdash \phi \leq [f]\psi \iff \models \phi \leq [f]\psi.$$

This result can either be proved directly, in similar fashion to Theorem 4.3.1; or it can be reduced to that result, since

$$\models \phi \leq [f]\psi \iff (f)^\circ, \Gamma_t \models (\phi \to \psi) \iff (f)^\circ, \Gamma_t \vdash (\phi \to \psi)$$

(where Γ_t is the constant map $x \mapsto t$). It thus suffices to prove:

$$(f)^\circ, \Gamma_t \vdash (\phi \to \psi) \implies \mathcal{DDL} \vdash \phi \leq [f]\psi.$$

In either approach, the argument is a straightforward variation on our work in section 3, which we omit since it adds nothing new.

Finally, we come to a limitative result, which differentiates $\mathcal{DDL}$ from the endogenous logic of Section 3, and shows that the restricted form of 4.4.2 is necessary. The result is of course not "surprising", since $\mathcal{DDL}$ is semantically more expressive than the endogenous logic, allowing the description of non-compact open sets.

Theorem 4.4.3 *The validity problem for $\mathcal{DDL}$ is Π^0_2-complete.*

Proof. We will need some notions on effectively given domains; see [Plo81, Chapter 7]. Firstly, each type expression in our meta-language has an effectively given domain as its denotation (since effectively given domains are closed under recursive definitions and all our type constructions [Plo81, Chapter 7 pp. 16, 21, Chapter 8 pp. 16, 54]). Similarly, each term $f : (\sigma, \tau)$

denotes a computable morphism from $\mathcal{D}(\sigma)$ to $\mathcal{D}(\tau)$. Moreover, each $\phi \in \mathcal{L}(\sigma)$ denotes a compact-open, and hence computable open set in $\mathcal{D}(\sigma)$; and computable open sets are closed under inverse images of computable maps [Plo81, Chapter 7 p. 9], and under finite unions and intersections [Plo81, Chapter 7 p. 7]. Thus each formula of $\mathcal{DDL}$ denotes a computable open set, and the problem of deciding the validity of the assertion $\phi \leq \psi$ can be reduced to that of deciding the inclusion of r.e. sets $[\![\phi]\!] \subseteq [\![\psi]\!]$, which as is well-known [Soa87, IV.1.6] is Π^0_2.

To complete the argument, we take a standard Π^0_2-complete problem, and reduce it to validity in $\mathcal{DDL}$. The problem we choose is

$$\mathsf{Tot} = \{x : W_x = \mathbb{N}\}$$

i.e. the set of codes of *total* recursive functions [Soa87, IV.3.2]. To perform the reduction, we proceed as follows:

- The type $\mathbb{N}_\perp \equiv \mathsf{rec}\,t.\,(\mathbf{1})_\perp \oplus t$ is used to model the flat domain of natural numbers.
- We can show that every partial recursive function $\varphi : \mathbb{N} \rightarrow \mathbb{N}$, thought of as a strict continuous function of type $\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp$, can be defined by a morphism term. This is quite standard: the numerals are constructed from the injections, lifting, and fold and unfold; the conditional and basic predicates from source tupling; and primitive recursion from general recursion (Y) and conditional. We omit the details.
- In particular, we can define a morphism term $N : (\mathbb{N}_\perp, \mathbb{N}_\perp)$ such that:

$$[\![N]\!]d = \begin{cases} \perp, & d = \perp \\ 0 & \text{otherwise} \end{cases}$$

- Now given a partial recursive function φ, represented by a morphism term f, the totality of φ is equivalent to the $\mathcal{DDL}$-validity of

$$N \leq [f][N]\bar{0}$$

where $\bar{0} \equiv ((t)_\perp \oplus f)$ (so $[\![\bar{0}]\!] = \{0\}$). ∎

4.5 Applications: The Logic of a Domain Equation

A denotational analysis of a computational situation results in the description of a domain which provides an appropriate semantic universe for this situation. Canonically, domains are specified by type expressions in a metalanguage. We can then use our approach to "turn the handle", and generate a logic for this situation in a quite mechanical way.

We shall now go on to develop two case studies of this kind, in the areas of concurrency (Chapter 5) and the λ-calculus (Chapter 6).

Chapter 5

Applications to Concurrency: A Domain Equation for Bisimulation

5.1 Introduction

Our aim in this Chapter is to treat some basic topics in the theory of concurrency from the point of view of domain logic. This will serve as a major case study for the general theory developed in the previous two Chapters; and will also weave another of the strands mentioned in Chapter 1 into our narrative. Our aim is not only to exemplify the general theory, but to *apply* it in order to shed some new light on concurrency. In particular, we shall study *bisimulation* [Par81, Mil83, HM85]. This notion has emerged as one of the more stable and mathematically natural concepts to have been formulated in the study of concurrency over the past decade. It is commonly accepted as the *finest* extensional or behavioural equivalence on processes one would want to impose. To date, bisimulation has been studied almost exclusively from the operational and logical points of view. Our aim is to show that this notion can be captured elegantly in the setting of domain theory, using Plotkin's powerdomain construction [Plo76]. Moreover, we shall make extensive use of the logical form of domain theory developed in the previous Chapter. Thus our motivation can be summarised as follows:

- To show that more can be done in the sphere of concurrency using domain-theoretic and denotational methods than seems to be commonly realised.

- To analyze the apparently *ad hoc* and "application oriented" notions of bisimulation over labelled transition systems and Hennessy-Milner logic by means of the general, mathematically basic, and "reusable" notions of domain theory, specifically type constructions and the solution of recursive domain equations.

- To form part of our general programme of connecting

  1. Domain theory and operational notions of observability
  2. Denotational semantics and program logics.

  This programme is made systematic by using the information conveyed in the syntactic description of domains by type expressions. It can be argued that a full domain-theoretic analysis of some computational situation is only obtained when we have written down an explicit type expression, rather than using some *ad hoc* construction of a cpo. At any rate, the benefits which flow from having such a description are very considerable. Using the ideas developed in the previous Chapter, we can derive a propositional theory from the type expression, and use this to explore the "observational logic" of the computational situation.

We now summarise the further contents of the Chapter. After reviewing some basic notions on transition systems etc., we introduce a domain of synchronisation trees defined by means of a domain equation (recursive type expression). Then we present a domain logic for transition systems, which is derived from this domain equation in the sense of Chapter 3. The main result of section 4 is that the finitary part of this logic is the Stone dual of our domain of synchronisation trees.

In section 5, we present a number of applications of this logic. It is shown to be equivalent to Hennessy-Milner logic in the infinitary case, and hence to characterise bisimulation. In the finitary case, it more powerful than Hennessy-Milner logic, and we obtain a more satisfactory characterisation result for it; namely, it is shown to characterise the "finitary part" of bisimulation for *all* transition systems.

We also develop an extension of Hennessy-Milner logic which is equivalent to the finitary domain logic. The infinitary domain logic is then used to *axiomatize* a suitable notion of "finitary transition system". These systems are shown indeed to be finitary in a strong sense — their bisimulation preorders are algebraic. Finally, the domain of synchronisation trees (i.e. the spectral space of the logic) is shown to be finitary *qua* transition system, and moreover to be *final* in a suitable category of such systems. This yields a syntax-free "universal semantics" for transition systems, which is fully abstract with respect to bisimulation.

In section 6, we give a conventional (syntax-directed) denotational semantics for the concurrent calculus SCCS [Mil83], based on our domain of synchronisation trees. A full abstraction result is proved for this semantics; as a by-product, our domain is shown to be isomorphic to Hennessy's term model [Hen81].

5.2 Transition Systems and Related Notions

We begin with the basic notion of a labelled transition system (with divergence), which abstracts from the operational semantics of many concurrent calculi.

Definition 5.2.1 A *transition system* is a structure

$$(\mathrm{Proc}, \mathsf{Act}, \rightarrow, \uparrow)$$

where:

- Proc is a set of *processes* or *agents*.
- Act is a set of atomic *actions* or *experiments*.
- $\rightarrow \subseteq \mathrm{Proc} \times \mathsf{Act} \times \mathrm{Proc}$ (notation: $p \xrightarrow{a} q$).
- $\uparrow \subseteq \mathrm{Proc}$ (notation: $p\uparrow$).

We write

$$p\downarrow \equiv \neg(p\uparrow).$$

We read $p \xrightarrow{a} q$ as "p has the capability to do a and become (i.e. change state to) q"; $p\uparrow$ as "p may diverge"; and $p\downarrow$ as "p definitely converges". We define

$$\mathsf{sort}(p) \equiv \{a \in \mathsf{Act} \mid \exists q, r.\, p \rightarrow^{\star} q \xrightarrow{a} r\}$$

where $p \rightarrow q \equiv \exists a \in \mathsf{Act}.\, p \xrightarrow{a} q$, and $\rightarrow^{\star}$ is the reflexive, transitive closure of $\rightarrow$.

We now define a number of finiteness conditions on transition systems:

image-finiteness	$\forall p \in \mathrm{Proc}, a \in \mathsf{Act}.\, \{q \mid p \xrightarrow{a} q\}$ is finite.
sort-finiteness	$\forall p \in \mathrm{Proc}.\, \mathsf{sort}(p)$ is finite.
finite-branching	$\forall p \in \mathrm{Proc}.\, \{q \mid p \rightarrow q\}$ is finite.
initials-finiteness	$\forall p \in \mathrm{Proc}.\, \{a \in \mathsf{Act} \mid \exists q.\, p \xrightarrow{a} q\}$ is finite.

Each of these properties has a weak form, obtained by making it conditional on convergence. For example:

weak image-finiteness $\forall p \in \text{Proc}, a \in \mathsf{Act}.\, p{\downarrow} \;\Rightarrow\; \{q \mid p \xrightarrow{a} q\}$ is finite.

We now introduce a particularly useful source of examples for transition systems, the *synchronisation trees*. Given a set Act of actions, $\mathsf{ST}_\infty(\mathsf{Act})$, the synchronisation trees over Act, are defined as the (proper) class of infinitary terms generated by the following inductive definition:

$$\frac{\{a_i \in \mathsf{Act}, t_i \in \mathsf{ST}_\infty(\mathsf{Act})\}_{i\in I}}{\sum_{i\in I} a_i t_i \; [+\Omega] \in \mathsf{ST}_\infty(\mathsf{Act})} \tag{5.1}$$

where $[+\Omega]$ means optional inclusion of Ω as a summand (i.e. there are really two clauses in this definition). We write

$$\begin{aligned} \mathbb{O} &\equiv \sum_{i\in\varnothing} a_i t_i \\ \Omega &\equiv \sum_{i\in\varnothing} a_i t_i + \Omega. \end{aligned}$$

The subclass of terms formed using only finite sums is denoted $\mathsf{ST}_\omega(\mathsf{Act})$. Given a synchronisation tree t formed according to 5.1, we stipulate:

- $t{\uparrow}$ iff Ω is included as a summand.
- $t \xrightarrow{a_i} t_i$ for each summand $a_i t_i$ $(i \in I)$.

This defines a (large) transition system $(\mathsf{ST}_\infty(\mathsf{Act}), \mathsf{Act}, \rightarrow, \uparrow)$; restriction to a subset of synchronisation trees yields a small transition system. In particular, by choosing a canonical system of representatives for $\mathsf{ST}_\omega(\mathsf{Act})$ which is closed under subtrees we obtain a countable transition system of finite synchronisation trees, which by abuse of notation we refer to also as $\mathsf{ST}_\omega(\mathsf{Act})$.

We are now ready to introduce the main concept we will study.

Definition 5.2.2 ([Par81, Mil80, Mil81]) A relation $R \subseteq \text{Proc} \times \text{Proc}$ is a *prebisimulation* if, for all $p, q \in \text{Proc}$:

$pRq \implies \forall a \in \mathsf{Act}.$

- $p \xrightarrow{a} p' \implies \exists q'.\, q \xrightarrow{a} q' \;\&\; p'Rq'$
- $p{\downarrow} \implies q{\downarrow} \;\&\; [q \xrightarrow{a} q' \;\Rightarrow\; \exists p'.\, p \xrightarrow{a} p' \;\&\; p'Rq']$.

We write

$$p \lesssim^B q \equiv \exists R.\, R \text{ is a prebisimulation and } pRq.$$

For an alternative description of $\lesssim^B$, let $Rel(\text{Proc})$ be the set of all binary relations over Proc; this is a complete lattice under set inclusion. Now define

$$F : Rel(\text{Proc}) \to Rel(\text{Proc})$$

$$\begin{array}{rcl} F(R) & = & \{(p,q) \mid \forall a \in \mathsf{Act}. \\ & & \bullet\ p \xrightarrow{a} p' \ \Rightarrow\ \exists q'.\, q \xrightarrow{a} q' \;\&\; p'Rq' \\ & & \bullet\ p{\downarrow}\ \Rightarrow\ q{\downarrow} \;\&\; [q \xrightarrow{a} q' \ \Rightarrow\ \exists p'.\, p \xrightarrow{a} p' \;\&\; p'Rq']\}. \end{array}$$

Clearly, R is a prebisimulation iff $R \subseteq F(R)$, i.e. R is a *pre-fixed point of* F. Since F is monotone, by Tarski's Theorem it has a maximal fixpoint, given by $\bigcup\{R \mid R \subseteq F(R)\}$, i.e. $\lesssim^B$. Thus $\lesssim^B$ is itself a prebisimulation, and evidently the largest one. Moreover, it is reflexive and transitive; the corresponding equivalence is denoted $\sim^B$.

We can also describe $\lesssim^B$ more explicitly, in terms of iterations of F. We define relations $\lesssim_\alpha$, $(\alpha \in \mathsf{Ord})$ (the class of ordinals), by the following ordinal recursion:

- $p \lesssim_0 q$ always (i.e. $\lesssim_0 = \text{Proc} \times \text{Proc}$, the top element in the lattice $Rel(\text{Proc})$).

- $p \lesssim_{\alpha+1} q$ iff

$$\begin{array}{l} \forall a \in \mathsf{Act}. \\ \bullet\ p \xrightarrow{a} p' \ \Longrightarrow\ \exists q'.\, q \xrightarrow{a} q' \;\&\; p' \lesssim_\alpha q' \\ \bullet\ p{\downarrow} \ \Longrightarrow\ q{\downarrow} \;\&\; [q \xrightarrow{a} q' \ \Rightarrow\ \exists p'.\, p \xrightarrow{a} p' \;\&\; p' \lesssim_\alpha q']. \end{array}$$

  (i.e. $\lesssim_{\alpha+1} = F(\lesssim_\alpha)$).

- For limit λ, $p \lesssim_\lambda q$ iff $\forall \alpha < \lambda.\, p \lesssim_\alpha q$ (i.e. $\lesssim_\lambda = \bigcap_{\alpha<\lambda} \lesssim_\alpha$).

This sequence of relations is decreasing, and bounded below by $\lesssim^B$; i.e. for all α

$$\lesssim_\alpha \supseteq \lesssim_{\alpha+1} \supseteq \lesssim^B.$$

For any (small) transition system the sequence is eventually stationary; for some λ, for all $\alpha > \lambda$, $\lesssim_\alpha = \lesssim_\lambda$. The least ordinal λ for which this holds is called the *closure ordinal* [Mos74]; and we have $\lesssim_\lambda = \lesssim^B$. Note that each $\lesssim_\alpha$ is relexive and transitive.

The relations $\lesssim^B$ and $\sim^B$ have been defined in the context of a given transition system. However, we frequently want to use them to compare processes from different transition systems. This is easily accomplished by forming the disjoint union of the two systems, and then using $\lesssim^B$ as defined above. In the sequel, we will do this without further comment.

We now introduce a program logic due to Hennessy and Milner [HM85]. The idea is to obtain a characterisation of $\lesssim^B$ in terms of a suitable notion of *property* of process; $p \lesssim^B q$ iff every property satisfied by p is satisfied by q.

Definition 5.2.3 Given a set of actions Act, the language $\mathrm{HML}_\infty(\mathsf{Act})$ (we henceforth elide the parameter Act) is defined by the following inductive clauses:

$$\frac{a \in \mathsf{Act},\ \phi \in \mathrm{HML}_\infty}{[a]\phi, \langle a \rangle\phi \in \mathrm{HML}_\infty}$$

$$\frac{\phi_i \in \mathrm{HML}_\infty\ (i \in I)}{\bigwedge_{i \in I} \phi_i, \bigvee_{i \in I} \phi_i \in \mathrm{HML}_\infty}$$

In particular, we write:

$$\begin{aligned} t &\equiv \bigwedge_{i \in \varnothing} \phi_i \\ f &\equiv \bigvee_{i \in \varnothing} \phi_i. \end{aligned}$$

We use the subscript ∞ to indicate the presence of infinite conjunctions and disjunctions. We write HML_ω for the sublanguage obtained by restricting the formation rules to finite conjunctions and disjunctions.

We now define a satisfaction relation $\models \subseteq \text{Proc} \times \text{HML}_\infty$.

$$
\begin{array}{lcl}
p \models \bigwedge_{i\in I} \phi_i & \equiv & \forall i \in I.\, p \models \phi_i \\
p \models \bigvee_{i\in I} \phi_i & \equiv & \exists i \in I.\, p \models \phi_i \\
p \models <a>\phi & \equiv & \exists q.\, p \xrightarrow{a} q \;\&\; q \models \phi \\
p \models [a]\phi & \equiv & \forall q.\, p \xrightarrow{a} q \implies q \models \phi.
\end{array}
$$

We write

$$
\text{HML}_\infty(p) \equiv \{\phi \in \text{HML}_\infty : p \models \phi\}
$$

plus obvious variations on this notation.

We define two useful assignments of ordinals to formulas in HML_∞, the *modal depth*:

$$
\begin{array}{lclcl}
\mathsf{md}(\bigwedge_{i\in I} \phi_i) & \equiv & \mathsf{md}(\bigvee_{i\in I} \phi_i) & \equiv & \sup\{\mathsf{md}(\phi_i) : i \in I\} \\
\mathsf{md}([a]\phi) & \equiv & \mathsf{md}(<a>\phi) & \equiv & \mathsf{md}(\phi) + 1
\end{array}
$$

and the *height*:

$$
\begin{array}{lclcl}
\mathsf{ht}(\bigwedge_{i\in I} \phi_i) & \equiv & \mathsf{ht}(\bigvee_{i\in I} \phi_i) & \equiv & \sup\{\mathsf{ht}(\phi_i) : i \in I\} + 1 \\
\mathsf{ht}([a]\phi) & \equiv & \mathsf{ht}(<a>\phi) & \equiv & \mathsf{ht}(\phi) + 1.
\end{array}
$$

We define $\mathsf{sort}(\phi)$ to be the set of action symbols which occur in ϕ.

Now given a set $A \subseteq \mathsf{Act}$ and an ordinal λ, we define a sublanguage of HML_∞:

$$
\text{HML}_\infty^{(A,\lambda)} = \{\phi \in \text{HML}_\infty : \mathsf{sort}(\phi) \subseteq A \;\&\; \mathsf{md}(\phi) \leq \lambda\}.
$$

We are now ready to prove a generalised and strengthened version of the Modal Characterisation Theorem [Mil81, Mil85, HM85].

Theorem 5.2.4 (Modal Characterisation Theorem) *Suppose that* $A \subseteq \mathsf{Act}$ *satisfies*

$$
\mathsf{sort}(p) \cup \mathsf{sort}(q) \subseteq A \neq \varnothing;
$$

then

$$p \lesssim_\lambda q \iff \mathrm{HML}_\infty^{(A,\lambda)}(p) \subseteq \mathrm{HML}_\infty^{(A,\lambda)}(q).$$

As an immediate consequence we obtain

$$p \lesssim^B q \iff \mathrm{HML}_\infty(p) \subseteq \mathrm{HML}_\infty(q).$$

Proof. The left-to-right implication is proved by induction on λ. The cases for $\lambda = 0$, λ a limit ordinal are trivial. For $\lambda = \alpha + 1$, we argue by induction on $\mathsf{ht}(\phi)$. The cases for $\bigwedge_{i \in I} \phi_i$, $\bigvee_{i \in I} \phi_i$ are trivial. Suppose $p \models <a>\phi$. Then for some p', $p \stackrel{a}{\rightarrow} p'$ and $p \models \phi$. Since $p \lesssim_\lambda q$, for some q', $q \stackrel{a}{\rightarrow} q'$ and $p' \lesssim_\alpha q'$. By the outer induction hypothesis, $q' \models \phi$, hence $q \models <a>\phi$, as required. The case for $[a]\phi$ is similar.

For the converse, we argue by induction on λ. Suppose $p \not\lesssim_\lambda q$: we must find $\phi \in \mathrm{HML}_\infty^{(A,\lambda)}(p) - \mathrm{HML}_\infty^{(A,\lambda)}(q)$.
Case 1: $p \stackrel{a}{\rightarrow} p'$ and for all q', $q \stackrel{a}{\rightarrow} q'$ implies $p' \not\lesssim_\alpha q'$ for some $\alpha < \lambda$. By induction hypothesis, for each such q' there is $\phi \in \mathrm{HML}_\infty^{(A,\alpha)}(p') - \mathrm{HML}_\infty^{(A,\alpha)}(q')$. Now take

$$\phi = <a> \bigwedge \{\phi_{q'} : q \stackrel{a}{\rightarrow} q'\}.$$

Case 2: $p\!\downarrow$ and $p\!\uparrow$. Take $\phi \equiv [a]t$, for any $a \in A$.
Case 3: $p\!\downarrow$, $q\!\downarrow$, $q \stackrel{a}{\rightarrow} q'$, and for all p', $p \stackrel{a}{\rightarrow} p'$ implies $p' \not\lesssim_\alpha q'$ for some $\alpha < \lambda$. Defining $\phi_{p'}$ analogously to Case 1,

$$\phi = [a] \bigvee \{\phi_{p'} : p \stackrel{a}{\rightarrow} p'\}. \quad \blacksquare$$

The reader familiar with infinitary logic will recognise the strong similarity between this result and Karp's Theorem [Bar75]. Similar remarks apply to "Master Formula Theorems" as in [Rou85], *vis a vis* the Scott Isomorphism Theorem [Bar75].

Note that, if A is a finite set and λ a finite ordinal, then (up to logical equivalence) $\mathrm{HML}_\infty^{(A,\lambda)}$ is finite. It follows easily from this observation that each formula in $\mathrm{HML}_\infty^{(A,\lambda)}$ is equivalent to one in $\mathrm{HML}_\omega^{(A,\lambda)}$. Hence as a Corollary to the Characterisation Theorem we obtain

Theorem 5.2.5 *[Abr87b] If the transition system is sort-finite, then*

$$p \lesssim_\omega q \iff \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q).$$

Moreover, we have the following result from [HM85]:

Theorem 5.2.6 *If the transition system is image-finite, then*

$$(i) \quad \lesssim_\omega = \lesssim^B$$
$$(ii) \quad p \lesssim_\omega q \iff \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q).$$

Unfortunately, if unguarded recursion is allowed in any of the standard concurrent calculi (SCCS, CCS, CSP, etc.) they are neither image-finite nor sort-finite (though sort-finiteness may be regained e.g. for CCS by imposing fairly mild restrictions on the relabelling operators). Thus these two Theorems cannot be applied. To see how weak finitary Hennessy-Milner logic is when the set of actions is finite, consider the following
Example.

$$\begin{array}{rcl} p & \equiv & a\mathbb{O} + \Omega \\ q & \equiv & \sum_{n \in \omega} a b_n \mathbb{O} + \Omega \end{array}$$

where we assume $b_m \neq b_n$ for $m \neq n$. Now $p \not\lesssim_2 q$, but we have

Proposition 5.2.7 $\mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q)$.

In order to prove this Proposition we need a lemma.

Lemma 5.2.8 *Every formula in* $\mathrm{HML}_\omega(\mathbb{O})$ *is satisfied by cofinitely many of the* $b_n\mathbb{O}$.

Proof. By induction on formulas in $\mathrm{HML}_\omega(\mathbb{O})$. For conjunctions and disjunctions, the intersection and union of finitely many cofinite sets are cofinite. (It is the case for conjunction which necessitates the strength of statement of the Lemma). The case for $<b>\phi$ is vacuous. For $[b]\phi$, cofinitely many (in fact, all but at most one) of the $b_n\mathbb{O}$ do not have a b-action, hence satisfy $[b]\phi$. ∎

The Proposition can now be proved by induction on formulas in HML_ω. The only non-trivial case is $<a>\phi$, which follows from the Lemma.

The deficiency of Hennessy-Milner logic illustrated by this example is disturbing, because processes generated by a finitary calculus (including p and q above) should be adequately modelled by a finitary semantics and logic. This suggests that Hennessy-Milner logic is not quite right as it stands.

5.3 A Domain Equation for Synchronisation Trees

In this section, we shall define a domain of synchronisation trees, and establish some of its basic properties. Since our definitions will use the Plotkin powerdomain, we need to work in a category which is closed under this construction. This means that we cannot use **SDom**, as we did in the previous two Chapters. Instead, we will use **SFP**. The only facts about **SFP** which we will need are that it is a category of algebraic domains closed under the following type constructions:

Separated Sum

Let A be a countable set, and $\{D_a\}_{a \in A}$ an A-indexed family of domains. Then $\sum_{a \in A} D_a$ is formed by taking the disjoint union of the D_a and adjoining a bottom element. We shall write elements of the disjoint union as $<a, d>$ $(a \in A,\ d \in D_a)$. Note that the ordering is defined so that

$$<a, d> \sqsubseteq <a', d'> \iff a = a' \;\&\; d \sqsubseteq_{D_a} d'.$$

- For each $a \in A$, the function

  $$D_a \to \sum_{a \in A} D_a$$

  $$d \mapsto <a, d>$$

  is continuous.

- Separated sum is functorial; given a family

  $$f_a : D_a \to E_a \quad (a \in A),$$

  $$\sum_{a \in A} f_a : \sum_{a \in A} D_a \to \sum_{a \in A} E_a$$

  is defined by:

  $$\begin{array}{lcl} (\sum_{a \in A} f_a)\bot & = & \bot \\ (\sum_{a \in A} f_a)<a, d> & = & <a, f_a d>. \end{array}$$

The Plotkin Powerdomain

We write $P[D]$ for the Plotkin powerdomain over D. Although this construction is best *characterised* abstractly, as in [HP79], for purposes of comparison with more concrete operational notions a good representation is invaluable. This is provided in [Plo76, Plo81].

Definition 5.3.1 For an algebraic domain D the *Lawson topology* on D is generated by the sub-basic sets

$$\uparrow b, \quad D - \uparrow b$$

for finite $b \in D$ (so the Lawson topology refines the Scott topology). We will write the closure operator associated with the Lawson topology as Cl. (NB: in [Plo76], the Lawson topology is called the Cantor topology).

Definition 5.3.2 For $X \subseteq D$,

$$\begin{array}{lll} (i) & Con(X) & \equiv \{d : \exists d_1, d_2 \in X.\, d_1 \sqsubseteq d \sqsubseteq d_2\} \\ (ii) & X^\star & \equiv Con \circ Cl. \end{array}$$

X is said to be

- *Lawson-closed* if $X = Cl\ X$
- *Convex-closed* if $X = Con\ X$
- *Closed* if $X = X^\star$.

Definition 5.3.3 The *Egli-Milner order*. For $X, Y \subseteq D$:

$$X \sqsubseteq_{EM} Y \quad \equiv \quad \forall x \in X.\, \exists y \in Y.\, x \sqsubseteq y \ \&\ \forall y \in Y.\, \exists x \in X.\, x \sqsubseteq y.$$

The representation of the Plotkin powerdomain can now be defined as follows:

$$P[D] \quad \equiv \quad (\{X \subseteq D : X \neq \varnothing, X = X^\star\}, \sqsubseteq_{EM}).$$

There are also a number of (continuous) operations associated with the Plotkin powerdomain, which we shall describe in terms of our representation of $P[D]$.

- Firstly, P is *functorial*: given $f\ D \to E$,

  $$Pf : P[D] \to P[E]$$

  is defined by

  $$Pf(X) \quad \equiv \quad \{f(x)|x \in X\}^{\star}.$$

- *Singleton:*

  $$\{\!|.|\!\} : D \to P[D]$$

  is defined by

  $$\{\!|d|\!\} \quad \equiv \quad \{d\}^{\star} = \{d\}.$$

- *Union:*

  $$\uplus : P[D]^2 \to P[D]$$

  is defined by

  $$X \uplus Y \quad \equiv \quad (X \uplus Y)^{\star} = Con(X \cup Y).$$

- *Big Union:*

  $$\biguplus : P[P[D]] \to P[D]$$

  is defined by

  $$\biguplus(\Theta) \quad \equiv \quad (\bigcup \Theta)^{\star} = Con(\bigcup \Theta).$$

- *Tensor Product* [HP79]. We will only need the following: given

  $$f : D^n \to D$$

  the *multilinear extension*

  $$f^{\dagger}\ P[D]^n \to P[D]$$

  is defined by

  $$f^{\dagger}(X_1, \ldots, X_n) \quad \equiv \quad \{f(x_1, \ldots, x_n) : x_i \in X_i\}^{\star}.$$

  (Note that for $n = 1$, $f^{\dagger} = Pf$.) This extension has the property

  $$\begin{aligned} f^{\dagger}(X_1, \ldots, X_i \uplus X_i', \ldots, X_n) \quad &= \quad f^{\dagger}(X_1, \ldots, X_i, \ldots, X_n) \\ &\qquad \uplus f^{\dagger}(X_1, \ldots, X_i', \ldots, X_n) \end{aligned}$$

  for $(1 \leq i \leq n)$.

Adjoining the empty set

To the best of my knowledge, the only significant precursor of our work in this Chapter is [MM79]. The main reason that something like our present programme could not have been carried through in their framework is that, because of a technical problem, they used the Smyth rather than the Plotkin powerdomain. This rules out any hope of gaining a correspondence with bisimulation. The technical problem is that of adjoining the empty set to the powerdomain to model the convergent process with no actions (NIL in CCS [Mil80], $\mathbb{O}$ in SCCS [Mil83], STOP in CSP [Hoa85], δ in ACP [BK84], etc.). If we add the empty set to our representation of $P[D]$, it is not related to anything except itself under $\sqsubseteq_{EM}$; in category-theoretic terms, the problem is the non-existence of a certain free construction ([Plo81]). Fortunately, we do not need these non-existent solutions. We shall adjoin the empty set to the Plotkin powerdomain in a way which has two advantages:

1. There is no theoretical overhead, since it is definable as a derived operation from standard type constructions.

2. It works, i.e. is exactly suited to our semantic purposes, as the results to follow will show.

For motivation, consider a transition system $(\mathrm{Proc}, \mathsf{Act}, \rightarrow, \uparrow)$ and processes $p, r \in \mathrm{Proc}$ such that

$$(i) \quad p\uparrow,\ r\downarrow$$
$$(ii) \quad p \not\rightarrow,\ r \not\rightarrow .$$

Then it is easy to see that, for all $q \in \mathrm{Proc}$:

$$\begin{aligned} (i) \quad r \lesssim^B q &\iff r \sim^B q \\ (ii) \quad q \lesssim^B r &\iff q \not\rightarrow \\ &\iff q \sim^B p \text{ or } q \sim^B r. \end{aligned}$$

This suggests the following

Definition 5.3.4 $P^0[D]$, the Plotkin powerdomain with empty set. Representation of $P^0[D]$:

Elements	$\{X \subseteq D : X = X^\star\} = P[D] \cup \{\varnothing\}$.
Ordering	$X \sqsubseteq Y \ \equiv\ X = \{\bot\}$ or $X \sqsubseteq_{EM} Y$.

Observation 5.3.5 $P^0[D] \ \cong\ (\mathbf{1})_\bot \oplus P[D]$.

In principle, we could work throughout with 3.5 as the *definition* of $P^0[D]$; in practice, it is much more convenient to work with the representation given by 3.4. This requires that we extend our definitions of the powerdomain operations to work on $P^0[D]$. In fact, all of the definitions following 3.3 still make sense for $P^0[D]$. It is easily checked that $\uplus$, $\biguplus$ and $\{\!|\cdot|\!\}$ are continuous on $P^0[D]$. For $P^0 f$ and $f^\dagger$ a technical point arises, which is not specific to 3.4, but stems from the use of coalesced sum in 3.5. As is well known, coalesced sum is functorial only on the category of *strict* functions. Hence we can only use $P^0 f$ if f is strict, and $f^\dagger$ if f is strict in each argument separately. With these provisos, the extended operations are continuous.

Notation. We use $\emptyset$ to denote the empty set in $P^0[D]$; if I is a finite index set, we write

$$\biguplus_{i \in I} X_i$$

meaning the iterated use of $\uplus$ (which is associative, commutative and idempotent on $P^0[D]$, just as it is on $P[D]$) if $I \neq \varnothing$, and $\emptyset$ otherwise. Also, we write

$$\{\!| d : A |\!\}$$

where $d \in D$ and A is some sentence, meaning $\{\!| d |\!\}$ if A is true, and $\emptyset$ otherwise.

We are now ready for the main definition of the section.

Definition 5.3.6 Let Act be a *countable* set of actions. Then $\mathcal{D}(\mathsf{Act})$, the domain of synchronisation trees over Act (we henceforth omit the parameter Act), is defined to be the initial solution of the domain equation

$$\mathcal{D} \;\cong\; P^0[\sum_{a \in \mathsf{Act}} \mathcal{D}]. \tag{5.2}$$

Here the sum $\sum_{a \in \mathsf{Act}} \mathcal{D}$ is the "copower" of Act copies of $\mathcal{D}$. The equation is essentially that of [MM79], minus the value passing and with a different powerdomain.

How can we relate this domain equation to the formalism of Chapter 4? Suppose we extend the metalanguage of types introduced there with a constructor $P_p(\cdot)$ for the Plotkin powerdomain. Then we can write

$$\mathcal{D} \;\equiv\; \mathsf{rec}\, t.(\mathbf{1})_\perp \oplus P_p[\sum_{a \in \mathsf{Act}} t]$$

using 3.5 to eliminate P^0. This is not yet a valid type expression because of the sum

$$\sum_{a \in \mathsf{Act}} t \tag{5.3}$$

Let us take the main case of interest, where Act is countably infinite, say $\mathsf{Act} = \{a_n\}_{n \in \omega}$. Then we can replace 5.3 by the recursive expression

$$\mathsf{rec}\, u.(t)_\perp \oplus u \tag{5.4}$$

yielding the overall expression

$$\mathcal{D} \;\equiv\; \mathsf{rec}\, t.(\mathbf{1})_\perp \oplus P_p[\mathsf{rec}\, u.(t)_\perp \oplus u] \tag{5.5}$$

the intention being that the i'th summand as we unfold 5.4 corresponds to $a_i \in \mathsf{Act}$.

The reader will by now probably appreciate our efforts to streamline the presentation. Nevertheless, we regard the "closed form" expression 5.5 as fundamental, and the logic we shall introduce in the next section could be derived mechanically from it in the manner detailed in Chapter 4.

In the remainder of this section, we shall apply some standard domain-theoretic methods to elucidate the structure of $\mathcal{D}$.

Notation. We write $\perp$ for the bottom element of $\sum_{a\in\mathsf{Act}} \mathcal{D}$; $\{\!|\perp|\!\}$ is then the bottom element of $P^0[\sum_{a\in\mathsf{Act}} \mathcal{D}]$.

How can we unpack the structure of $\mathcal{D}$ from the domain equation 5.2? This is best done in two parts:

1. A *specified* isomorphism pair

$$\mathcal{D} \underset{\theta}{\overset{\eta}{\rightleftarrows}} P^0[\sum_{a\in\mathsf{Act}} \mathcal{D}].$$

   In fact, we shall elide η and θ, and treat 5.2 as an identity; this is only a notational convenience, and the reader can put η and θ back without encountering any difficulties.

2. *Initiality.* The categorical framework is clumsy to work with for our purposes. Instead, we will use an "intrinsic" (or in the terminology of [SP82] a "local" or "**O**-notion") formulation.

Definition 5.3.7 We define a sequence of functions

$$\pi_k : \mathcal{D} \to \mathcal{D}$$

as follows:

$$\begin{aligned} \pi_0 &\equiv \lambda x \in \mathcal{D}.\{\!|\perp|\!\} \\ \pi_{k+1} &\equiv P^0 \textstyle\sum_{a\in\mathsf{Act}} \pi_k. \end{aligned}$$

Note that $\sum_{a \in \mathsf{Act}}$ always produces a strict function, so this is well-defined.

Now the following proposition is standard ([Plo81, Chapter 5 Theorem 3]):

Proposition 5.3.8 *$\mathcal{D}$ is the "internal colimit" of the π_k:*

$$\begin{array}{rl} (i) & \text{Each } \pi_k \text{ is continuous and } \pi_k \sqsubseteq \pi_{k+1} \\ (ii) & \bigsqcup_k \pi_k = \mathsf{id}_{\mathcal{D}} \\ (iii) & \pi_k \circ \pi_k = \pi_k \\ (iv) & \forall d_1, d_2 \in \mathcal{D}.\, d_1 \sqsubseteq d_2 \iff \forall k.\, \pi_k d_1 \sqsubseteq \pi_k d_2. \end{array}$$

In particular, we will use part (iv) of this Proposition as the cutting edge of initiality.

Next, it will be useful to have an explicit description of the finite elements of $\mathcal{D}$, which, as already noted, is in **SFP**, and hence algebraic.

Definition 5.3.9 $K(\mathcal{D}) \subseteq \mathcal{D}$ is defined inductively as follows:

- $\emptyset \in K(\mathcal{D})$
- $\{\!|\bot|\!\} \in K(\mathcal{D})$
- $a \in \mathsf{Act}, d \in K(\mathcal{D}) \Rightarrow \{\!|<a, d>|\!\} \in K(\mathcal{D})$
- $d_1, d_2 \in K(\mathcal{D}) \Rightarrow d_1 \uplus d_2 \in K(\mathcal{D})$.

The following is again standard:

Proposition 5.3.10 *$K(\mathcal{D})$ is exactly the set of finite elements of $\mathcal{D}$.*

Finally, we consider $\mathcal{D}$ as a *transition system* $(\mathcal{D}, \mathsf{Act}, \rightarrow, \uparrow)$ defined by:

- $d \xrightarrow{a} d' \equiv <a, d'> \in d$
- $d\uparrow \equiv \bot \in d$.

Proposition 5.3.11 *$\mathcal{D}$ is "internally fully abstract", i.e.*

$$\forall d_1, d_2 \in \mathcal{D}\ .\ d_1 \lesssim^B d_2 \iff d_1 \sqsubseteq d_2.$$

Proof. We shall prove

$$(1)\ \ \forall k.\ d_1 \lesssim_k d_2 \implies \pi_k d_1 \sqsubseteq \pi_k d_2$$

and

$$(2)\ \ \sqsubseteq\ \subseteq\ \lesssim^B.$$

Clearly (1) implies

$$(3)\ \ \lesssim_\omega\ \subseteq\ \sqsubseteq$$

by 5.3.8(iv), and since

$$(4)\ \ \lesssim^B\ \subseteq\ \lesssim_\omega,$$

we obtain $\lesssim^B = \sqsubseteq$, as required.

(1). By induction on k. The basis is trivial. For the inductive step, assume $d \lesssim_{k+1} e$. Now $d = \emptyset$ and $d \lesssim_{k+1} e$ implies $e = \emptyset$, while $d = \{\!|\bot|\!\}$ implies $d \sqsubseteq e$, so we may assume $d \neq \emptyset \neq e$, and it suffices to prove $d \sqsubseteq_{EM} e$.

From the definitions we have $\pi_{k+1} d = X^\star$, where

$$X = \{<a, \pi_k d'> : <a, d'> \in d\} \cup \{\bot : \bot \in d\},$$

and similarly $\pi_{k+1} e = Y^\star$. Now

$$\begin{aligned}
\bullet \quad & <a, \pi_k d'> \in X \\
\implies \quad & d \xrightarrow{a} d' \\
\implies \quad & \exists e'.\, e \xrightarrow{a} e' \,\&\, d' \lesssim_k e' \\
\implies \quad & \exists e'.\, <a, e'> \in e \,\&\, \pi_k d' \sqsubseteq \pi_k e' \ \text{ by induction hypothesis} \\
\implies \quad & \exists <a, \pi_k e'> \in Y.\, <a, \pi_k d'> \sqsubseteq <a, \pi_k e'>.
\end{aligned}$$

Again,

$$\begin{aligned}
\bullet \quad & \bot \notin X \\
\implies \quad & \bot \notin d \\
\implies \quad & \bot \notin e \,\&\, [e \xrightarrow{a} e' \Rightarrow \exists d'.\, d \xrightarrow{a} d' \,\&\, d' \lesssim_k e'] \\
\implies \quad & \bot \notin Y \,\&\, \forall <a, \pi_k e'> \in Y.\, \exists <a, \pi_k d'> \in X.\, \pi_k d' \sqsubseteq \pi_k e'
\end{aligned}$$

by the induction hypothesis again, and we have shown $X \sqsubseteq_{EM} Y$, which implies $X^\star \sqsubseteq_{EM} Y^\star$, as required.
(2). It suffices to show that $\sqsubseteq$ is a prebisimulation. This is a simple calculation:

$$\begin{array}{ll} \bullet & d \sqsubseteq e \\ \Longrightarrow & \forall{<}a, d'{>} \in d.\, \exists{<}a, e'{>} \in e.\, d' \sqsubseteq e' \\ & \& \perp \notin d \;\Rightarrow\; \perp \notin e \;\&\; [\forall{<}a, e'{>} \in e.\, \exists{<}a, d'{>} \in d.\, d' \sqsubseteq e'] \\ \Longrightarrow & \forall a \in \mathsf{Act}.\, d \xrightarrow{a} d' \;\Rightarrow\; \exists e'.\, e \xrightarrow{a} e' \;\&\; d' \sqsubseteq e' \\ & \& \; d{\downarrow} \;\Rightarrow\; e{\downarrow} \;\&\; [e \xrightarrow{a} e' \;\Rightarrow\; \exists d'.\, d \xrightarrow{a} d' \;\&\; d' \sqsubseteq e']. \quad \blacksquare \end{array}$$

We finish with some examples to illustrate the richness of $\mathcal{D}$ as a transition system.

Examples

(1). $\mathcal{D}$ is not sort-finite.

$$\begin{array}{rcl} d_0 & \equiv & \{\!|{<}a_0, \{\!|\perp|\!\}{>}|\!\} \\ d_1 & \equiv & \{\!|{<}a_0, \{\!|{<}a_1, \{\!|\perp|\!\}{>}|\!\}{>}|\!\} \\ & \vdots & \\ \mathsf{sort}(\bigsqcup d_k) & = & \{a_0, a_1, \ldots\} \end{array}$$

(2). $\mathcal{D}$ is not weakly image-finite.

$$\begin{array}{rcl} c_k & \equiv & \sum_{i \le k} a^i \mathbb{O} + a^k \Omega \quad (k \in \omega) \\ \bigsqcup c_k & = & \sum_{k \in \omega} a^k \mathbb{O} + a^\omega. \end{array}$$

5.4 A Domain Logic for Transition Systems

We now introduce our domain logic in an infintary version $\mathcal{L}_\infty$, with a finitary subset $\mathcal{L}_\omega$. We show how $\mathcal{L}_\infty$ can be interpreted in any transition system, present a proof system, and establish its soundness. We then turn to $\mathcal{L}_\omega$, and prove the main result of the section: $\mathcal{L}_\omega$ is the Stone dual of $\mathcal{D}$. That is, $\mathcal{D}$ is isomorphic to the spectral space of $\mathcal{L}_\omega$, while $\mathcal{L}_\omega$ is isomorphic to the lattice of compact-open subsets of $\mathcal{D}$. This duality will be crucial to our work in the next section.

Definition 5.4.1 The language $\mathcal{L}_\infty$ has two *sorts*: π (process) and κ (capability). We write $\mathcal{L}_{\infty\pi}$ ($\mathcal{L}_{\infty\kappa}$) for the class of formulae of sort π (κ), which are defined inductively as follows:

- $\dfrac{\{\phi_i \in \mathcal{L}_{\infty\sigma}\}_{i\in I}}{\bigvee_{i\in I}\phi_i, \bigwedge_{i\in I}\phi_i \in \mathcal{L}_{\infty\sigma}} \quad (\sigma \in \{\pi, \kappa\})$
- $\dfrac{a \in \mathsf{Act}, \quad \phi \in \mathcal{L}_{\infty\pi}}{a(\phi) \in \mathcal{L}_{\infty\kappa}}$
- $\dfrac{\phi \in \mathcal{L}_{\infty\kappa}}{\Box\phi, \Diamond\phi \in \mathcal{L}_{\infty\pi}}.$

Notation. We write $t \equiv \bigwedge_{i\in\varnothing}\phi_i$, $f \equiv \bigvee_{i\in\varnothing}\phi_i$.

The sublanguage of $\mathcal{L}_\infty$ obtained by the restriction to *finite* conjunctions and disjunctions is denoted $\mathcal{L}_\omega$. *Height*, *modal depth* and *sort* are defined for $\mathcal{L}$ in entirely analogous fashion to HML. For example:

- $\mathsf{md}(\bigwedge_{i\in I}\phi_i) \equiv \mathsf{md}(\bigwedge_{i\in I}\phi_i) \equiv \sup\{\mathsf{md}(\phi_i : i \in I\}$
- $\mathsf{md}(a(\phi)) \equiv \mathsf{md}(\phi)$
- $\mathsf{md}(\Box\phi) \equiv \mathsf{md}(\Diamond\phi) \equiv \mathsf{md}(\phi) + 1.$

For each $A \subseteq \mathsf{Act}$ and ordinal λ:

$$\mathcal{L}_\infty^{(A,\lambda)} \equiv \{\phi \in \mathcal{L}_\infty : \mathsf{sort}(\phi) \subseteq A \;\&\; \mathsf{md}(\phi) \leq \lambda\}.$$

It should be clear how the form of our language is derived from the type expression

$$\mathsf{rec}\, t.\, P^0[\sum_{a\in\mathsf{Act}} t].$$

The two-sorted structure of $\mathcal{L}$ corresponds to the type constructions P^0 (π) and $\sum_{a \in \mathsf{Act}}$ (κ). The recursion in the type expression is mirrored by the mutual recursion between the two sorts. Note that the Plotkin powerdomain is built from the combination of the *must* modality $\Box$ of the Smyth powerdomain and the *may* modality $\Diamond$ of the Hoare powerdomain (*cf.* [Abr83a, Win83]).

Interpretation of $\mathcal{L}$ in transition systems

Given a transition system $(\mathrm{Proc}, \mathsf{Act}, \rightarrow, \uparrow)$, we define

$$\mathrm{Cap} \equiv \{\bot\} \cup (\mathsf{Act} \times \mathrm{Proc})$$

$$C : \mathrm{Proc} \rightarrow \wp(\mathrm{Cap})$$

$$C(p) = \{\bot : p\uparrow\} \cup \{<a, q> : p \xrightarrow{a} q\}.$$

$C(p)$ is the set of *capabilities* of p. We can now define satisfaction relations

$$\models_\pi \subseteq \mathrm{Proc} \times \mathcal{L}_{\infty\pi},$$

$$\models_\kappa \subseteq \mathrm{Proc} \times \mathcal{L}_{\infty\kappa} :$$

For $\sigma \in \{\pi, \kappa\}$:

$$\begin{array}{lcl} w \models_\sigma \bigwedge_{i\in I} \phi_i & \equiv & \forall i \in I.\, w \models_\sigma \phi_i \\ w \models_\sigma \bigvee_{i\in I} \phi_i & \equiv & \exists i \in I.\, w \models_\sigma \phi_i \\ p \models_\pi \Box\phi & \equiv & \forall c \in C(p).\, c \models_\kappa \phi \\ p \models_\pi \Diamond\phi & \equiv & \exists c \in C(p) \cup \{\bot\}.\, c \models_\kappa \phi \\ c \models_\kappa a(\phi) & \equiv & c = <a, q> \;\&\; q \models_\pi \phi. \end{array}$$

The *assertions* over $\mathcal{L}$ have the form

$$\phi \leq_\sigma \psi, \;\; \phi =_\sigma \psi \quad (\sigma \in \{\pi, \kappa\}, \phi, \psi \in \mathcal{L}_{\infty\sigma}).$$

The satisfaction relation between transition systems and assertions is defined by:

$$\begin{array}{lcl} \mathcal{T} \models \phi \leq_\sigma \psi & \equiv & \forall w \in S_\sigma.\, w \models_\sigma \phi \implies w \models_\sigma \psi \\ \mathcal{T} \models \phi =_\sigma \psi & \equiv & \forall w \in S_\sigma.\, w \models_\sigma \phi \iff w \models_\sigma \psi. \end{array}$$

$$(\sigma \in \{\pi, \kappa\}, S_\pi = \mathrm{Proc}, S_\kappa = \mathrm{Cap}).$$

This is extended to a class of transition systems $\mathbf{C}$ by:

$$\mathbf{C} \models A \quad \equiv \quad \forall \mathcal{T} \in \mathbf{C}.\, \mathcal{T} \models A.$$

If $\mathbf{C}$ is the class of all transition systems, we simply write $\models A$.

A Proof System For $\mathcal{L}_\infty$

Firstly, we define a predicate $(\cdot)\!\downarrow$ on $\mathcal{L}_\infty$:

$$\begin{array}{lcl}
(\bigwedge_{i \in I} \phi_i)\!\downarrow & \equiv & \exists i \in I.\, \phi_i\!\downarrow \\
(\bigwedge_{i \in I} \phi_i)\!\downarrow & \equiv & \forall i \in I.\, \phi_i\!\downarrow \\
a(\phi)\!\downarrow & \equiv & \mathsf{true} \\
(\Box\phi)\!\downarrow & \equiv & \phi\!\downarrow \\
(\Diamond\phi)\!\downarrow & \equiv & \phi\!\downarrow.
\end{array}$$

Intuitively, $\phi\!\downarrow$ means that at least the completely undefined process does *not* satisfy ϕ (i.e. $\phi \neq t$). We will use it to restrict one of our axiom schemes.

We now present a proof system for assertions over $\mathcal{L}_\infty$. Sort subscripts are omitted.

Logical Axioms

Exactly as in Chapter 4, except that the restriction to finite index sets on conjunctions and disjunctions is lifted.

Modal Axioms

$$(a - \leq) \quad \frac{\phi \leq \psi}{a(\phi) \leq a(\psi)}$$

$$(a - \wedge)(i) \quad a(\bigwedge_{i \in I} \phi_i) = \bigwedge_{i \in I} a(\phi_i) \quad (I \neq \varnothing)$$

$$(a - \wedge)(ii) \quad a(\phi) \wedge b(\psi) = f \quad (a \neq b)$$

$$(a - \vee) \quad a(\bigvee_{i \in I} \phi_i) = \bigvee_{i \in I} a(\phi_i)$$

$$(\Box - \leq) \quad \frac{\phi \leq \psi}{\Box\phi \leq \Box\psi}$$

$$(\Box - \wedge) \quad \Box \bigwedge_{i \in I} \phi_i = \bigwedge_{i \in I} \Box\phi_i$$

$$(\Diamond - \leq) \quad \frac{\phi \leq \psi}{\Diamond\phi \leq \Diamond\psi}$$

$$(\Diamond - \vee) \quad \Diamond \bigvee_{i \in I} \phi_i = \bigvee_{i \in I} \Diamond\phi_i$$

$$(\Box - \vee) \quad \Box(\phi \vee \psi) \leq \Box\phi \vee \Diamond\psi$$

$$(\Diamond - \wedge) \quad \Box\phi \wedge \Diamond\psi \leq \Diamond(\phi \wedge \psi) \quad (\psi\downarrow)$$

$$(\Diamond - t) \quad \Diamond t \ = t.$$

The form of our axiomatisation follows the same pattern as that of Chapter 4, of (the general approach exemplified by) which it is of course a special case. The first group of axioms and rules give the logical structure of entailment, conjunction and disjunction. They give (the Lindenbaum algebra of) $\mathcal{L}_\infty$ the structure of a (large) *completely distributive lattice* [Joh82]. We then articulate the modal structure by showing how the constructors interact with the logical structure. The axioms for the $a(\cdot)$ constructor correspond to those for coalesced sum given in Chapter 4; the fact that *separated* sum is intended here is reflected by the side-condition on $(a - \wedge)(i)$. The axioms for $\Box$ and $\Diamond$ individually correspond to those presented for the upper and lower powerdomains in Chapter 4; however, these two modalities interact in the Plotkin powerdomain, resulting in its greater complexity; these interactions are expressed in logical terms by $(\Box - \vee)$ and $(\Diamond - \wedge)$. Our surgery on the ordering to keep a least element while adding the empty set is reflected by the presence of $(\Diamond - t)$ and the side condition on $(\Diamond - \wedge)$.

We write $\mathcal{L} \vdash A$ or just $\vdash A$ if an assertion A is derivable from the above rules and axioms. It will be convenient to have equational versions of $(\Box - \vee)$ and $(\Diamond - \wedge)$, which can be obtained as theorems of $\mathcal{L}$:

$$(D1) \quad \vdash \Box(\phi \vee \psi) = \Box\phi \vee (\Box(\phi \vee \psi) \wedge \Diamond\psi)$$

$$(D2) \quad \vdash \Box\phi \wedge \Diamond\psi = \Box\phi \wedge \Diamond(\phi \wedge \psi) \qquad (\psi\downarrow).$$

We now turn to the question of soundness for our system. As a first step, we show that our auxiliary predicate $()\downarrow$ works as intended.

Proposition 5.4.2 *(i)* $\forall \phi \in \mathcal{L}_{\infty\kappa}.\, \phi\downarrow \iff \bot \not\models_\kappa \phi$.
(ii) $\forall \phi \in \mathcal{L}_{\infty\pi}.\, \phi\downarrow \iff p \models_\pi \phi \Rightarrow C(p) \neq \{\bot\}$.

Proof. We prove (i) and (ii) simultaneously by induction on ϕ. We consider the two non-trivial cases:
$\Box\phi$: Assume $(\Box\phi)\downarrow \equiv \phi\downarrow$, and $p \models_\pi \Box\phi$. $C(p) = \{\bot\}$ would then imply $\bot \models_\kappa \phi$, but this is impossible by the induction hypothesis. For the converse, suppose $(\Box\phi)\uparrow$, i.e. $\phi\uparrow$. Then by induction hypothesis, $\bot \models_\kappa \phi$, and hence $\Omega \models_\pi \Box\phi$ with $C(\Omega) = \{\bot\}$.
$\Diamond\phi$: Assume $\phi\downarrow$ and $p \models_\pi \Diamond\phi$. Then $\bot \not\models_\kappa \phi$, and so there must be $c \in C(p) - \{\bot\}$ with $c \models_\kappa \phi$. The converse is proved by the same argument as for $\Box\phi$. ∎

Theorem 5.4.3 (Soundness of $\mathcal{L}$) $\vdash A \implies \models A$.

Proof. By a routine induction over proofs. For illustration, we consider $(\Diamond - \wedge)$. Assume $\psi\downarrow$ and $p \models_\pi \Box\phi \wedge \Diamond\psi$. Then $p \models_\pi \Diamond\psi$, and so by 5.4.2, $C(p) \neq \{\bot\}$ and $\bot \not\models_\kappa \psi$, and there must be $c \in C(p) - \{\bot\}$ such that $c \models_\kappa \psi$. But then $p \models_\pi \Box\phi$ implies that $c \models_\kappa \phi$, and so $p \models_\pi \Diamond(\phi \wedge \psi)$ as required. ∎

We now turn to the finitary logic $\mathcal{L}_\omega$. Henceforth we assume that Act is countable. It is then clear that $\mathcal{L}_\omega$ can be made into a countable set by a suitable choice of canonical representatives of logical equivalence classes.

Recall that $\mathsf{Spec}\ \mathcal{L}_\omega$ is the set of *prime filters* over $\mathcal{L}_{\omega\pi}$, i.e. subsets $x \subseteq \mathcal{L}_{\omega\pi}$ satisfying

- $\phi \in x \;\&\; \vdash \phi \leq \psi \;\; \Rightarrow \;\; \psi \in x$
- $t \in x$
- $\phi, \psi \in x \;\; \Rightarrow \;\; \phi \wedge \psi \in x$
- $f \notin x$
- $\phi \vee \psi \in x \;\; \Rightarrow \;\; \phi \in x$ or $\psi \in x$.

$\mathsf{Spec}\ \mathcal{L}_\omega$ is topologised by taking as basic opens

$$U_\phi \;\equiv\; \{x \in \mathsf{Spec}\ \mathcal{L}_\omega \;: \phi \in x\} \quad (\phi \in \mathcal{L}_{\omega\pi}),$$

or, equivalently in our context, by taking the Scott topology over the specialisation order on $\mathsf{Spec}\ \mathcal{L}_\omega$, which is simply set inclusion.

Our aim is to prove the following fundamental result, which ahows that the logic $\mathcal{L}_\omega$ does indeed correspond exactly to the domain $\mathcal{D}$:

Theorem 5.4.4 (Stone Duality) *$\mathcal{D}$ and $\mathcal{L}_\omega$ are Stone duals, i.e.*

(*i*) $\mathcal{D} \;\cong\; \mathsf{Spec}\ \mathcal{L}_\omega$

(*ii*) $K\Omega(\mathcal{D}\,) \;\cong\; (\mathcal{L}_{\omega\pi}/{=_\pi}, \leq_\pi/{=_\pi})$.

Here $K\Omega(D)$ is the lattice of compact-open subsets of $\mathcal{D}$, while

$$(\mathcal{L}_{\omega\pi}/{=_\pi}, \leq_\pi/{=_\pi})$$

is the *Lindebaum algebra* of $\mathcal{L}_\omega$. Since $\mathcal{D}$ is coherent, (i) and (ii) are indeed equivalent ([Joh82]).

The Stone Duality Theorem is entirely analogous to Theorem 4.2.5, and our proof strategy is identical. However, some of the technical details are more complex; in particular, the syntactic identification of primes is less obvious than for Scott domains, since primes are no longer preserved under meets.

We begin by defining a normal form for $\mathcal{L}_\omega$.

Definition 5.4.5 (i) ϕ is in *strong disjunctive normal form* (SDNF) if it has the form $\bigvee_{i \in I} \phi_i$, where each ϕ_i is in *prime normal form* (PNF).
(ii) ϕ is in PNF if it has one of the forms

- $\bigwedge_{i \in I} \Diamond a_i(\phi_i)$, where each ϕ_i is in PNF.
- $\Box \bigvee_{i \in I} a_i(\phi_i) \ \wedge \ \bigwedge_{j \in J} \Diamond b_j(\psi_j)$, where
    1. Each ϕ_i and ψ_j is in PNF.
    2. $\forall i \in I. \exists j \in J. \vdash b_j(\psi_j) \leq a_i(\phi_i)$.
    3. $\forall j \in J. \exists i \in I. \vdash b_j(\psi_j) \leq a_i(\phi_i)$.

We call (2) and (3) the *convexity conditions* (note the resemblance to the Egli–Milner ordering).

The combinatorics are concentrated in the following

Theorem 5.4.6 (SDNF) *For every $\phi \in \mathcal{L}_{\omega\pi}$, there is (effectively) a ψ in SDNF such that*

$$\vdash \phi =_\pi \psi.$$

PROOF. By induction on $\mathsf{md}(\phi)$. The idea is to form a sequence of "transformations"

$$\phi \equiv \phi_0 \rightsquigarrow \phi_1 \rightsquigarrow \cdots \rightsquigarrow \phi_n$$

such that

(1) $\vdash \phi_i = \phi_{i+1} \quad (0 \leq i < n)$

(2) $\mathsf{md}(\phi_{i+1}) \leq \mathsf{md}(\phi_i) \quad (0 \leq i < n)$

(3) ϕ_n is in SDNF.

(Condition (2) is needed to keep the induction going.) To keep the notation bearable, we shall omit indices in conjunctions and disjunctions, writing e.g. $\bigvee\{\phi\}$.

Firstly, using the distributive lattice laws we can transform ϕ_0 into

$$\bigvee\{\bigwedge\{\Box \bigwedge\{\bigvee\{a(\phi)\}\}\} \ \wedge \ \bigwedge\{\Diamond \bigwedge\{\bigvee\{b(\psi)\}\}\}\} \tag{5.6}$$

Using $(\Box - \wedge)$ in the outwards direction for each $\Box$-conjunct in 5.6, and the distributive law and then $(\Diamond - \vee)$, followed by the distributive law again, in each $\Diamond$-conjunct, we otain

$$\bigvee\{\bigwedge\{\Box \bigvee\{a(\phi)\}\} \ \wedge \ \bigwedge\{\Diamond \bigwedge\{b(\psi)\}\}\} \tag{5.7}$$

Now for each non-empty conjunction

$$\bigwedge\{\Box \bigvee\{a(\phi)\}\}$$

in 5.7, we can use $(\Box - \wedge)$, the distributive law, and $(a - \wedge)$ (i) or (ii); similarly, inside each $\Diamond \bigwedge\{b(\psi)\}$ we can use $(\Diamond - t)$ if the conjunction is empty, and otherwise $(b - \wedge)$ (i) or (ii) (with further applications of $(\Diamond - \vee)$ and the distributive laws as in the previous step if $(b - \wedge)(ii)$ is applicable), to obtain

$$\bigvee\{\theta\} \tag{5.8}$$

where each θ is in one of the forms

$$\bigwedge\{\Diamond b(\psi)\} \tag{5.9}$$

or

$$\Box \bigvee\{a(\phi)\} \ \wedge \ \bigwedge\{\Diamond b(\psi)\} \tag{5.10}$$

Since we have not increased modal depth in obtaining 5.8, we can apply the inductive hypothesis to each ϕ and ψ to obtain $\bigvee\{\phi'\}$, $\bigvee\{\psi'\}$ with each ϕ' and ψ' in PNF. Using $(a - \vee)$, $(\Diamond - \vee)$ and the distributive laws, we can thus obtain a formula of the same form as 5.8, in which each ϕ and ψ in 5.9 and 5.10 is in PNF.

At this point, our formula 5.8 can only fail to be in SDNF because of disjuncts 5.10 which do not satisfy the convexity conditions

- For each $a(\phi)$, for some $b(\psi)$: $\vdash b(\psi) \leq a(\phi)$.
- For each $b(\psi)$, for some $a(\phi)$: $\vdash b(\psi) \leq a(\phi)$.

Our strategy is to remove any failures of these two conditions, using our derived equations $(D1)$ and $(D2)$ respectively. We begin with the first condition. We argue by induction on (m, n) in the lexicographic ordering on $\omega \times \omega$, where:

- m is the maximum number of $a(\phi)$ occurring in one of the disjuncts 5.10 of our formula 5.8 such that there is no $b(\psi)$ with $\vdash b(\psi) \leq a(\phi)$.
- n is the number of disjuncts attaining this maximum.

If $m = 0$, there is nothing to prove. Otherwise, choose such an $a(\phi)$ in one of the maximal disjuncts. We can apply $(D1)$ to

$$\Box \bigvee \{a'(\phi')\} \;\vee\; a(\phi)$$

to obtain

$$\Box \bigvee \{a'(\phi')\} \quad \vee \quad [\Box(\bigvee \{a'(\phi')\} \vee a(\phi)) \wedge \Diamond a(\phi)] \tag{5.11}$$

We can then use the distributive law to obtain a new formula of the form 5.8 to which the inner induction hypothesis can be applied, since the first disjunct in 5.11 has jettisoned $a(\phi)$, while the second disjunct evidently contains a $\Diamond b(\psi)$ such that $\vdash b(\psi) \leq a(\phi)$, namely $a(\phi)$ itself.

The final stage is to remove failures of the second condition. We argue by induction in the same way as for the previous stage. Suppose we are given a $b(\psi)$ in 5.10 with no $a(\phi)$ such that $\vdash b(\psi) \leq a(\phi)$. Firstly, we note that $\psi\uparrow$ implies $\vdash \psi = t$, which is easily proved by induction on ψ. Hence if $\psi\uparrow$, we can use $(\Diamond - t)$ to eliminate the conjunct $\Diamond b(\psi)$. Otherwise, we can use $(D2)$ to obtain

$$\Box \bigvee \{a(\phi)\} \;\wedge\; \Diamond[b(\psi) \wedge \bigvee \{a(\phi)\}] \;\wedge\; \bigwedge \{\Diamond b'(\psi')\} \tag{5.12}$$

Now we can use the distributive law inside the second main conjunct in 5.12, followed by $(a - \wedge)$, $(\Diamond - \vee)$, and the distributive law again. In this way, the disjunct 5.12 of our main formula is replaced by the disjunction of all those formulae

$$\Box \bigvee \{a(\phi)\} \;\wedge\; \Diamond b(\phi' \wedge \psi) \;\wedge\; \bigwedge \{\Diamond b'(\psi')\} \tag{5.13}$$

for $a'(\phi') \in \{a(\phi)\}$ with $a' = b$. For each such $\phi' \wedge \psi$, we can apply the outer induction hypothesis to obtain $\bigvee\{\theta'\}$ with each θ' in PNF. Applying $(b - \vee)$, $(\Diamond - \vee)$ and the distributive laws as before, we obtain disjuncts of the form

$$\Box \bigvee \{a(\phi)\} \;\wedge\; \Diamond b(\theta') \;\wedge\; \bigwedge \{\Diamond b'(\psi')\} \tag{5.14}$$

Since

$$\vdash \theta' \le \bigvee\{\theta'\} = \phi' \wedge \psi \le \phi',$$

we can apply the inner induction hypothesis to 5.14. This completes the process of transforming ϕ into SDNF. ∎

We shall now prove that formulae in PNF denote primes in $K\Omega(\mathcal{D})$.

Proposition 5.4.7 *For all ϕ in PNF there exsists $k(\phi) \in \mathcal{K}(\mathcal{D})$ such that:*

$$\forall d \in \mathcal{D}.\, d \models \phi \iff k(\phi) \sqsubseteq d.$$

Proof. We define $k(\phi)$ (which must clearly be unique) by induction on ϕ:

- $k(\bigwedge_{i \in I} \Diamond a_i(\phi_i)) \equiv \biguplus_{i \in I} \{\!|{<}a_i, k(\phi_i){>}|\!\} \uplus \{\!|\bot|\!\}$

- $k(\Box \bigvee_{i \in I} a_i(\phi_i) \wedge \bigwedge_{j \in J} \Diamond b_j(\psi_j)) \equiv$

  $\biguplus_{i \in I} \{\!|{<}a_i, k(\phi_i){>}|\!\} \uplus \biguplus_{j \in J} \{\!|{<}b_j, k(\psi_j){>}|\!\}.$

We shall prove the proposition by induction on ϕ. Note that in the statement of the proposition, we are viewing $\mathcal{D}$ as a transition system, according to 5.3.11. With our convention of eliding the isomorphisms between $\mathcal{D}$ and $P^0[\sum_{a \in \mathsf{Act}} \mathcal{D}]$, we have: $d = C(d)$, $(d \in \mathcal{D})$.
Case 1: $\phi \equiv \bigwedge_{i \in I} \Diamond a_i(\phi_i)$.

- $\quad d \models \bigwedge_{i \in I} \Diamond a_i(\phi_i)$

$\iff \forall i \in I.\, \exists {<}a_i, d_i{>} \in d.\, d_i \models \phi_i$

$\iff \forall i \in I.\, \exists {<}a_i, d_i{>} \in d.\, k(\phi_i) \sqsubseteq d_i$ by induction hypothesis

$\iff k(\phi) \sqsubseteq d.$

Case 2: $\phi \equiv \Box \bigvee_{i\in I} a_i(\phi_i) \wedge \bigwedge_{j\in J} \Diamond b_j(\psi_j)$. Let $\Phi = \{a_i(\phi_i) : i \in I\} \cup \{b_j(\psi_j) : j \in J\}$.

$$\begin{aligned}
\bullet \quad & d \models \phi \\
\iff \quad & \forall \langle a, d' \rangle \in d.\, \exists i \in I.\, a = a_i \ \&\ d' \models \phi_i \\
& \&\ \bot \notin d \ \&\ \forall j \in J.\, \exists \langle b_j, d_j \rangle \in d.\, d_j \models \psi_j \\
\iff \quad & \forall \langle a, d' \rangle \in d.\, \exists a(\theta) \in \Phi.\, d' \models \theta \\
& \&\ \bot \notin d \ \&\ \forall a(\theta) \in \Phi.\, \exists \langle a, d' \rangle \in d.\, d \models \theta \\
& \text{by the convexity conditions and the Soundness Theorem,} \\
\iff \quad & k(\phi) \sqsubseteq d, \text{ by induction hypothesis. } \blacksquare
\end{aligned}$$

Theorem 5.4.8 (Prime Completeness) *For all ϕ, ϕ' in PNF:*

$$\mathcal{D} \models \phi \leq \phi' \implies \mathcal{L} \vdash \phi \leq \phi'.$$

PROOF. By 4.7,

$$\mathcal{D} \models \phi \leq \phi' \iff k(\phi') \sqsubseteq k(\phi).$$

Suppose then that $k(\phi') \sqsubseteq k(\phi)$. We argue by induction on ϕ. There are a number of cases, according to the forms of ϕ and ϕ'. We consider the case

$$\begin{aligned}
\phi \ &\equiv\ \Box \bigvee_{i\in I} a_i(\phi_i) \ \wedge\ \bigwedge_{j\in J} \Diamond b_j(\psi_j), \\
\phi' \ &\equiv\ \Box \bigvee_{i'\in I'} a_{i'}(\phi_{i'}) \ \wedge\ \bigwedge_{j'\in J'} \Diamond b_{j'}(\psi_{j'}).
\end{aligned}$$

$$\begin{aligned}
\bullet \quad & k(\phi') \sqsubseteq k(\phi) \\
\iff \quad & \forall j' \in J'.\, \exists j \in J.\, b_j = b_{j'} \ \&\ k(\psi_{j'}) \sqsubseteq k(\psi_j) \\
& \&\ \forall i \in I.\, \exists i' \in I'.\, a_i = a_{i'} \ \&\ k(\phi_{i'}) \sqsubseteq k(\phi_i), \\
& \text{by the convexity conditions, Soundness, and 5.4.7} \\
\implies \quad & \forall j' \in J'.\, \exists j \in J.\ \vdash b_j(\psi_j) \leq b_{j'}(\psi_{j'}) \\
& \&\ \forall i \in I.\, \exists i' \in I'.\ \vdash\ a_i(\phi_i) \leq a_{i'}(\phi_{i'}), \\
& \text{by the induction hypothesis,} \\
\implies \quad & \vdash\ \phi \leq \phi'. \ \blacksquare
\end{aligned}$$

We can now use the same arguments as in Chapter 3 T7 to prove

Theorem 5.4.9 (Completeness) *For all* $\phi, \psi \in \mathcal{L}_\omega$*:*

$$\mathcal{D} \models \phi \leq \psi \implies \mathcal{L}_\omega \vdash \phi \leq \psi.$$

We now establish a converse to 5.4.7.

Theorem 5.4.10 (Definability) *For all* $d \in \mathcal{K}(\mathcal{D})$*, for some* ϕ *in PNF,* $k(\phi) = d$*.*

PROOF. We define $\phi(d)$ by induction on the construction of d according to 5.3.9:

$$\begin{aligned} \phi(\biguplus_{i \in I} \{\!|<a_i, d_i>|\!\} \uplus \{\!|\bot|\!\}) &\equiv \bigwedge_{i \in I} \Diamond a_i(\phi(d_i)) \\ \phi(\biguplus_{i \in I} \{\!|<a_i, d_i>|\!\}) &\equiv \Box \bigvee_{i \in I} a_i(\phi(d_i)) \ \wedge \ \bigwedge_{i \in I} \Diamond a_i(\phi(d_i)). \end{aligned}$$

Note in particular that $\phi(\emptyset) = \Box f$. It is easily verified that $\phi(d)$ is in PNF and that $k(\phi(d)) = d$. ∎

The Duality Theorem is an immediate consequence of Soundness, Completeness and Definability, just as in Chapter 3 T8.

Combining Soundness and Completeness we obtain

Theorem 5.4.11 (Completeness for $\mathcal{L}_\omega$) *Let* **C** *be any class of transition systems containing* $\mathcal{D}$*. Then for* $\phi, \psi \in \mathcal{L}_\omega$*:*

$$\mathbf{C} \models \phi \leq \psi \iff \mathcal{D} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

5.5 Applications of the Domain Logic

We shall now use domain logic to study bisimulation. Our results in this section can be grouped under four main headings:

1. Comparisons with Hennessy-Milner logic
2. Characterisation Theorems
3. Finitary Transition Systems
4. Universal Semantics

Of these, (1) and (2) will confirm the appropriateness of our definitions, while (3) and (4) will represent a distinctive payoff for our approach.

Comparison with Hennessy-Milner logic

We begin with some technicalities on normal forms.

Definition 5.5.1 We define a class of normal forms $\mathsf{N}\mathcal{L}_\infty \subseteq \mathcal{L}_{\infty\pi}$ inductively as follows:

- $\dfrac{\{\phi_i \in \mathsf{N}\mathcal{L}_\infty\}_{i\in I}}{\bigwedge_{i\in I}\phi_i, \bigvee_{i\in I}\phi_i \in \mathsf{N}\mathcal{L}_\infty}$
- $\dfrac{\phi \in \mathsf{N}\mathcal{L}_\infty\ ,\quad a \in \mathsf{Act}}{\Diamond a(\phi) \in \mathsf{N}\mathcal{L}_\infty}$
- $\dfrac{\{\phi_i \in \mathsf{N}\mathcal{L}_\infty\}_{i\in I},\quad \{a_i \in \mathsf{Act}\}_{i\in I}\quad \{i \neq j \ \Rightarrow\ a_i \neq a_j\}_{i,j\in I}}{\Box \bigvee_{i\in I} a_i(\phi_i) \in \mathsf{N}\mathcal{L}_\infty}$

Lemma 5.5.2 (Normal Forms) *For all $\phi \in \mathcal{L}_{\infty\pi}$, for some $\psi \in \mathsf{N}\mathcal{L}_\infty$:*

$$\mathcal{L}_\infty \vdash \phi = \psi.$$

Proof. By induction on $\mathsf{md}(\phi)$. We consider the two non-trivial cases.
$\Diamond\phi$: In this case, using the distributive lattice laws there is ϕ' of the form

$$\bigvee_{i\in I}\bigwedge_{j\in J_i} a_{ij}(\phi_{ij})$$

such that $\vdash \phi = \phi'$, and $\mathsf{md}(\phi') \leq \mathsf{md}(\phi)$. By the induction hypothesis, for each ϕ_{ij} there is $\phi'_{ij} \in \mathsf{N}\mathcal{L}_\infty$ such that $\vdash \phi_{ij} = \phi'_{ij}$. Using $(a-\leq)$ and $(\Diamond-\leq)$, we have

$$\vdash \Diamond\phi = \Diamond \bigvee_{i\in I} \bigwedge_{j\in J_i} a_{ij}(\phi_{ij}). \tag{5.15}$$

Now for each $i \in I$, there are three cases:

1. $J_i = \varnothing$. In this case, $\vdash \Diamond\phi = \Diamond t$, and we can use $(\Diamond - t\ )$ to obtain a normal form.

2. $\exists j_1, j_2 \in J_i$, $a_{j_1} \neq a_{j_2}$. In this case, we can use $(a - \wedge)$ to delete the i'th disjunct in the RHS of 5.15.

3. $\{a_{ij} : j \in J_i\} = \{a\}$, for some $a \in \mathsf{Act}$. In this case, we can use $(a - \wedge)(i)$.

In this way, we obtain *either*

$$\vdash \Diamond\phi = t,$$

if case (1) is ever applicable, *or*

$$\vdash \Diamond\phi = \Diamond \bigvee_{i'\in I'} a_{i'}(\psi_{i'}) \quad (\psi_{i'} \in \mathsf{N}\mathcal{L}_\infty).$$

In the latter case, we can apply $(\Diamond - \vee)$ to get a normal form.
$\Box\phi$: Similarly to the previous case, we have

$$\vdash \Box\phi = \Box \bigwedge_{i\in I} \bigvee_{j\in J_i} a_{ij}(\phi_{ij}) \quad (\phi_{ij} \in \mathsf{N}\mathcal{L}_\infty).$$

We can then use $(\Box - \wedge)$ to get

$$\vdash \Box\phi = \bigwedge_{i\in I} \Box \bigvee_{j\in J_i} a_{ij}(\phi_{ij}).$$

Now if we partition each J_i by $\sim_i$, with

$$j \sim_i k \iff a_{ij} = a_{ik} \quad (j, k \in J_i),$$

we have

$$\vdash \Box\phi = \bigwedge_{i\in I} \Box \bigvee_{[j]\in J_i/\sim_i} (\bigvee_{k\in[j]} a_{ij}(\phi_{ik}))$$

using the lattice laws; we can then apply $(a - \vee)$ to get a normal form. ∎

Definition 5.5.3 We define translation functions

$$(\cdot)^* : \mathrm{HML}_\infty \longrightarrow \mathsf{N}\mathcal{L}_\infty\ ,$$

$$(\cdot)^\dagger : \mathsf{N}\mathcal{L}_\infty \longrightarrow \mathrm{HML}_\infty.$$

$$\begin{array}{lcl}
(\bigwedge_{i\in I}\phi_i)^* & = & \bigwedge_{i\in I}(\phi_i)^* \\
(\bigvee_{i\in I}\phi_i)^* & = & \bigvee_{i\in I}(\phi_i)^* \\
(<\!a\!>\phi)^* & = & \Diamond a(\phi^*) \\
([a]\phi)^* & = & \Box a((\phi)^*) \vee \bigvee\{b(t) : b \in \mathsf{Act} - \{a\}\}) \\
(\bigwedge_{i\in I}\phi_i)^\dagger & = & \bigwedge_{i\in I}(\phi_i)^\dagger \\
(\bigvee_{i\in I}\phi_i)^\dagger & = & \bigvee_{i\in I}(\phi_i)^\dagger \\
(\Diamond a(\phi))^\dagger & = & <\!a\!>(\phi)^\dagger \\
(\Box \bigvee_{i\in I} a_i(\phi_i))^\dagger & = & \bigwedge_{i\in I}[a_i](\phi_i)^\dagger\ \wedge\ \bigwedge\{[b]f : b \in \mathsf{Act} - \{a_i : i \in I\}\}
\end{array}$$

The following is easily verified.

Proposition 5.5.4 *For all $\phi \in \mathrm{HML}_\infty, \psi \in \mathsf{N}\mathcal{L}_\infty$:*

$$\begin{array}{rlcl}
(i) & \mathsf{md}(\phi) & = & \mathsf{md}(\phi^*) \\
(ii) & \mathsf{md}(\psi) & = & \mathsf{md}(\psi^\dagger) \\
(iii) & p \models \phi & \iff & p \models \phi^* \\
(iv) & p \models \psi & \iff & p \models \psi^\dagger.
\end{array}$$

As an immediate consequence of this Proposition together with 5.5.2, we have

Theorem 5.5.5 (Comparison Theorem (Infinitary Case)) *For $p, q \in$ Proc in any transition system, $A \subseteq \mathsf{Act}$ and $\lambda \in \mathsf{Ord}$:*

$$\mathcal{L}_\infty^{(A,\lambda)}(p) \subseteq \mathcal{L}_\infty^{(A,\lambda)}(q) \iff \mathrm{HML}_\infty^{(A,\lambda)}(p) \subseteq \mathrm{HML}_\infty^{(A,\lambda)}(q).$$

Thus in the infinitary case, $\mathcal{L}_\infty$ determines the same preorder on processes as HML_∞. However, when Act is infinite this does *not* cut down to a corresponding result for the finitary case, since our translation functions introduce infinite disjunctions in translating $[a]$, and infinite conjunctions in translating $\Box$, even for finite formulas. Our general considerations on observability in Chapter 2 suggest that the introduction of infinite conjunctions is more serious, and indicates a weakness of expressive power in HML_∞ as an "observational logic". This is in keeping with our remarks at the end of Section 2. In fact, our translation functions suggest an appropriate way of extending HML_∞ so as to render it equivalent to $\mathcal{L}_\omega$. This will be the content of a second Comparison Theorem which we will prove later in this section, when we have some additional machinery at our disposal.

Characterisation Theorems

Combining the Comparison Theorem with the Modal Characterisation Theorem 5.2.4, we have:

Theorem 5.5.6 (Characterisation Theorem for $\mathcal{L}_\infty$) *With notation as in the previous Theorem,*

$$p \lesssim_\lambda q \iff \mathcal{L}_\infty^{(\mathsf{Act},\lambda)}(p) \subseteq \mathcal{L}_\infty^{(\mathsf{Act},\lambda)}(q)$$

and therefore

$$p \lesssim^B q \iff \mathcal{L}_\infty(p) \subseteq \mathcal{L}_\infty(q).$$

We now turn to the question of finding a Characterisation Theorem for $\mathcal{L}_\omega$. Intuitively, $\mathcal{L}_\omega$ represents finitely observable properties of processes, hence should correspond to the "finitely observable part" of bisimulation. If we accept the finite synchronisation trees ST_ω as a suitable notion of *finite process*, we can use them to determine the *algebraic* part of the bisimulation preorder, in the sense e.g. of [Gue81].

Definition 5.5.7 The *finitary preorder* $\lesssim^F$ is defined on any transition system by:

$$p \lesssim^F q \equiv \forall t \in \mathsf{ST}_\omega.\, t \lesssim^B p \Rightarrow t \lesssim^B q.$$

Our aim is to prove

Theorem 5.5.8 (Characterisation Theorem for $\mathcal{L}_\omega$) *With notation as in the previous Theorem,*

$$p \lesssim^F q \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

We will need a few auxiliary results which also have some independent interest.

Definition 5.5.9 The *height* of a synchronisation tree is defined by:

$$\mathsf{ht}(\sum_{i \in I} a_i t_i \ [+\Omega]) = \sup \{\mathsf{ht}(t_i) : i \in I\} + 1$$

Lemma 5.5.10 *For any synchronisation tree $T \in \mathsf{ST}_\infty$, $\mathsf{ht}(T) < \lambda$ implies*

$$T \lesssim^B p \iff T \lesssim_\lambda p.$$

PROOF. The left-to-right implication is immediate; the converse is an easy induction on $\mathsf{ht}(T)$. ∎

In particular, we see that for a finite synchronisation tree $t \in \mathsf{ST}_\omega$, $t \lesssim^B p \Leftrightarrow t \lesssim_\omega p$. Thus we have the inclusions

$$\lesssim^B \subseteq \lesssim_\omega \subseteq \lesssim^F.$$

In general, these inclusions are strict.

Examples

(1) $\lesssim^B \neq \lesssim_\omega$.

$$p \equiv a^\omega + \Omega, \quad q \equiv \sum_{k \in \omega} a^k \mathbb{O} \ + \Omega$$

Then $p \lesssim_\omega q$, but $p \not\lesssim_{\omega+1} q$.

(2) $\lesssim_\omega \neq \lesssim^F$.

$$p \equiv a(\sum_{n \in \omega} b_n \mathbb{O} + \Omega) + \Omega$$

$$q \equiv \sum_{n \in \omega} a(\sum_{m \in \omega - \{n\}} b_n \mathbb{O} + \Omega) + \Omega$$

Then $p \lesssim^F q$, but $p \not\lesssim_2 q$.

These examples gain in significance because all the processes involved can be defined in finitary calculi, in particular SCCS, as we shall see in the next section.

Lemma 5.5.11 (Sort Lemma) *In any transition system, let* $p, q \in \mathrm{Proc}$*,* $\mathsf{sort}(p) \subseteq A \subseteq \mathsf{Act}$*,* $\lambda \in \mathsf{Ord}$*. Then*

$$p \not\lesssim_\lambda q \implies \mathcal{L}_\infty^{(A,\lambda)}(p) \not\subseteq \mathcal{L}_\infty^{(A,\lambda)}(q).$$

PROOF. By induction on λ. We assume $p \not\lesssim_\lambda q$, and must construct $\phi \in \mathcal{L}_\infty^{(A,\lambda)}(p) - \mathcal{L}_\infty^{(A,\lambda)}(q)$. There are three cases.
(1) $p \xrightarrow{a} p'$ and for all q', $q \xrightarrow{a} q'$ implies $p' \not\lesssim_\alpha q'$ for some $\alpha < \lambda$. By induction hypothesis, for each such q there is $\phi_{q'} \in \mathcal{L}_\infty^{(A,\alpha)}(p') - \mathcal{L}_\infty^{(A,\alpha)}(q')$. Now define

$$\phi \equiv \Diamond a(\bigwedge\{\phi_{q'} : q \xrightarrow{a} q'\}).$$

(2) $p{\downarrow}$ and $q{\uparrow}$. Let $\phi \equiv \Box \bigvee\{a(t) : \exists p'.\, p \xrightarrow{a} p'\}$.
(3) $p{\downarrow}$, $q{\downarrow}$, $q \xrightarrow{a} q'$, and for all p', $p \xrightarrow{a} p'$ implies $p' \not\lesssim_\alpha q'$ for some $\alpha < \lambda$. Define $\phi_{p'}$ similarly to case (1). Then we define

$$\phi \equiv \Box(\bigvee\{a(\phi_{p'}) : p \xrightarrow{a} p'\} \vee \bigvee\{b(t) : b \neq a \,\&\, \exists r.\, p \xrightarrow{a} r\}). \quad \blacksquare$$

Note that this result is stronger than the Modal Characterisation Theorem 5.2.4 for Hennessy-Milner logic, since we only require $\mathsf{sort}(p) \subseteq A$. This is significant in the light of the example at the end of Section 2.

Proposition 5.5.12 *For all* $t \in \mathsf{ST}_\omega$*:*

$$t \lesssim^B p \iff \mathcal{L}_\omega(t) \subseteq \mathcal{L}_\omega(p).$$

PROOF. Combining 5.5.10 and 5.5.11, we see that

$$t \lesssim^B p \iff \mathcal{L}_\infty^{(A,k)}(t) \subseteq \mathcal{L}_\infty^{(A,k)}(p),$$

where $A = \mathsf{sort}(t)$ and $k = \mathsf{ht}(t)$. Since A and k are both finite, $\mathcal{L}_\infty^{(A,k)}$ is finite up to logical equivalence (i.e. the Lindenbaum algenbra is finite). Thus each formula in $\mathcal{L}_\infty^{(A,k)}$ is equivalent to one in $\mathcal{L}_\omega$, and the proposition is proved. $\blacksquare$

We need one more auxiliary result, which will in fact be a consequence of our work on SCCS in the next section. Firstly, we define a map from prime normal forms to finite synchronisation trees

$$\mathsf{st} : \mathrm{PNF} \to \mathsf{ST}_\omega$$

as follows:

$$\mathsf{st}(\bigwedge_{i \in I} \Diamond a_i(\phi_i)) \equiv \sum_{i \in I} a_i \mathsf{st}(\phi_i) + \Omega$$

$$\mathsf{st}(\Box \bigvee_{i \in I} a_i(\phi_i) \wedge \bigwedge_{j \in J} \Diamond b_j(\psi_i)) \equiv \sum_{i \in I} a_i \mathsf{st}(\phi_i) + \sum_{j \in J} b_j \mathsf{st}(\psi_j).$$

Now analogously to 5.4.7 we have

Proposition 5.5.13 *For all ϕ in PNF, and $p \in$ Proc in any transition system:*

$$p \models \phi \iff \mathsf{st}(\phi) \lesssim^B p.$$

The proof is entirely analogous to 5.4.7.

We can now prove 5.5.8. Firstly, $\mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q)$ implies $p \lesssim^F q$, by 5.5.12. For the converse, assume $p \lesssim^F q$ and $p \models \phi$, ($\phi \in \mathcal{L}_\omega$). By the SDNF Theorem 5.4.6,

$$\begin{array}{lll} \bullet & \vdash \phi = \bigvee_{i \in I} \phi_i & (\phi_i \in \mathrm{PNF}) \\ \Longrightarrow & \exists i \in I.\, p \models \phi_i & \\ \Longrightarrow & \mathsf{st}(\phi_i) \lesssim^B p & 5.5.13 \\ \Longrightarrow & \mathsf{st}(\phi_i) \lesssim^B q & p \lesssim^F q \\ \Longrightarrow & q \models \phi_i & 5.5.13 \\ \Longrightarrow & q \models \phi. & \blacksquare \end{array}$$

Finitary Transition Systems

We now embark on our next topic. The various finiteness conditions on transition systems defined in section 2 reflect attempts to capture features of finitary processes. Nowever, none of these conditions seems to capture exactly the right class of systems unless we make some unwelcome assumptions such as that the set of actions is finite. We shall adopt what seems to be a novel approach, of using our program logic to axiomatize a class of systems which we propose as the finitary ones. Our axiomatisation consists of two schemes over $\mathcal{L}_\infty$.

Notation. $\mathsf{Fin}(I)$ is the set of finite subsets of I.

- The axiom scheme of *bounded non-determinacy*:

$$\text{(BN)} \quad \Box \bigvee_{i \in I} \phi_i \leq \bigvee_{J \in \mathsf{Fin}(I)} \Box \bigvee_{j \in J} \phi_j \quad (\phi_i \in \mathcal{L}_\omega \ ).$$

- The axiom scheme of *finite approximability*:

$$\text{(FA)} \quad \bigwedge_{J \in \mathsf{Fin}(I)} \Box \bigwedge_{j \in J} \phi_j \leq \Diamond \bigwedge_{i \in I} \phi_i \quad (\phi_i \in \mathcal{L}_\omega \ ).$$

Note that these axioms are duals. Since the opposite entailments are theorems of $\mathcal{L}_\infty$, we shall in fact use (BN) and (FA) to denote the corresponding *equations*. The axioms could equivalently be formulated as: $\Box$ preserves directed joins, $\Diamond$ preserves filtered meets.

What are the intuitions behind these axioms? (BN) is (thinking of each process as the set of its capabilities and each ϕ_i as an open set) exactly a statement of *compactness*; the link between compactness and the computational notion of bounded non-determinacy is well-known from the literature on powerdomains [Plo81, Smy83b].

The axiom of finite approximability is less familiar from either the topological or the computer science literature. It is best understood as a logical (or localic) expression of the idea that only *closed* sets are taken as elements of a finitary powerdomain construction (or, better put, that from the point of view of finite observability we cannot distinguish between a set and its closure). The best way to get a more precise understanding is probably to read the proof of the next Theorem.

The duality between the two axioms is reminiscent of the discussion of finite *breadth* (BN) and finite *length* (FA) limitations of testing in [Abr83a].

Definition 5.5.14 A transition system is *finitary* if it satisfies (all instances of) (BN) and (FA). The class of finitary transition systems is denoted **FTS**.

As a first step, we shall give a substantive example of a finitary transition system. As we will see, it is actually the best possible example.

Theorem 5.5.15 *$\mathcal{D}$ is a finitary transition system.*

Proof. By the Duality Theorem 5.4.4, we have a map

$$[\![\cdot]\!] : \mathcal{L}_{\omega\pi} \longrightarrow K\Omega(\mathcal{D})$$

$$[\![\phi]\!] \equiv \{d \in \mathcal{D} : d \models \phi\}.$$

Now for $d \in \mathcal{D}$,

$$d \models \Box \bigvee_{i \in I} \phi_i \implies d \models \bigvee_{J \in \mathsf{Fin}(I)} \Box \bigvee_{j \in J} \phi_j$$

is just the statement

$$d \subseteq \bigcup_{i \in I} O_i \implies \exists J \in \mathsf{Fin}(I).\, d \subseteq \bigcup_{j \in J} O_j,$$

where $O_i = [\![\phi_i]\!]$, i.e. that d is compact as a subset of $\sum_{a \in \mathsf{Act}} \mathcal{D}$. Since $d \in \mathcal{D} \cong P^0[\sum_{a \in \mathsf{Act}} \mathcal{D}]$, and elements of the Plotkin powerdomain are Scott-compact subsets of the base domain ([Plo81]), this proves that $\mathcal{D}$ satisfies (BN).

Next we show that $\mathcal{D}$ satisfies (FA). Since there are only countably many distinct formulae in $\mathcal{L}_\omega$, it suffices to prove the following:

- Given a sequence $\{U_n\}$ of compact-open subsets of $\mathcal{D}$, with $U_n \supseteq U_{n+1}$ $(n \in \omega)$, and an element $d \in \mathcal{D}$ such that $d \cap U_n \neq \varnothing$ $(n \in \omega)$, then $d \cap \bigcap_{n \in \omega} U_n \neq \varnothing$.

(The alternative case for $d \models U_n$, namely $\bot \in U_n$ for all n, is trivial.)

Since each U_n is compact-open, it has the form $\uparrow B_n$, where B_n is a finite subset of $\mathcal{K}(\mathcal{D}\,)$. Also, $B_n \sqsubseteq_u B_{n+1}$, where

$$X \sqsubseteq_u Y \;\equiv\; \forall y \in Y.\, \exists x \in X.\, x \sqsubseteq y \quad (X, Y \subseteq \mathcal{D}\,).$$

Now define

$$C_n \;\equiv\; \{b \in B_n : \exists x \in d.\, b \sqsubseteq x\} \quad (n \in \omega).$$

Since $d \cap U_n \neq \varnothing$, $C_n \neq \varnothing$ for all n. Also, $C_n \sqsubseteq_u C_{n+1}$. Thus by König's Lemma in the form given e.g. in [Niv81], there is a sequence $\{c_n\}$ with $c_n \sqsubseteq c_{n+1}$ and $c_n \in C_n$. Now define

$$e_n \;\equiv\; \{\!|c_n|\!\} \uplus \{\!|\bot|\!\} \quad (n \in \omega).$$

Clearly $e_n \sqsubseteq e_{n+1}$ and $e_n \sqsubseteq d$ for all n, whence $\bigsqcup e_n \sqsubseteq d$. But $\bigsqcup c_n \in \bigsqcup e_n$ (using the description of least upper bounds of chains in the Plotkin powerdomain given in [Plo76, Theorem 8]), and so for some $x \in d$, $\bigsqcup c_n \sqsubseteq x$. Since $\bigsqcup c_n \in U_n$ for all n, $d \cap \bigcap_{n \in \omega} U_n \neq \varnothing$, and the proof is complete. ∎

We now draw some striking consequences from the finitary axioms.

Definition 5.5.16 A formula $\phi \in \mathcal{L}_\infty$ is in *finitary normal form* if it has the form

$$\bigwedge_{i \in I} \bigvee_{j \in J_i} \phi_{ij} \quad (\phi_{ij} \in \mathcal{L}_\omega).$$

Lemma 5.5.17 *For each $\phi \in \mathcal{L}_\infty$, for some finitary normal form ψ:*

$$(BN) + (FA) \vdash \phi = \psi.$$

PROOF. An easy induction on $\mathsf{ht}(\phi)$. ∎

Proposition 5.5.18 *In any finitary transition system $\mathcal{T}$, for all $p, q \in \mathrm{Proc}$:*

$$\mathcal{L}_\infty(p) \subseteq \mathcal{L}_\infty(q) \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

PROOF. The left to right implication is immediate. For the converse, suppose $\mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q)$, and $p \models \phi$, $(\phi \in \mathcal{L}_\infty)$. By 5.5.17,

$$(\mathrm{BN}) + (\mathrm{FA}) \vdash \phi = \bigwedge_{i \in I} \bigvee_{j \in J_i} \phi_{ij} \quad (\phi_{ij} \in \mathcal{L}_\omega)$$

hence since $\mathcal{T} \models (\mathrm{BN}) + (\mathrm{FA})$, $\mathcal{T} \models \phi = \bigwedge_{i \in I} \bigvee_{j \in J_i} \phi_{ij}$, and

$$\begin{array}{ll} \bullet & p \models \bigwedge_{i\in I} \bigvee_{j \in J_i} \phi_{ij} \\ \Longrightarrow & \forall i \in I.\, \exists j \in J_i.\, p \models \phi_{ij} \\ \Longrightarrow & \forall i \in I.\, \exists j \in J_i.\, q \models \phi_{ij} \\ \Longrightarrow & q \models \bigwedge_{i\in I} \bigvee_{j \in J_i} \phi_{ij} \\ \Longrightarrow & q \models \phi. \end{array}$$

∎

Theorem 5.5.19 (Finitary Characterisation Theorem) *With notation as in the previous Proposition:*

$$p \lesssim^B q \iff p \lesssim_\omega q \iff p \lesssim^F q \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

PROOF. Combine Theorems 5.5.6, 5.5.8 and 5.5.18. ∎

In order to continue our study of finitary transition systems, we need to introduce some notions from our final topic of this section.

Universal Semantics

Given any transition system and $p \in \mathrm{Proc}$, it is easy to see that $\mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega$ satisfies the axioms of a prime filter; hence we have a map

$$\mathcal{L}_\omega(\cdot) : \mathrm{Proc} \longrightarrow \mathsf{Spec}\, \mathcal{L}_\omega\ .$$

If we compose this with the isomorphism $\mathsf{Spec}\ \mathcal{L}_\omega \cong \mathcal{D}$ from the Duality Theorem 5.4.4, we get a map

$$[\![\cdot]\!] : \mathrm{Proc} \longrightarrow \mathcal{D}$$

which takes each process to an element of our domain. This map can be regarded as a *syntax-free denotational semantics*; it is *universal* since it is defined on every transition system.

Theorem 5.5.20 (Universal Semantics) *For any transition system $\mathcal{T}$ with* $p, q \in \mathrm{Proc}$*:*

$$(i) \quad p \lesssim^F q \iff [\![p]\!] \sqsubseteq [\![q]\!]$$

$$(ii) \quad p \sim^F [\![p]\!].$$

If $\mathcal{T}$ is finitary, then:

$$(iii) \quad p \lesssim^B q \iff [\![p]\!] \sqsubseteq [\![q]\!]$$

$$(iv) \quad p \sim^B [\![p]\!].$$

PROOF. Clearly (i) follows from (ii), and (iii) from (iv). Now $\mathcal{L}_\omega(p) = \mathcal{L}_\omega([\![p]\!])$; and so (ii) follows from 5.5.8; while (iv) follows from 5.5.19. ∎

We can think of 5.5.20 as a *full abstraction theorem* [Mil75, Plo77, Mil77] for our semantics; it says that every transition system (finitary transition system) can be embedded in $\mathcal{D}$ with as much identification as possible modulo the finitary equivalence (bisimulation).

Since $\mathcal{D}$ can itself be viewed as a transition system, we can tie things up even more neatly. Let **TS** be the category with objects the transition systems, and morphisms $\mathcal{T}_1 \to \mathcal{T}_2$ maps

$$f : \mathrm{Proc}_1 \to \mathrm{Proc}_2$$

for which

$$\mathcal{L}_\omega(p) = \mathcal{L}_\omega(f(p)) \quad (p \in \mathrm{Proc}_1).$$

It is clear that for such f

$$p \lesssim^F q \iff f(p) \lesssim^F f(q),$$

and if $\mathcal{T}_1$ and $\mathcal{T}_2$ are finitary,

$$p \lesssim^B q \iff f(p) \lesssim^B f(q).$$

Now we have

Theorem 5.5.21 (Final Algebra Theorem) *$\mathcal{D}$ is final in* **TS**, *and also in the subcategory* **FTS** *of finitary transition systems.*

PROOF. All we need to show is that the semantic map $[\![\cdot]\!]$ is the unique morphism from a transition system to $\mathcal{D}$. But for $d_1, d_2 \in \mathcal{D}$,

$$\begin{aligned} \mathcal{L}_\omega(d_1) \subseteq \mathcal{L}_\omega(d_2) &\iff K\Omega(d_1) \subseteq K\Omega(d_2) && \text{by 5.4.4} \\ &\iff d_1 \sqsubseteq d_2 && \text{since } \mathcal{D} \text{ is coherent,} \end{aligned}$$

which gives uniqueness. ∎

Finitary Transition Systems Resumed

Firstly, some conditions equivalent to finitariness.

Proposition 5.5.22 *For any transition system $\mathcal{T}$, the following conditions are equivalent:*

(i) $\mathcal{T}$ *is finitary*

(ii) $\forall p \in \text{Proc}.\, p \sim^B [\![p]\!]$

(iii) $\lesssim^B = \lesssim^F$ *in the combined system $\mathcal{T} + \mathcal{D}$ (disjoint union).*

PROOF. $(i) \implies (ii)$ is 5.5.20 (iv); $(ii) \implies (iii)$ since $\mathcal{D}$ is finitary.
$(ii) \implies (i)$. Suppose that $\mathcal{T}$ is not finitary, in particular that (BN) fails; i.e. that for some $p \in \text{Proc}$,

$$p \models \Box \bigvee_{i \in I} \phi_i \quad (\phi_i \in \mathcal{L}_\omega)$$

and $\forall J \in \mathsf{Fin}(I).\, p \not\models \bigvee_{j \in J} \phi_j$. Since $\mathcal{L}_\omega(p) = \mathcal{L}_\omega([\![p]\!])$, and each $\bigvee_{j \in J} \phi_j \in \mathcal{L}_\omega$, $[\![p]\!] \not\models \bigvee_{j \in J} \phi_j$ for all $J \in \mathsf{Fin}(I)$; hence since $[\![p]\!] \in \mathcal{D}$ and $\mathcal{D}$ is finitary, $[\![p]\!] \not\models \Box \bigvee_{i \in I} \phi_i$. Thus $\mathcal{L}_\infty([\![p]\!]) \neq \mathcal{L}_\infty(p)$, and so by 5.5.6 $p \sim^B [\![p]\!]$. The case when (FA) fails is similar.
$(iii) \implies (ii)$. Suppose for some p, $p \not\sim^B [\![p]\!]$. Then since $p \sim^F [\![p]\!]$ by 5.5.20 (ii), $\lesssim^B \neq \lesssim^F$. ∎

Note that in part (iii) of this Proposition we have "added in" $\mathcal{D}$ to the given transition system $\mathcal{T}$. This is to overcome the problem that there may not be enough processes in $\mathcal{T}$ alone to cause $\lesssim^B = \lesssim^F$ to fail.

Now we relate some of the finitariness conditions of Section 2 to our axioms.

Proposition 5.5.23 *(i) Weakly finite branching is equivalent to weakly image finite plus weakly initials finite.*
(ii) Weakly finite branching implies (BN).
(iii) (BN) implies weakly initials finite.
(iv) (BN) + (FA) do not *imply weakly image finite.*

Proof. (i). Easy.
(ii). Suppose $p \models \Box \bigvee_{i\in I} \phi_i$. $(\bigvee_{i\in I} \phi_i)\uparrow \Leftrightarrow \exists i \in I.\, \phi_i\uparrow$, in which case $\vdash \phi_i = t$, and the conclusion is trivial. Otherwise, $p\downarrow$, and so $C(p)$ is finite, say

$$C(p) = \{<a_1, p_1> , \ldots , <a_n, p_n>\}.$$

Then for each k with $1 \leq k \leq n$, $<a_k, p_k> \models \phi_{i_k}$ for some $i_k \in I$, and so $p \models \Box \bigvee_{j\in J} \phi_j$, where $J = \{i_1, \ldots , i_n\}$.
(iii). Assume (BN) and $p\downarrow$. Then $p \models \Box \bigvee_{a\in\mathsf{Act}} a(t\ )$, and so by (BN)

$$p \models \bigvee_{J\in\mathsf{Fin}(\mathsf{Act})} \Box \bigvee_{a\in J} a(t\ ),$$

which says exactly that p has a finite set of initial actions.
(iv). $\sum_{n\in\omega} a^n + a^\omega$ is in $\mathcal{D}$. ∎

All the usual finitary calculi are weakly finite branching, and so satisfy (BN). However, in general these calculi do *not* satisfy (FA) (analogously to the fact that generating trees over domains do not yield closed sets, although they always yield compact ones; cf. [Plo81]). As a standard counterexample, define

$$\begin{aligned} p &\equiv \textstyle\sum_{n\in\omega} a^n\mathbb{O} + \Omega \\ \phi_0 &\equiv t \\ \phi_{k+1} &\equiv a(\Diamond\phi_k). \end{aligned}$$

Then for all $J \in \mathsf{Fin}(\omega)$, $p \models \Diamond \bigwedge_{j\in J} \phi_j$, but $p \not\models \Diamond \bigwedge_{i\in\omega} \phi_i$.

Thus if p can be defined in our calculus, it does not satisfy (FA). Since p *can* be defined in CCS, SCCS (see next section), etc., these calculi are not finitary transition systems according to Definition 5.5.14. However, we can take the view that if we only take account of *observable* information via the semantics $[\![\cdot]\!]$, we have collapsed the given system into a finitary one which will actually, by Theorems 5.5.20 and 5.5.21, be isomorphic to a subsystem (or, topologically, a subspace) of $\mathcal{D}$.

Comparison Theorems Resumed

We now return to the question of finding a suitable correspondence between the finitary parts of HML and $\mathcal{L}$. As confirmation of our claim that HML_ω is unsatisfactory, we have:

Observation. HML_ω does not characterise $\lesssim^F$.

In fact, 5.2.7 provides a counter-example since, with the notation used there, $p \not\lesssim^F q$ while $\mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q)$.

We can get an idea of how to extend HML_ω by inspection of the translation functions 5.5.3. Although $(\cdot)^\dagger$ introduces infinitary conjunctions, these are of a special kind, for which a finitary counterpart can be found.

Definition 5.5.24 HML^+ is the extension of HML_ω with additional atomic fomulae of the form

$$\mathsf{init}(A) \quad (A \in \mathsf{Fin}(\mathsf{Act})).$$

The definition of the satisfaction relation is extended by

$$p \models \mathsf{init}(A) \equiv p\downarrow \; \& \; \{a \in \mathsf{Act} : \exists q.\, p \xrightarrow{a} q\} \subseteq A.$$

We can now modify the translation function $(\cdot)^\dagger$ as follows:

$$(\Box \bigvee_{i \in I} a_i(\phi_i))^\dagger \;\equiv\; \bigwedge_{i \in I} [a_i](\phi_i)^\dagger \;\wedge\; \mathsf{init}(\{a_i : i \in I\}).$$

Proposition 5.5.4 clearly still holds with this modification, and $(\cdot)^\dagger$ now cuts down to a function

$$\mathsf{N}\mathcal{L}_\omega \longrightarrow \mathrm{HML}^+.$$

There is still a mismatch in the other direction, since $(\cdot)^*$ introduces infinite disjunctions. To overcome this, we have to make the assumption that the transition system satisfies (BN)—a mild one, as 5.5.23 and the ensuing discussion shows.

Let $\mathcal{L}_{\bigvee\infty}$ be the sublanguage of $\mathcal{L}_\infty$ obtained by the restriction to finite *conjunctions* (but with infinite disjunctions still allowed).

Proposition 5.5.25 *In any transition system satisfying (BN), for all $p, q \in$ Proc:*

$$\mathcal{L}_{\bigvee\infty}(p) \subseteq \mathcal{L}_{\bigvee\infty}(q) \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

PROOF. Just like 5.5.18. ∎

Clearly, $(\cdot)^*$, extended by the clause

$$(\mathsf{init}(A))^* \equiv \Box \bigvee \{a(t \ : a \in A\}$$

cuts down to a function

$$\mathrm{HML}^+ \longrightarrow \mathsf{N}\mathcal{L}_{\bigvee \infty}.$$

We thus arrive at our

Theorem 5.5.26 (Comparison Theorem (Finitary Case)) *With notation as in the previous Proposition:*

$$\mathrm{HML}^+(p) \subseteq \mathrm{HML}^+(q) \iff \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q).$$

5.6 Full Abstraction for SCCS

So far, we have worked with abstract transition systems, in a syntax-free fashion. This degree of abstraction carries a price; we lose compositionality. Indeed, we need syntax to *define* compositionality. Accordingly, in this Section we turn to a particular transition system specified by an algebraic syntax, namely Milner's SCCS [Mil83]. We equip our domain $\mathcal{D}$ with a continuous algebraic structure corresponding to the signature of SCCS. Our main result is that the resulting denotational semantics for SCCS is *fully abstract* [Mil75, Plo77] with respect to bisimulation for finite terms, and with respect to the finitary preorder for recursive terms. As a by-product we will show that $\mathcal{D}$ is isomorphic to Hennessy's term model [Hen81], and hence obtain a complete axiomatisation of its equational theory as an immediate consequence of Hennessy's results.

Our choice of SCCS is for illustrative purposes, because it is simple and yet expressive. Similar accounts could be given for CCS [Mil80], MEIJE [AB84], ACP [BK84], etc. Note, however, that our semantics is fully abstract with respect to the *strong* congruence in Milner's terminology [Mil83], where all actions are observable. A corresponding treatment of *observation equivalence* [HM85], where unobservable actions are factored out, is still an open problem as far as I know; some hints of a possible approach may be gleaned from [Abr87b].

We begin by recalling some basic definitions on SCCS from [Mil83, Hen81]. We assume familiarity with basic notions of universal algebra; see e.g. [GTW78, EM85].

We fix a set of actions Act, which we assume comes equipped with an *abelian monoid* structure comprising

- an associative, commutative binary operation which we denote by juxtaposition, e.g. ab
- a unit 1.

The (one-sorted) signature Σ of SCCS is then defined as follows:

Definition 5.6.1 $\Sigma = \{\Sigma_n\}_{n\in\omega}$, where Σ_n is the set of operation symbols of arity n in Σ.

$$\Sigma_0 \equiv \{\mathbb{O}, \Omega\}$$

$$\begin{array}{rcl}\Sigma_1 & \equiv & \{a\_ : a \in \mathsf{Act}\} \cup \{\_\lceil A : A \subseteq \mathsf{Act}\} \\ & & \cup \{\_[S] : S \text{ is a monoid endomorphism on } \mathsf{Act}\} \\ \Sigma_2 & \equiv & \{+, \times\} \\ \Sigma_n & \equiv & \varnothing, \quad n > 2.\end{array}$$

Thus our version of SCCS only has *finite* sums (in contrast with [Mil83]), and has a constant for the undefined process as in [Hen81].

We define the subsignature $\Sigma' \subseteq \Sigma$ to be obtained by omitting the *restriction* operators $\_\lceil A$, the *relabelling* operators $\_[S]$, and the *synchronous product* operator $\times$, leaving only the *nullary sum* $\mathbb{O}$, the *binary sum* $+$, *prefixing* $a\_$, and the *undefined* process Ω.

We take the *finite processes* of SCCS to be the terms over the signature Σ, i.e. the elements of the term algebra T_Σ. Evidently, we can take the elements of $T_{\Sigma'}$ as notations for the finite synchronisation trees ST_ω.

Definition 5.6.2 (Operational Semantics) We make T_Σ into a transition system by defining the transition relation and divergence predicate in a syntax-directed way, as the *least* relations satisfying the following axioms and rules:

$$(D\Omega) \quad \Omega\uparrow$$

$$(D+L) \quad \frac{t_1\uparrow}{(t_1+t_2)\uparrow} \qquad (D+R) \quad \frac{t_2\uparrow}{(t_1+t_2)\uparrow}$$

$$(D\lceil) \quad \frac{t\uparrow}{(t\lceil A)\uparrow} \qquad (DS) \quad \frac{t\uparrow}{t[S]\uparrow}$$

$$(D\times L) \quad \frac{t_1\uparrow}{t_1 \times t_2\uparrow} \qquad (D\times R) \quad \frac{t_2\uparrow}{t_1 \times t_2\uparrow}$$

$$(Ta) \quad at \xrightarrow{a} t$$

$$(T+L) \quad \frac{t_1 \xrightarrow{a} t_1'}{t_1 + t_2 \xrightarrow{a} t_1'} \qquad (T+R) \quad \frac{t_2 \xrightarrow{a} t_2'}{t_1 + t_2 \xrightarrow{a} t_2'}$$

$$(T\lceil) \quad \frac{t \xrightarrow{a} t',\ a \in A}{t\lceil A \xrightarrow{a} t'\lceil A} \qquad (TS) \quad \frac{t \xrightarrow{a} t'}{t[S] \xrightarrow{Sa} t'[S]}$$

$$(T\times)\ \frac{t_1 \xrightarrow{a} t_1' \quad t_2 \xrightarrow{b} t_2'}{t_1 \times t_2 \xrightarrow{ab} t_1' \times t_2'}$$

For an illuminating discussion of the conceptual basis for these and related axioms, see [Mil86].

We now have a transition system $(T_\Sigma, \mathsf{Act}, \rightarrow, \uparrow)$ implicitly defined by 5.6.2. The following proposition gives a more explicit description of this system.

Proposition 5.6.3 *For all* $t, t_1, t_2 \in T_\Sigma$*:*

$$\begin{array}{rlcl}
(i)(a) & \mathbb{O}\downarrow & & (b)\ \mathbb{O} \stackrel{a}{\not\rightarrow} \\
(ii)(a) & \Omega\uparrow & & (b)\ \Omega \stackrel{a}{\not\rightarrow} \\
(iii)(a) & at\downarrow & & \\
(b) & at_1 \xrightarrow{b} t_2 & \iff & b = a \;\&\; t_1 = t_2 \\
(iv)(a) & (t_1 + t_2)\uparrow & \iff & t_1\uparrow \text{ or } t_2\uparrow \\
(b) & (t_1 + t_2) \xrightarrow{a} t & \iff & t_1 \xrightarrow{a} t \text{ or } t_2 \xrightarrow{a} t \\
(v)(a) & (t\restriction A)\uparrow & \iff & t\uparrow \\
(b) & t_1\restriction A \xrightarrow{a} t_2 & \iff & \exists t.\, t_1 \xrightarrow{a} t \;\&\; t_2 = t\restriction A \;\&\; a \in A \\
(vi)(a) & t[S]\uparrow & \iff & t\uparrow \\
(b) & t_1[S] \xrightarrow{a} t_2 & \iff & \exists b, t.\, t_1 \xrightarrow{b} t \;\&\; t_2 = t[S] \;\&\; a = Sb \\
(vii)(a) & (t_1 \times t_2)\uparrow & \iff & t_1\uparrow \text{ or } t_2\uparrow \\
(b) & t_1 \times t_2 \xrightarrow{a} t & \iff & \exists t_1', t_2', b_1, b_2.\, t_i \xrightarrow{b_i} t_i'\ (i = 1, 2) \\
 & & & \&\; t = t_1' \times t_2' \;\&\; a = b_1 b_2.
\end{array}$$

Proof. By induction on the length of proofs of $t\uparrow$ and $t_1 \xrightarrow{a} t_2$. ∎

Now given any Σ-algebra $\mathcal{A}$, by initiality of T_Σ there is a unique Σ-homomorphism

$$[\![\cdot]\!]^{\mathcal{A}} : T_\Sigma \longrightarrow \mathcal{A},$$

which is just another notation for a compositional denotational semantics as in [MS76, Sto77, Gor79]. Thus to form a denotational semantics $[\![\cdot]\!]^{\mathcal{D}}$ based on our domain $\mathcal{D}$, it suffices to define each operation in Σ as a function of the appropriate arity over $\mathcal{D}$. We shall in fact define the operations so that they are *continuous* over $\mathcal{D}$.

Definition 5.6.4 We specify a Σ-structure on $\mathcal{D}$:

$$\begin{array}{rlcl} (i) & \mathbb{O}^{\mathcal{D}} & \equiv & \emptyset \\ (ii) & \Omega^{\mathcal{D}} & \equiv & \{\!|\bot|\!\} \\ (iii) & a\_^{\mathcal{D}} & \equiv & \lambda d \in \mathcal{D}.\{\!|<a, d>|\!\} \\ (iv) & +^{\mathcal{D}} & \equiv & \uplus \end{array}$$

Restriction:

$$(v)\quad (\_\lceil A)^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D} \to \mathcal{D}].\ \biguplus \circ P^0(g_A \Phi)$$

where

$$g_A : [\mathcal{D} \to \mathcal{D}] \to [\sum_{a \in \mathsf{Act}} \mathcal{D} \to \mathcal{D}]$$

is defined by

$$\begin{array}{rcl} g_A \Phi \bot & = & \{\!|\bot|\!\} \\ g_A \Phi <a, d> & = & \begin{cases} \{\!|<a, \Phi d>|\!\} & \text{if } a \in A \\ \emptyset & \text{otherwise} \end{cases} \end{array}$$

(i.e.

$$g_A \Phi = \coprod_{a \in A} \lambda d \in \mathcal{D}.\{\!|<a, \Phi d>|\!\} \amalg \coprod_{a \in \mathsf{Act}-A} \lambda d \in \mathcal{D}.\emptyset,$$

where $\amalg$ is "source tupling" [WBT85]).
Relabelling:

$$(vi)\quad (\_[S])^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D} \to \mathcal{D}].\ P^0(g_S \Phi)$$

where

$$g_S : [\mathcal{D} \to \mathcal{D}] \to [\sum_{a \in \mathsf{Act}} \mathcal{D} \to \sum_{a \in \mathsf{Act}} \mathcal{D}]$$

is defined by

$$\begin{aligned} g_S \Phi \bot &= \bot \\ g_S \Phi {<}a, d{>} &= {<}Sa, \Phi d{>} \end{aligned}$$

Product:

$$(vii)\ \ \times^{\mathcal{D}} \equiv \mu\Phi \in [\mathcal{D}^2 \to \mathcal{D}].\,(f\Phi)^{\dagger}$$

where

$$f : [\mathcal{D}^2 \to \mathcal{D}] \to [(\sum_{a \in \mathsf{Act}} \mathcal{D})^2 \to \sum_{a \in \mathsf{Act}} \mathcal{D}]$$

is defined by

$$\begin{aligned} f\Phi(x, \bot) = f\Phi(\bot, x) &= \bot \\ f\Phi({<}a, d{>}, {<}b, e{>}) &= {<}ab, \Phi(d, e){>} \end{aligned}$$

The only point which needs to be checked to ensure that this definition yields well-defined continuous functions is that $g_A\Phi$, $g_S\Phi$ and $f\Phi$ are (bi)strict and continuous, which is immediate from the definitions. Note that restriction, relabelling and product are defined recursively, while sum and prefixing are interpreted by the basic operations derived from the domain equation for $\mathcal{D}$. This corresponds to the fact that restriction, relabelling and product can be *eliminated* (for finite terms) in the equational theory of SCCS modulo bisimulation.

The continuous Σ-algebra defined by 5.6.4 is denoted $\mathcal{D}_\Sigma$. The following is an easy consequence of 5.6.4 and 5.3.10.

Proposition 5.6.5 *The semantic function*

$$[\![\cdot]\!]^{\mathcal{D}} : T_\Sigma \longrightarrow \mathcal{D}_\Sigma$$

cuts down to surjections

$$T_\Sigma \twoheadrightarrow \mathcal{K}(\mathcal{D}), \quad T_{\Sigma'} \twoheadrightarrow \mathcal{K}(\mathcal{D}).$$

Thus the finite synchronisation trees provide a notation for the finite elements of $\mathcal{D}$.

We now relate our definitions of the SCCS operations on $\mathcal{D}$ to the transition system view of $\mathcal{D}$.

Proposition 5.6.6 *For all $d, d_1, d_2 \in \mathcal{K}(\mathcal{D})$:*

$$
\begin{array}{rlcl}
(i)(a) & \mathbb{0}^{\mathcal{D}}\downarrow & & (b)\ \mathbb{0}^{\mathcal{D}} \stackrel{a}{\not\rightarrow} \\
(ii)(a) & \Omega^{\mathcal{D}}\uparrow & & (b)\ \Omega^{\mathcal{D}} \stackrel{a}{\not\rightarrow} \\
(iii)(a) & a^{\mathcal{D}}d\downarrow & & \\
(b) & a^{\mathcal{D}}d_1 \stackrel{b}{\rightarrow} d_2 & \iff & b = a \;\&\; d_1 = d_2 \\
(iv)(a) & (d_1 +^{\mathcal{D}} d_2)\uparrow & \iff & d_1\uparrow \textit{ or } d_2\uparrow \\
(b) & d_1 +^{\mathcal{D}} d_2 \stackrel{a}{\rightarrow} d & \iff & d_1 \stackrel{a}{\rightarrow} d \textit{ or } d_2 \stackrel{a}{\rightarrow} d
\end{array}
$$

Restriction:

$$
\begin{array}{rlcl}
(v)(a) & (d\lceil^{\mathcal{D}}A)\uparrow & \iff & d\uparrow \\
(b) & d_1\lceil^{\mathcal{D}}A \stackrel{a}{\rightarrow} d_2 & \iff & \exists e_1, e_2.\, d_1 \stackrel{a}{\rightarrow} e_i,\ (i = 1, 2) \\
 & & & \&\; e_1\lceil^{\mathcal{D}}A \sqsubseteq d_2 \sqsubseteq e_2\lceil^{\mathcal{D}}A \\
 & & & \&\; a \in A
\end{array}
$$

Relabelling:

$$
\begin{array}{rlcl}
(vi)(a) & (d[S]^{\mathcal{D}})\uparrow & \iff & d\uparrow \\
(b) & d_1[S]^{\mathcal{D}} \stackrel{a}{\rightarrow} d_2 & \iff & \exists e_1, e_2, b_1, b_2.\, d_1 \stackrel{a}{\rightarrow} e_i,\ (i = 1, 2) \\
 & & & \&\; e_1[S]^{\mathcal{D}} \sqsubseteq d_2 \sqsubseteq e_2[S]^{\mathcal{D}} \\
 & & & \&\; Sb_1 = a = Sb_2
\end{array}
$$

Product:

$$(vii)(a) \quad (d_1 \times^{\mathcal{D}} d_2)\uparrow \iff d_1\uparrow \textit{ or } d_2\uparrow$$

$$\begin{aligned}(b) \quad d_1 \times^{\mathcal{D}} d_2 \xrightarrow{a} d \iff & \exists u_i, v_i, b_i, c_i \ (i = 1,2). \\ & d_1 \xrightarrow{b_i} u_i \ \& \ d_2 \xrightarrow{c_i} v_i \ (i = 1,2) \\ & \& \ (u_1 \times^{\mathcal{D}} v_1) \sqsubseteq d \sqsubseteq (u_2 \times^{\mathcal{D}} v_2) \\ & \& \ b_i c_i = a \ (i = 1,2).\end{aligned}$$

Proof. We give two cases for illustration.
(v). We define

$$\begin{aligned}\Theta \ \equiv \ & \{\{<a, d'\lceil^{\mathcal{D}} A>\} : <a, d'> \in d, a \in A\} \\ & \cup \{\varnothing : d = \emptyset \text{ or } \exists <a, d'> \in d.\, a \notin A\} \\ & \cup \{\{\bot\} : \bot \in d\}.\end{aligned}$$

Now

$$\begin{aligned}d\lceil^{\mathcal{D}} A \ &= \ Con(\bigcup \Theta^{\star}) \\ &= \ Con((\bigcup \Theta)^{\star}) \text{ by [Plo76] p. 477} \\ &= \ Con(\bigcup \Theta) \text{ since } d \in \mathcal{K}(\mathcal{D}) \\ &= \ Con(\{<a, d'\lceil^{\mathcal{D}} A> : <a, d'> \in d \ \& \ a \in A\} \\ & \qquad \cup \{\bot : \bot \in d\}),\end{aligned}$$

and (v) is readily derived from this description.
(vii). Similarly to (v),

$$\begin{aligned}d_1 \times^{\mathcal{D}} d_2 \ = \ & Con(\{<b_1 b_2, e_1 \times^{\mathcal{D}} e_2> : <b_i, e_i> \in d_i, \ i = 1,2\} \\ & \cup \{\bot : \bot \in d_1 \text{ or } \bot \in d_2\}). \ \blacksquare\end{aligned}$$

Proposition 5.6.7 *For all $t \in T_{\Sigma}$, $t \sim^{B} [\![t]\!]^{\mathcal{D}}$.*

Proof. Firstly, we define a height function on T_{Σ} in the obvious way:

$$\mathsf{ht}(\sigma(t_1, \ldots, t_n) = \sup \{\mathsf{ht}(t_i : 1 \leq i \leq n\} + 1.$$

As an easy consequence of 5.6.3, we have:

$$t \xrightarrow{a} t' \implies \mathsf{ht}(t') < \mathsf{ht}(t).$$

The proposition is proved by induction on $\mathsf{ht}(t)$, and cases on the construction of t. The cases arising from operations in Σ' are immediate in the light of the parallelism between 5.6.3 and 5.6.6. We give one of the remaining cases for illustration.

$t \equiv t_1 \restriction^{\mathcal{D}} A$. Firstly,

$$\begin{array}{rcll} t\uparrow & \iff & t_1\uparrow & \text{by 5.6.3(v)} \\ & \iff & [\![t_1]\!]^{\mathcal{D}}\uparrow & \text{by induction hypothesis} \\ & \iff & ([\![t_1]\!]^{\mathcal{D}} \restriction^{\mathcal{D}} A)\uparrow & \text{by 5.6.6(v)} \\ & \iff & [\![t_1 \restriction A)]\!]^{\mathcal{D}}\uparrow. & \end{array}$$

Next,

$$\begin{array}{lll} \bullet & t \xrightarrow{a} t' & \\ \implies & t_1 \xrightarrow{a} t_1' \;\&\; t' = t_1' \restriction A \;\&\; a \in A & \text{by 5.6.3(v)} \\ \implies & \exists d'.\, [\![t_1]\!]^{\mathcal{D}} \xrightarrow{a} d' \;\&\; t_1' \lesssim^B d' & \text{ind. hyp. on } t_1 \\ \implies & t_1' \restriction A \sim^B [\![t_1' \restriction A]\!]^{\mathcal{D}} & \text{ind. hyp. on } t_1' \restriction A \\ & \quad = [\![t_1']\!]^{\mathcal{D}} \restriction^{\mathcal{D}} A & \\ & \lesssim^B d' \restriction^{\mathcal{D}} A & \text{by 5.3.11} \\ & \text{(since } \restriction^{\mathcal{D}} \text{ is monotone)} & \\ \implies & \exists u.\, [\![t]\!]^{\mathcal{D}} \xrightarrow{a} u \;\&\; t' \lesssim^B u & \text{by 5.6.6(v).} \end{array}$$

Similarly, we can show

$$t \xrightarrow{a} t' \;\Rightarrow\; \exists u.\, [\![t]\!]^{\mathcal{D}} \xrightarrow{a} u \;\&\; u \lesssim^B t'.$$

Again,

$$
\begin{array}{ll}
\bullet & [\![t]\!]^{\mathcal{D}} \xrightarrow{a} d \\
\Longrightarrow & \exists d_1, d_2.\ [\![t_1]\!]^{\mathcal{D}} \xrightarrow{a} d_i,\ i = 1, 2 \\
& \&\ d_1{\restriction}^{\mathcal{D}} A \sqsubseteq d \sqsubseteq d_2{\restriction}^{\mathcal{D}} A \\
& \&\ a \in A \qquad\qquad\qquad\qquad \text{by 5.6.6(v)} \\
\Longrightarrow & \exists t'_1, t'_2.\ t_1 \xrightarrow{a} t'_i,\ i = 1, 2 \\
& \&\ t'_1 \lesssim^B d_1,\ d_2 \lesssim^B t'_2 \qquad \text{by induction hypothesis} \\
\Longrightarrow & t \xrightarrow{a} t'_i{\restriction} A,\ i = 1, 2 \\
& \&\ t'_1{\restriction} A \sim^B [\![t'_1{\restriction} A]\!]^{\mathcal{D}} \qquad \text{by induction hypothesis} \\
& = [\![t'_1]\!]^{\mathcal{D}}{\restriction}^{\mathcal{D}} A \lesssim^B d_1{\restriction}^{\mathcal{D}} A \lesssim^B d,
\end{array}
$$

and similarly $d \lesssim^B t'_2{\restriction} A$. Altogether, we have $t \sim^B [\![t]\!]^{\mathcal{D}}$. ∎

As an immediate consequence of this Proposition and 5.3.11 we have

Theorem 5.6.8 (Full Abstraction for Finite Terms) *For all* $t_1, t_2 \in T_\Sigma$*:*

$$t_1 \lesssim^B t_2 \iff [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}.$$

As further consequences of 5.6.8 we have

- $[\![\cdot]\!]^{\mathcal{D}}$ agrees with the syntax-free map $[\![\cdot]\!]$ defined in Section 5. Indeed, $t \sim^B [\![t]\!]^{\mathcal{D}}$ implies $\mathcal{L}_\omega([\![t]\!]^{\mathcal{D}}) = \mathcal{L}_\omega(t) = \mathcal{L}_\omega([\![t]\!])$, which implies $[\![t]\!]^{\mathcal{D}} = [\![t]\!]$.
- T_Σ is a finitary transition system, by 5.5.22.

Moreover, we can derive two further characterisations of $\mathcal{D}$.

Theorem 5.6.9 *(i)* $\mathcal{K}(\mathcal{D}) \cong (T_{\Sigma'}/{\sim^B}, \lesssim^B/{\sim^B})$*, and therefore*
(ii) $D \cong \mathsf{Idl}\,(T_{\Sigma'}/{\sim^B}, \lesssim^B/{\sim^B})$.

Proof. Immediate from 5.6.5 and 5.6.8. ∎

We recall the notion of *continuous* Σ*-algebra* [GTW78, Gue81]. This is just a Σ-algebra whose carrier is a cpo, and whose operations are continuous. A homomorphism of such algebras which is continuous on the carriers is a *continuous* Σ*-homomorphism.* The category of these algebras and homomorphisms is denoted $\mathbf{CAlg}(\Sigma)$.

Definition 5.6.10 **SCCS-Alg** is the full subcategory of $\mathbf{CAlg}(\Sigma)$ of those algebras $\mathcal{A}$ satisfying

$$\forall t_1, t_2 \in T_\Sigma.\, t_1 \lesssim^B t_2 \implies [\![t_1]\!]^{\mathcal{A}} \sqsubseteq [\![t_2]\!]^{\mathcal{A}}.$$

Theorem 5.6.11 *$\mathcal{D}_\Sigma$ is initial in* **SCCS-Alg**.

PROOF. We begin by recalling a useful fact about continuous algebras ([Gue81] Proposition 3.12). Suppose $\mathcal{A}$ is a continuous algebra whose carrier A is an algebraic domain, such that the finite elements $\mathcal{K}(A)$ form a Σ-subalgebra. Then, given any monotonic Σ-homomorphism

$$f : \mathcal{K}(A) \longrightarrow \mathcal{B}$$

to a continuous Σ-algebra $\mathcal{B}$, there is a unique extension

$$\hat{f} : \mathcal{A} \longrightarrow \mathcal{B}$$

to a continuous Σ-homomorphism on $\mathcal{A}$.

By 5.6.5, $\mathcal{K}(\mathcal{D})$ is closed under the Σ-operations. Hence it suffices to construct a unique monotone Σ-homomorphism

$$f : \mathcal{K}(\mathcal{D}) \longrightarrow \mathcal{A}$$

to any $\mathcal{A}$ in **SCCS-Alg**. Given $d \in \mathcal{K}(\mathcal{D})$, by 5.6.5 there is $t \in T_\Sigma$ with $[\![t]\!]^{\mathcal{D}} = d$, and the only possible definition for f giving a Σ-homomorphism is

$$f : d \mapsto [\![t]\!]^{\mathcal{A}}.$$

This establishes uniqueness. For existence,

$$\begin{aligned} [\![t_1]\!]^{\mathcal{D}} = [\![t_2]\!]^{\mathcal{D}} &\iff [\![t_1]\!]^{\mathcal{D}} \sim^B [\![t_2]\!]^{\mathcal{D}} && \text{by 5.3.11} \\ &\iff t_1 \sim^B t_2 && \text{by 5.6.8} \\ &\implies [\![t_1]\!]^{\mathcal{A}} = [\![t_2]\!]^{\mathcal{A}} \end{aligned}$$

since $\mathcal{A}$ is in **SCCS-Alg**, and so f is well-defined. Similarly,

$$[\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}} \Rightarrow t_1 \lesssim^B t_2 \Rightarrow [\![t_1]\!]^{\mathcal{A}} \sqsubseteq [\![t_2]\!]^{\mathcal{A}},$$

and so f is monotone. ∎

The purely algebraic part of SCCS which we have developed so far only allows the description of *finite* processes. We now extend the calculus with recursion.

Definition 5.6.12 We fix a set of variables Var, ranged over by x, y, z. The syntax of *recursive terms* REC_Σ, is then defined by

$$t \ ::= \ \sigma(t_1, \ldots, t_n) \ \ (\sigma \in \Sigma_n) \mid x \mid \mathsf{rec}\, x.t$$

In an obvious way, we can take T_Σ as a subset of REC_Σ. Note that $\mathsf{rec}\, x.t$ is a variable-binding construct. The set of *closed recursive terms* is denoted CREC_Σ.

We now extend the definition of the operational semantics to CREC_Σ:

$$(D\mathsf{rec}) \ \frac{t[\Omega/x]\!\uparrow}{\mathsf{rec}\, x.t\!\uparrow} \qquad (T\mathsf{rec}) \ \frac{t[\mathsf{rec}\, x.t/x] \stackrel{a}{\to} t'}{\mathsf{rec}\, x.t \stackrel{a}{\to} t'}$$

We thus obtain a transition system $(\mathrm{CREC}_\Sigma, \mathsf{Act}, \to, \uparrow)$. It is not too hard to see that this system is weakly finite-branching, and therefore by 5.5.23 satisfies (BN). However, most of the other finiteness conditions on transition systems fail, as the following examples show.

Examples

(1) **Failure of sort-finiteness.** Assume Act is infinite, in particular that $\{a_n\}$ is a sequence of distinct actions, and that S is a relabelling such that

$$Sa_n = a_{n+1} \ \ (n \in \omega).$$

Then

$$\mathsf{rec}\, x.\, a_0\mathbb{O} + x[S]$$

has the behaviour described by the synchronisation tree

$$\sum_{n \in \omega} a_n\mathbb{O} + \Omega.$$

(2) **Failure of (FA), and $\lesssim_\omega \neq \lesssim^B$.** By the example following 5.5.23, it suffices to show that the synchronisation tree

$$p \equiv \sum_{n \in \omega} a^n\mathbb{O} + \Omega$$

can be defined in SCCS to disprove (FA); while the same example shows that $\lesssim_\omega \neq \lesssim^B$, since

$$p \sim_\omega p + a^\omega, \ \ p \not\sim_{\omega+1} p + a^\omega,$$

and we can define $a^\omega \equiv \mathsf{rec} x. ax$. But using *unguarded* recursion (cf. [Mil83]), we can define

$$p \equiv (\mathsf{rec}\, x. (\Delta a + (\Delta a \times x))) \upharpoonright \{a\}$$

where $\Delta a \equiv \mathsf{rec}\, y.\, a1^\omega + 1y$.
(3) $\lesssim^F \neq \lesssim_\omega$. Again, following the examples after 5.5.10, it suffices to show that the synchronisation trees

$$\begin{aligned} p &\equiv a(\sum_{n \in \mathbb{N}} b_n \mathbb{O}) + \Omega \\ q &\equiv \sum_{n \in \mathbb{N}} a(\sum_{m \in \mathbb{N} - \{n\}} b_m \mathbb{O} + \Omega) + \Omega \end{aligned}$$

are definable in SCCS. Clearly p is definable in the same way as Example (1). For q, we need some additional assumptions on Act:

- There are $c, \{c_n\} \in \mathsf{Act}$ such that, for $k, m \in \mathbb{N}$:

$$\begin{aligned} c^{(k)} c_m &= b_m \quad (k \neq m) \\ c^{(m)} c_m &= b_{m+1} \end{aligned}$$

  where $c^{(k)} \equiv \underbrace{c \ldots c}_{k}$, i.e. the product in the monoid Act.

- There is a relabelling S such that

$$Sc_n = c_{n+1} \quad (n \in \mathbb{N}).$$

(To see that these requirements can be met, let Act be the free abelian monoid over the generators $0, a, b_k, c, c_k$ $(k \in \mathbb{N})$ subject to the relations

$$0x = x0 = 0, \quad c^{(k)} c_m = b_m \ (k \neq m), \quad c^{(m)} c_m = b_{m+1}$$

for $k, m \in \mathbb{N}$. Let S be the endomorphism induced by

$$S0 = Sa = Sb_k = Sc = 0, \quad Sc_k = c_{k+1},$$

which is well-defined since S preserves the relations.)

Then we can define

$$\begin{array}{rcl} q & \equiv & \mathsf{rec}\, x.\, ar + (1c\mathbb{O} \times x) \\ r & \equiv & \mathsf{rec}\, y.\, c_1\mathbb{O} + x[S], \end{array}$$

and calculate:

$$\begin{array}{rcl} r & = & \sum_{n\in\mathbb{N}} c_n\mathbb{O} + \Omega, \\ q & = & \sum_{n\in\mathbb{N}} (\prod_{i=1}^{n} 1c\mathbb{O} \times ar) + \Omega \\ & = & \sum_{n\in\mathbb{N}} a(c^{(n)}\mathbb{O} \times \sum_{m\in\mathbb{N}} c_m\mathbb{O} + \Omega) + \Omega \\ & = & \sum_{n\in\mathbb{N}} a(\sum_{m\in\mathbb{N}} (c^{(n)}c_m)\mathbb{O} + \Omega) + \Omega \\ & = & \sum_{n\in\mathbb{N}} a(\sum_{m\in\mathbb{N}-\{n\}} b_m\mathbb{O} + \Omega) + \Omega \end{array}$$

as required.

By contrast with Example (3), Hennessy claims in [Hen81] Theorem 4.1 that $\lesssim^F = \lesssim_\omega$ for SCCS. The defect in his argument occurs in the definition of $p^{(n)}$ at the start of section 4 of [Hen81]; there appears to be an implicit assumption that SCCS is sort-finite. Indeed, as an easy consequence of our work in the previous Section, we have

Proposition 5.6.13 *In any sort-finite transition system satisfying (BN):*

$$\lesssim^F = \lesssim_\omega.$$

Proof. Let $p, q \in \mathrm{Proc}$ in such a system.

$$\begin{array}{rcll} p\lesssim^F q & \Longrightarrow & \mathcal{L}_\omega(p) \subseteq \mathcal{L}_\omega(q) & \\ & \Longrightarrow & \mathcal{L}_{\bigvee\infty}(p) \subseteq \mathcal{L}_{\bigvee\infty}(q) & \text{(BN)} \\ & \Longrightarrow & \mathrm{HML}_\omega(p) \subseteq \mathrm{HML}_\omega(q) & \\ & \Longrightarrow & p\lesssim_\omega q & \text{sort-finiteness.} \blacksquare \end{array}$$

Nevertheless, Hennessy's results on full abstraction are valid when $\lesssim_\omega$ is replaced by $\lesssim^F$, and we shall make use of them shortly.

Firstly, we need to extend our denotational semantics $[\![\cdot]\!]^{\mathcal{D}}$ to recursive terms. This is done in the standard way; we introduce environments to deal with variables, and interpret recursion by least fixed points.

Definition 5.6.14 Denotational semantics of recursive terms:

$$\mathsf{Env} \equiv \mathcal{D}^{\mathsf{Var}}$$

$$[\![\cdot]\!]^{\mathcal{D}} : \mathrm{REC}_{\Sigma} \longrightarrow \mathsf{Env} \longrightarrow \mathcal{D}$$

$$\begin{array}{lcl} [\![x]\!]^{\mathcal{D}}\rho & \equiv & \rho x \\ [\![\sigma(t_1,\ldots,t_n)]\!]^{\mathcal{D}}\rho & \equiv & \sigma^{\mathcal{D}}([\![t_1]\!]^{\mathcal{D}}\rho,\ldots,[\![t_n]\!]^{\mathcal{D}}\rho) \\ [\![\mathsf{rec}\, x.\, t]\!]^{\mathcal{D}}\rho & \equiv & \mu d \in \mathcal{D}.\, [\![t]\!]^{\mathcal{D}}\rho[x \mapsto d]. \end{array}$$

We now want to extend our Full Abstraction Theorem to recursive terms. We can use Hennessy's results in [Hen81] to get a cheap proof. In that paper, Hennessy constructs a term model $\mathcal{I}$ with the following properties:

1. $\mathcal{I}$ is an algebraic continuous Σ-algebra all finite elements of which are definable in T_{Σ}.

2. $\mathcal{I}$ is fully abstract for recursive terms with repect to the finitary preorder; for all $t_1, t_2 \in \mathrm{CREC}_{\Sigma}$:

$$t_1 \precsim^F t_2 \iff [\![t_1]\!]^{\mathcal{I}} \sqsubseteq [\![t_2]\!]^{\mathcal{I}}.$$

Combining (1) and (2) with Theorem 5.6.11, we obtain

Theorem 5.6.15 *$\mathcal{D}_{\Sigma}$ and $\mathcal{I}$ are isomorphic as continuous Σ-algebras.*

Let $h : \mathcal{D}_{\Sigma} \to \mathcal{I}$ be the isomorphism given by Theorem 5.6.15. It is immediate that h preserves denotations of terms in T_{Σ}:

$$\forall t \in T_{\Sigma}.\, h([\![t]\!]^{\mathcal{D}}) = [\![t]\!]^{\mathcal{I}}.$$

To extend this to recursive terms we need one further piece of machinery.

Definition 5.6.16 Let $\simeq$ be the least Σ-congruence over REC_Σ generated by

$$\mathsf{rec}\, x.\, t \simeq t[\mathsf{rec}\, x.\, t/x].$$

Let t_Ω be the term obtained from t by replacing each subexpression of the form $\mathsf{rec}\, x.\, t'$ by Ω. The *syntactic approximants* of t are defined by:

$$SA(t) \equiv \{t'_\Omega : t' \simeq t\}.$$

Note that $SA(t) \subseteq T_\Sigma$ for all $t \in \text{CREC}_\Sigma$.

Now the following is standard (cf. e.g. [GTWW77]):

Lemma 5.6.17 (Syntactic Approximation) *For all* $t \in \text{CREC}_\Sigma$*:*

$$[\![t]\!]^{\mathcal{D}} = \bigsqcup\{[\![t']\!]^{\mathcal{D}} : t' \in SA(t)\}.$$

Hennessy proves the corresponding result for $[\![\cdot]\!]^{\mathcal{I}}$ as his Lemma 3.4.

Proposition 5.6.18 *For all* $t \in \text{CREC}_\Sigma$*:*

$$h([\![t]\!]^{\mathcal{D}}) = [\![t]\!]^{\mathcal{I}}.$$

Proof.

$$\begin{array}{rcll} h([\![t]\!]^{\mathcal{D}}) & = & h(\bigsqcup\{[\![t']\!]^{\mathcal{D}} : t' \in SA(t)\}) & \text{by 5.6.17} \\ & = & \bigsqcup\{h([\![t']\!]^{\mathcal{D}}) : t' \in SA(t)\} & h \text{ is continuous} \\ & = & \bigsqcup\{[\![t']\!]^{\mathcal{I}} : t' \in SA(t)\} & \text{by 5.6.15} \\ & = & [\![t]\!]^{\mathcal{I}}. & \blacksquare \end{array}$$

Theorem 5.6.19 (Full Abstraction for Recursive Terms) *For all* $t_1, t_2 \in \text{CREC}_\Sigma$*:*

$$t_1 \lesssim^F t_2 \iff [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}.$$

Proof.

$$\begin{array}{rcl} t_1 \lesssim^F t_2 & \iff & [\![t_1]\!]^{\mathcal{I}} \sqsubseteq [\![t_2]\!]^{\mathcal{I}} \\ & \iff & [\![t_1]\!]^{\mathcal{D}} \sqsubseteq [\![t_2]\!]^{\mathcal{D}}, \end{array}$$

by 5.6.18 and since h is an order-isomorphism. $\blacksquare$

Since $\mathcal{D}$ is algebraic, this result extends to terms with variables in the obvious way. It follows that the axiomatisation of the order and equality relations between terms of SCCS presented in [Hen81] is sound and complete for $\mathcal{D}_\Sigma$.

Chapter 6

Applications to Functional Programming: The Lazy Lambda-Calculus

6.1 Introduction

In this Chapter, we turn to our second case study, which concerns the foundations of functional programming. Once again, we aim not merely to exemplify our theory, but to use it in order to break some new ground.

The commonly accepted basis for functional programming is the λ-calculus; and it is folklore that the λ-calculus *is* the prototypical functional language in purified form. But what is the λ-calculus? The syntax is simple and classical; variables, abstraction and application in the pure calculus, with applied calculi obtained by adding constants. The further elaboration of the theory, covering conversion, reduction, theories and models, is laid out in Barendregt's already classical treatise [Bar84]. It is instructive to recall the following crux, which occurs rather early in that work (p. 39):

Meaning of λ-terms: first attempt

- The meaning of a λ-term is its normal form (if it exists).
- All terms without normal forms are identified.

This proposal incorporates such a simple and natural interpretation of the λ-calculus as a programming language, that if it worked there would surely be no doubt that it was the right one. However, it gives rise to an inconsistent theory! (see the above reference).

Second attempt

- The meaning of λ-terms is based on head normal forms via the notion of *Bohm tree*.
- All unsolvable terms (no head normal form) are identified.

This second attempt forms the central theme of Barendregt's book, and gives rise to a very beautiful and successful theory (henceforth referred to as the "standard theory"), as that work shows.

This, then, is the commonly accepted foundation for functional programming; more precisely, for the *lazy* functional languages, which represent the mainstream of current functional programming practice. Examples: MIRANDA [Tur85], LML [Aug84], LISPKIT [Hen80], ORWELL [Wad85], PONDER [Fai85], TALE [BvL86]. But do these languages as defined and implemented actually evaluate terms to head normal form? To the best of my knowledge, *not a single one of them does so*. Instead, they evaluate to *weak head normal form*, i.e. they do not evaluate under abstractions.

Example

$\lambda x.(\lambda y.y)M$ is in weak head normal form, but not in head normal form, since it contains the head redex $(\lambda y.y)M$.

So we have a mismatch between theory and practice. Since current practice is well-motivated by efficiency considerations and is unlikely to be abandoned readily, it makes sense to see if a good modified theory can be developed for it. To see that the theory really does need to be modified:

Example

Let $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ be the standard unsolvable term. Then

$$\lambda x.\Omega = \Omega$$

in the standard theory, since $\lambda x.\Omega$ is also unsolvable; but $\lambda x.\Omega$ is in weak head normal form, hence should be distinguished from Ω in our "lazy" theory.

We now turn to a second point in which the standard theory is not completely satisfactory.

Is the λ-calculus a programming language?

In the standard theory, the λ-calculus may be regarded as being characterised by the type equation

$$D = [D \to D]$$

(for justification of this in a general categorical framework, see e.g. [Sco80b], [Koy82, LS86]).

It is one of the most remarkable features of the various categories of domains used in denotational semantics that they admit non-trivial solutions of this equation. However, there is no *canonical* solution in any of these categories (in particular, the initial solution is trivial – the one-point domain).

I regard this as a symptom of the fact that the pure λ-calculus in the standard theory *is not a programming language*. Of course, this is to some extent a matter of terminology, but I feel that the expression "programming language" should be reserved for a formalism with a definite computational interpretation (an operational semantics). The pure λ-calculus as ordinarily conceived is too schematic to qualify.

A further indication of the same point is that studies such as Plotkin's "LCF Considered as a Programming Language" [Plo77] have not been carried over to the pure λ-calculus, for lack of any convincing way of doing do in the standard theory. This in turn impedes the development of a theory which integrates the λ-calculus with concurrency and other computational notions.

We shall see that by contrast with this situation, the lazy λ-calculus we shall develop does have a canonical model; that Plotkin's ideas can be carried over to it in a very natural way; and that the theory we shall develop will run quite strikingly in parallel with our treatment of concurrency in the previous Chapter.

The plan of the remainder of the Chapter is as follows. In the next section, we introduce the intuitions on which our theory is based, in the concrete setting of λ-terms. We then set up the axiomatic framework for our theory, based on the notion of *applicative transition systems*. This forms a bridge both to the standard theory, and to concurrency and other computational notions. Just as in Chapter 4, we introduce a domain equation for applicative transition systems, and the corresponding domain logic. We prove Duality, Characterisation, and Final Algebra theorems.

We then show how the ideas of [Plo77] can be formulated in our setting. Two distinctive features of our approach are:

- the axiomatic treatment of concepts and results usually presented concretely in work on programming language semantics

- the use of our domain logic as a tool in studying the equational theory over our "programs" (λ-terms).

Our results can also be interpreted as settling a number of questions and conjectures concerning the Domain Interpretation of Martin-Lof's Intuitionistic Type Theory raised at the 1983 Chalmers University Workshop on Semantics of Programming Languages [DNPS83].

Finally, we consider some extensions and variations of the theory.

6.2 The Lazy Lambda-Calculus

We begin with the syntax, which is standard.

Definition 6.2.1 We assume a set Var of variables, ranged over by x, y, z. The set $\mathbf{\Lambda}$ of λ-terms, ranged over by M, N, P, Q, R is defined by

$$M \quad ::= \quad x \mid \lambda x.M \mid MN.$$

For standard notions of free and bound variables etc. we refer to [Bar84]. The reader should also refer to that work for definitions of notation such as: $\mathsf{FV}(M)$, $C[\cdot]$, Λ^0. Our one point of difference concerns substitution; we write $M[N/x]$ rather than $M[x := N]$.

Definition 6.2.2 The relation $M{\Downarrow}N$ ("M converges to principal weak head normal form N") is defined inductively over Λ^0 as follows:

- $\lambda x.M {\Downarrow} \lambda x.M$
- $\dfrac{M{\Downarrow}\lambda x.P \quad P[N/x]{\Downarrow}Q}{MN{\Downarrow}Q}$

Notation

$$M{\Downarrow} \quad \equiv \quad \exists N.M{\Downarrow}N \quad (\text{“}M\text{ converges”})$$

$$M{\Uparrow} \quad \equiv \quad \neg(M{\Downarrow}) \qquad (\text{“}M\text{ diverges”})$$

It is clear that $\Downarrow$ is a partial function, i.e. evaluation is deterministic.

We now have an (unlabelled) transition system $(\Lambda^0, \_{\Downarrow}\_)$. The relation $\Downarrow$ by itself is too "shallow" to yield information about the behaviour of a term under all experiments. However, just as in the study of concurrency, we shall use it as a building block for a deeper relation, which we shall call *applicative bisimulation*. To motivate this relation, let us spell out the observational scenario we have in mind.

Given a closed term M, the only experiment of depth 1 we can do is to evaluate M and see if it converges to some abstraction (weak head normal form) $\lambda x.M_1$. If it does so, we can continue the experiment to depth 2 by supplying a term N_1 as input to M_1, and so on. Note that what the experimenter can observe at each stage is only the *fact* of convergence, not which term lies under the abstraction. We can picture matters thus:

$$
\begin{array}{ll}
\text{Stage 1 of experiment:} & M{\Downarrow}\lambda x.M_1; \\
& \text{environment "consumes" } \lambda, \\
& \text{produces } N_1 \text{ as input} \\
\text{Stage 2 of experiment:} & M_1[N_1/x]{\Downarrow}\ldots \\
\vdots &
\end{array}
$$

Definition 6.2.3 (Applicative Bisimulation) We define a sequence of relations $\{\lesssim_k\}_{k\in\omega}$ on Λ^0:

$$M \lesssim_0 N \quad \text{always}$$

$$
\begin{array}{lllll}
M \lesssim_{k+1} n & \iff & M{\Downarrow}\lambda x.M_1 & \Rightarrow & \exists N_1.\, N{\Downarrow}\lambda y.N_1 \ \&\ \forall P \in \Lambda^0. \\
& & & & M_1[P/x] \lesssim_k N_1[P/x]
\end{array}
$$

$$M \lesssim^B N \ \equiv\ \forall k \in \omega.\, M \lesssim_k N$$

Clearly each $\lesssim_k$ and $\lesssim^B$ is a preorder. We extend $\lesssim^B$ to Λ by:

$$M \lesssim^B N \ \equiv\ \forall \sigma : \mathsf{Var} \to \Lambda^0.\, M\sigma \lesssim^B N\sigma$$

(where e.g. $M\sigma$ means the result of substituting σx for each $x \in FV(M)$ in M). Finally,

$$M \sim^B N \ \equiv\ M \lesssim^B N \ \&\ N \lesssim^B M.$$

Analogously to our treatment of bisimulation in the previous Chapter, $\lesssim^B$ can be shown to be the maximal fixpoint of a certain function, and hence to satisfy:

$$
\begin{array}{lllll}
M \lesssim^B N & \iff & M{\Downarrow}\lambda x.M_1 & \Rightarrow & \exists N_1.\, N{\Downarrow}\lambda y.N_1 \ \&\ \forall P \in \Lambda^0. \\
& & & & M_1[P/x] \lesssim^B N_1[P/y]
\end{array}
$$

Further details are given in the next section.

The applicative bisimulation relation can be dexcribed in a more traditional way (from the point of view of λ-calculus) as a "Morris-style contextual congruence" [Mor68, Plo77, Mil77, Bar84].

Definition 6.2.4 The relation $\lesssim^C$ on Λ^0 is defined by

$$M \lesssim^C N \quad \equiv \quad \forall C[\cdot] \in \Lambda^0 .\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow .$$

This is extended to Λ in the same way as $\lesssim^B$.

Proposition 6.2.5 $\lesssim^B = \lesssim^C$.

This is a special case of a result we will prove later. Our proof will make essential use of domain logic, despite the fact that the *statement* of the result does not mention domains at all. The reader who may be sceptical of our approach is invited to attempt a direct proof.

We now list some basic properties of the relation $\lesssim^B$ (superscript omitted).

Proposition 6.2.6 *For all* $M, N, P \in \Lambda$*:*

$$\begin{array}{rl}
(i) & M \lesssim M \\
(ii) & M \lesssim N \;\&\; N \lesssim P \quad \Rightarrow \quad M \lesssim P \\
(iii) & M \lesssim N \quad \Rightarrow \quad M[P/x] \lesssim N[P/x] \\
(iv) & M \lesssim N \quad \Rightarrow \quad P[M/x] \lesssim P[N/x] \\
(v) & \lambda x.M \sim \lambda y.M[y/x] \\
(vi) & M \lesssim N \quad \Rightarrow \quad \lambda x.M \lesssim \lambda x.N \\
(vii) & M_i \lesssim N_i \; (i = 1, 2) \quad \Rightarrow \quad M_1 M_2 \lesssim N_1 N_2 .
\end{array}$$

Proof. (i)–(iii) and (v)–(vi) are trivial; (vii) follows from (ii) and (iv), since taking $C_1 \equiv [\cdot]M_2$, $M_1M_2 \lesssim N_1M_2$, and taking $C_2 \equiv N_1[\cdot]$, $N_1M_2 \lesssim N_1N_2$, whence $M_1M_2 \lesssim N_1N_2$. It remains to prove (iv), which by 2.5 is equivalent to

$$M \lesssim^C N \quad \Rightarrow \quad P[M/x] \lesssim^C P[N/x].$$

We rename all bound variables in P to avoid clashes with M and N, and replace x by $[\cdot]$ to obtain a context $P[\cdot]$ such that

$$P[M/x] = P[M], \qquad P[N/x] = P[N].$$

Now let $C[\cdot] \in \Lambda^0$ and $\sigma \in \mathsf{Var} \to \Lambda^0$ be given. Let $C_1[\cdot] \equiv C[P[\cdot]\sigma]$. $M \lesssim^C N$ implies

$$C_1[M\sigma]\Downarrow \;\;\Rightarrow\;\; C_1[N\sigma]\Downarrow$$

which, since $(P[M/x])\sigma = (P[\cdot]\sigma)[M\sigma]$, yields

$$C[(P[M/x])\sigma]\Downarrow \;\;\Rightarrow\;\; C[(P[N/x])\sigma]\Downarrow,$$

as required. ∎

This Proposition can be summarised as saying that $\lesssim^B$ is a *precongruence*. We thus have an (in)equational theory $\lambda\ell = (\Lambda, \sqsubseteq, =)$, where:

$$\lambda\ell \;\vdash\; M \sqsubseteq N \quad\equiv\quad M \lesssim^B N$$

$$\lambda\ell \;\vdash\; M = N \quad\equiv\quad M \sim^B N.$$

What does this theory look like?

Proposition 6.2.7 *(i) The theory* λ *[Bar84] is included in* $\lambda\ell$*; in particular,*

$$\lambda\ell \;\vdash\; (\lambda x.M)N = M[N/x] \quad (\beta).$$

(ii) $\mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$ *is a least element for* $\sqsubseteq$*, i.e.*

$$\lambda\ell \;\vdash\; \mathbf{\Omega} \sqsubseteq x.$$

(iii) (η) *is not valid in* $\lambda\ell$*, e.g.*

$$\lambda\ell \;\not\vdash\; \lambda x.\mathbf{\Omega}x = \mathbf{\Omega},$$

but we do have the following conditional version of η*:*

$$(\Downarrow\eta) \;\; \lambda\ell \;\vdash\; \lambda x.Mx = M \quad (M\Downarrow,\; x \notin FV(M))$$

$$(M\Downarrow \;\equiv\; \forall \sigma \in \mathsf{Var} \to \Lambda^0.\,(M\sigma)\Downarrow).$$

(iv) $\mathbf{YK}$ *is a greatest element for* $\sqsubseteq$*, i.e.*

$$\lambda\ell \;\vdash\; x \sqsubseteq \mathbf{YK}.$$

Proof. *(i)* is an easy consequence of 6.2.6.
(ii). $\mathbf{\Omega}\Uparrow$, hence $\mathbf{\Omega} \lesssim^B M$ for all $M \in \Lambda^0$.
(iii). $\lambda x.\mathbf{\Omega}x \not\lesssim_1 \mathbf{\Omega}$, since $(\lambda x.\mathbf{\Omega}x)\Downarrow$. Now suppose $M\Downarrow$, and let $\sigma : \mathsf{Var} \rightarrow \Lambda^0$ be given. Then $(M\sigma)\Downarrow\lambda y.N$, and $(\lambda x.\mathbf{\Omega}x)\sigma\Downarrow\lambda x.\mathbf{\Omega}x$. For any $P \in \Lambda^0$,

$$\begin{aligned}(M\sigma)P\Downarrow Q \;&\Leftrightarrow\; ((M\sigma)x)[P/x]\Downarrow Q \quad \text{since } x \notin FV(M),\\ &\Leftrightarrow\; ((\lambda x.Mx)\sigma)P\Downarrow Q,\end{aligned}$$

and so $M \sim^B \lambda x.Mx$, as required.
(iv). Note that $\mathbf{YK}\Downarrow\lambda y.N$, where $N \equiv (\lambda x.\mathbf{K}(xx))(\lambda x.\mathbf{K}(xx))$, and that for all P,

$$N[P/y]\Downarrow\lambda y.N.$$

Hence for all $P_1, \ldots, P_n$ $(n \geq 0)$,

$$\mathbf{YK}P_1 \ldots P_n\Downarrow,$$

and so $M \lesssim^B \mathbf{YK}$ for all $M \in \Lambda^0$. ∎

To understand *(iv)*, we can think of $\mathbf{YK}$ as the infinite process

$$\overset{\lambda}{\circlearrowleft}$$

solving the equation

$$\xi = \lambda x.\xi.$$

This is a top element in our applicative bisimulation ordering because it converges under all finite stages of evaluation for all arguments—the experimenter can always observe convergence (or "consume an infinite λ-stream").

We can make some connections between the theory $\lambda\ell$ and [Lon83], as pointed out to me by Luke Ong. Firstly, 6.2.7(ii) can be generalised to:

- The set of terms in Λ^0 which are least in $\lambda\ell$ are exactly the PO_0 terms in the terminology of [Lon83].

Moreover, $\mathbf{YK}$ is an O_∞ term in the terminology of [Lon83], although it is *not* a greatest element in the ordering proposed there.

6.3 Applicative Transition Systems

The theory $\lambda\ell$ defined in the previous section was derived from a particular operational model, the transition system $(\Lambda^0, \Downarrow)$. What is the general concept of which this is an example?

Definition 6.3.1 A *quasi-applicative transition system* is a structure (A, ev) where

$$ev : A \rightharpoonup (A \rightarrow A).$$

Notations:

$$\begin{array}{rrcl} (i) & a{\Downarrow}f & \equiv & a \in \mathsf{dom}\, ev \;\&\; ev(a) = f \\ (ii) & a{\Downarrow} & \equiv & a \in \mathsf{dom}\, ev \\ (iii) & a{\Uparrow} & \equiv & a \notin \mathsf{dom}\, ev \end{array}$$

Definition 6.3.2 (Applicative Bisimulation) Let (A, ev) be a quasi-ats. We define

$$F : Rel(A) \rightarrow Rel(A)$$

by

$$F(R) = \{(a, b) : a{\Downarrow}f \implies b{\Downarrow}g \;\&\; \forall c \in A.\, f(c)Rg(c)\}.$$

Then $R \in Rel(A)$ is an *applicative bisimulation* iff $R \subseteq F(R)$; and $\lesssim^B \in Rel(A)$ is defined by

$$a \lesssim^B b \;\equiv\; aRb \text{ for some applicative bisimulation } R.$$

Thus $\lesssim^B = \bigcup\{R \in Rel(A) : R \subseteq F(R)\}$, and hence is the maximal fixpoint of the monotone function F. Since the relation $\Downarrow$ is a partial function, it is easily shown that the closure ordinal of F is $\leq \omega$, and we can thus describe $\lesssim^B$ more explicitly as follows:

- $a \lesssim^B b \;\equiv\; \forall k \in \omega.\, a \lesssim_k b$
- $a \lesssim_0 b$ always

- $a \lesssim_{k+1} b \;\equiv\; a \Downarrow f \implies b \Downarrow g \;\&\; \forall c \in A.\, f(c) \lesssim_k g(c)$
- $a \sim^B b \;\equiv\; a \lesssim^B b \;\&\; b \lesssim^B a.$

It is easily seen that $\lesssim^B$, and also each $\lesssim_k$, is a preorder; $\sim^B$ is therefore an equivalence.

We now come to our main definition.

Definition 6.3.3 An *applicative transition system* (ats) is a quasi-ats (A, ev) satisfying:

$$\forall a, b, c \in A.\, a \Downarrow f \;\&\; b \lesssim^B c \;\Rightarrow\; f(b) \lesssim^B f(c).$$

An ats has a well-defined quotient $(A/{\sim^B}, ev/{\sim^B})$, where

$$ev/{\sim^B}([a]) = \begin{cases} [b] \mapsto [f(b)], & a \Downarrow f \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The reader should now refresh her memory of such notions as *applicative structure, combinatory algebra* and *lambda model* from [Bar84, Chapter 5].

Definition 6.3.4 A *quasi-applicative structure with divergence* is a structure $(A, \cdot\, , \Uparrow)$ such that $(A, \cdot\,)$ is an applicative structure, and $\Uparrow \subseteq A$ is a divergence predicate satisfying

$$x \Uparrow \implies (x \cdot y) \Uparrow.$$

Given $(A, \cdot\, , \Uparrow)$, we can define

$$a \lesssim^A b \;\equiv\; a \Downarrow \implies b \Downarrow \;\&\; \forall c \in A.\, a \cdot c \lesssim^A b \cdot c$$

as the maximal fixpoint of a monotone function along identical lines to 6.3.2.

Applicative transition systems and applicative structures with divergence are not quite equivalent, but are sufficiently so for our purposes:

Proposition 6.3.5 *Given an ats $\mathcal{B} = (A, ev)$, we define $\mathcal{A} = (A, \cdot\, , \Uparrow)$ by*

$$a \cdot b \;\equiv\; \begin{cases} a, & a \Uparrow \\ f(b) & a \Downarrow f. \end{cases}$$

Then

$$a \lesssim^A b \iff a \lesssim^B b,$$

and moreover we can recover $\mathcal{B}$ from $\mathcal{A}$ by

$$ev(a) = \begin{cases} b \mapsto a \cdot b, & a\Downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Furthermore, $\cdot$ is compatible with $\lesssim^B$, i.e.

$$a_i \lesssim^B b_i \ (i = 1, 2) \ \Rightarrow \ a_1 \cdot a_2 \lesssim^B b_1 \cdot b_2. \ \blacksquare$$

We now turn to a language for talking about these structures.

Definition 6.3.6 We assume a fixed set of variables Var. Given an applicative structure $\mathcal{A} = (A, \cdot\ )$, we define $CL(\mathcal{A})$, the *combinatory terms over* $\mathcal{A}$, by

- $\mathsf{Var} \subseteq CL(\mathcal{A})$
- $\{c_a : a \in A\} \subseteq CL(\mathcal{A})$
- $M, N \in CL(\mathcal{A}) \ \Rightarrow \ MN \in CL(\mathcal{A})$.

Let $Env(\mathcal{A}) \equiv \mathsf{Var} \to A$. Then the *interpretation function*

$$[\![]\!]^{\mathcal{A}} : CL(\mathcal{A}) \to Env(\mathcal{A}) \to A$$

is defined by:

$$\begin{array}{rcl} [\![x]\!]^{\mathcal{A}}_{\rho} & = & \rho x \\ [\![c_a]\!]^{\mathcal{A}}_{\rho} & = & a \\ [\![MN]\!]^{\mathcal{A}}_{\rho} & = & ([\![M]\!]^{\mathcal{A}}_{\rho}) \cdot ([\![N]\!]^{\mathcal{A}}_{\rho}). \end{array}$$

Given an ats $\mathcal{A} = (A, ev)$, with derived applicative structure $(A, \cdot\ )$, the satisfaction relation between $\mathcal{A}$ and atomic formulae over $CL(\mathcal{A})$, of the forms

$$M \sqsubseteq N, \ \ M = N, \ \ M\Downarrow \ \ M\Uparrow$$

is defined by:

$$
\begin{array}{rcl}
\mathcal{A}, \rho \models M \sqsubseteq N & \equiv & [\![M]\!]_{\rho}^{\mathcal{A}} \lesssim^{B} [\![N]\!]_{\rho}^{\mathcal{A}} \\
\mathcal{A}, \rho \models M = N & \equiv & [\![M]\!]_{\rho}^{\mathcal{A}} \sim^{B} [\![N]\!]_{\rho}^{\mathcal{A}} \\
\mathcal{A}, \rho \models M\Downarrow & \equiv & [\![M]\!]_{\rho}^{\mathcal{A}}\Downarrow \\
\mathcal{A}, \rho \models M\Uparrow & \equiv & [\![M]\!]_{\rho}^{\mathcal{A}}\Uparrow
\end{array}
$$

while

$$\mathcal{A} \models \phi \ \equiv \ \forall \rho \in Env(\mathcal{A}).\, \mathcal{A}, \rho \models \phi.$$

This is extended to first-order formulae in the usual way.

Note that equality in $CL(\mathcal{A})$ is being interpreted by bisimulation in $\mathcal{A}$. We could have retained the standard notion of interpretation as in [Bar84] by working in the quotient structure $(A/{\sim^B}, \cdot\ /{\sim^B})$. This is equivalent, in the sense that the same sentences are satisfied.

Definition 6.3.7 A *lambda transition system* (lts) is a structure (A, ev, k, s), where:

- (A, ev) is an ats
- $k, s \in A$, and A satisfies the following axioms (writing **K, S** for c_k, c_s):
  - $\mathbf{K}\Downarrow, \quad \mathbf{K}x\Downarrow$
  - $\mathbf{K}xy = x$
  - $\mathbf{S}\Downarrow, \quad \mathbf{S}x\Downarrow, \quad \mathbf{S}xy\Downarrow$
  - $\mathbf{S}xyz = (xz)(yz)$

We now check that these definitions do indeed capture our original example.

Example

We define $\ell = (\Lambda^0, ev)$, where

$$ev(M) = \begin{cases} P \mapsto N[P/x], & M \Downarrow \lambda x.N \\ \text{undefined} & \text{otherwise.} \end{cases}$$

ℓ is indeed an ats by 6.2.6(iv). Moreover, it is an lts via the definitions

$$k \equiv \lambda x.\lambda y.x$$

$$s \equiv \lambda x.\lambda y.\lambda z.(xz)(yz).$$

We now see how to interpret λ-terms in any lts.

Definition 6.3.8 Given an lts $\mathcal{A}$, we define $\Lambda(\mathcal{A})$, the λ-terms over $\mathcal{A}$, by the same clauses as for $CL(\mathcal{A})$, plus the additional one:

- $x \in \mathsf{Var}, M \in \Lambda(\mathcal{A}) \Rightarrow \lambda x.M \in \Lambda(\mathcal{A})$.

We define a translation

$$(\cdot)_{CL} : \Lambda(\mathcal{A}) \to CL(\mathcal{A})$$

by

$$\begin{aligned} (x)_{CL} &\equiv x \\ (c_a)_{CL} &\equiv c_a \\ (MN)_{CL} &\equiv (M)_{CL}(N)_{CL} \\ (\lambda x.M)_{CL} &\equiv \lambda^* x.(M)_{CL} \end{aligned}$$

where

$$\begin{aligned} \lambda^* x.x &\equiv \mathbf{I}\ (\equiv \mathbf{SKK}) \\ \lambda^* x.M &\equiv \mathbf{K}M \quad (x \notin FV(M)) \\ \lambda^* x.MN &\equiv \mathbf{S}(\lambda^* x.M)(\lambda^* x.N). \end{aligned}$$

We now extend $[\![\cdot]\!]$ to $\Lambda(\mathcal{A})$ by:

$$[\![M]\!]^{\mathcal{A}}_{\rho} \equiv [\![(M)_{CL}]\!]^{\mathcal{A}}_{\rho}.$$

Definition 6.3.9 We define two sets of formulae over Λ:

- *Atomic formulae:*

$$\mathsf{AF} \equiv \{M \sqsubseteq N,\ M = N,\ M\Uparrow,\ N\Uparrow \mid M, N \in \Lambda\}$$

- *Conditional formulae:*

$$\mathsf{CF} \equiv \{\bigwedge_{i\in I} M_i\Downarrow \wedge \bigwedge_{j\in J} N_j\Uparrow \Rightarrow F : F \in \mathsf{AF}, M_i, N_i \in \Lambda, \\ I, J \text{ finite}\}$$

Note that, taking $I = J = \varnothing$, $\mathsf{AF} \subseteq \mathsf{CF}$. Now given an lts $\mathcal{A}$, $\Im(\mathcal{A})$, the *theory* of $\mathcal{A}$, is defined by

$$\Im(\mathcal{A}) \equiv \{C \in \mathsf{CF} : \mathcal{A} \models C\}.$$

We also write $\Im^0(\mathcal{A})$ for the restriction of $\Im(\mathcal{A})$ to closed formulae; and given a set Con of constants and an interpretation $\mathsf{Con} \to A$, we write $\Im(\mathcal{A}, \mathsf{Con})$ for the theory of conditional formulae built from terms in $\Lambda(\mathsf{Con})$.

Example (continued). We set $\lambda\ell = \Im(\ell)$. This is consistent with our usage in the previous section. We saw there that $\lambda\ell$ satisfied much stronger properties than the simple combinatory algebra axioms in our definition of lts. It might be expected that these would fail for general lts; but this is to overlook the powerful extensionality principle built into our definition of the theory of an ats through the applicative bisimulation relation.

Proposition 6.3.10 *Let $\mathcal{A}$ be an ats. The axiom scheme of* conditional extensionality *over $CL(\mathcal{A})$:*

$$(\Downarrow\text{ext}) \quad M\Downarrow \,\&\, N\Downarrow \;\Rightarrow\; ([\forall x.Mx = Nx] \;\Rightarrow\; M = N)$$

$$(x \notin FV(M) \cup FV(N))$$

is valid in $\mathcal{A}$.

PROOF. Let $\rho \in Env(\mathcal{A})$.

$$\begin{aligned} \mathcal{A}, \rho &\models M\Downarrow \,\&\, N\Downarrow \,\&\, \forall x.\, Mx = Nx \\ &\Rightarrow \llbracket M \rrbracket^{\mathcal{A}}_{\rho}\Downarrow \,\&\, \llbracket N \rrbracket^{\mathcal{A}}_{\rho}\Downarrow \,\&\, \forall a \in A.\, \llbracket M \rrbracket^{\mathcal{A}}_{\rho} \cdot a = \llbracket N \rrbracket^{\mathcal{A}}_{\rho} \cdot a \\ &\qquad\qquad \text{since } x \notin FV(M) \cup FV(N) \\ &\Rightarrow \llbracket M \rrbracket^{\mathcal{A}}_{\rho} \sim^{A} \llbracket N \rrbracket^{\mathcal{A}}_{\rho} \\ &\Rightarrow \llbracket M \rrbracket^{\mathcal{A}}_{\rho} \sim^{B} \llbracket N \rrbracket^{\mathcal{A}}_{\rho} \\ &\Rightarrow \mathcal{A}, \rho \models M = N. \end{aligned}$$ ∎

Using this Proposition, we can now generalise most of 6.2.7 to an arbitrary lts.

Theorem 6.3.11 *Let $\mathcal{A} = (A, ev, k, s)$ be an lts. Then*
(i) $(A, ., k, s)$ is a lambda model, and hence $\lambda \subseteq \Im(\mathcal{A})$.
(ii) $\mathcal{A}$ satisfies the conditional η axiom scheme:

$$(\Downarrow\eta) \;\; M\Downarrow \;\Rightarrow\; \lambda x.Mx = M \quad (x \notin FV(M))$$

(iii) For all $M \in \Lambda^0$:

$$\lambda\ell \vdash M\Downarrow \;\;\Rightarrow\;\; \mathcal{A} \models M\Downarrow$$

(iv) $\mathcal{A} \models x \sqsubseteq \mathbf{YK}$.
(v) $\sqsubseteq$ is a precongruence in $\Im(\mathcal{A})$.

PROOF. (i). Firstly, by the very definition of lts, $\mathcal{A}$ is a combinatory algebra. We now use the following result due to Meyer and Scott, cited from [Bar84, Theorem 5.6.3, p. 117]:

- Let $\mathcal{M}$ be a combinatory algebra. Define

$$\mathbf{1} \equiv \mathbf{1}_1 \equiv \mathbf{S}(\mathbf{KI}),$$

$$\mathbf{1}_{k+1} \equiv \mathbf{S}(\mathbf{K1}_{\mathrm{k}}).$$

Then $\mathcal{M}$ is a lambda model iff it satisfies

$$\begin{array}{rl} \text{(I)} & \forall x.\, ax = bx \;\Rightarrow\; \mathbf{1}a = \mathbf{1}b \\ \text{(II)} & \mathbf{1}_2\mathbf{K} = \mathbf{K} \\ \text{(III)} & \mathbf{1}_3\mathbf{S} = \mathbf{S}. \end{array}$$

Thus it is sufficient to check that $\mathcal{A}$ satisfies (I)–(III). For (I), note firstly that $\mathcal{A} \models \mathbf{1}a{\Downarrow}x \,\&\, \mathbf{1}b{\Downarrow}$ by the convergence axioms for an lts. Hence we can apply 6.3.10 to obtain

$$\mathcal{A} \models [\forall x.\, \mathbf{1}ax = \mathbf{1}bx] \;\Rightarrow\; \mathbf{1}a = \mathbf{1}b.$$

We now assume $\forall x.\, ax = bx$ and prove $\forall x.\, \mathbf{1}ax = \mathbf{1}bx$:

$$\begin{array}{rcl} \mathbf{1}ax & = & \mathbf{S}(\mathbf{KI})ax \\ & = & (\mathbf{KI})x(ax) \\ & = & (\mathbf{KI})x(bx) \\ & = & \mathbf{S}(\mathbf{KI})bx \\ & = & \mathbf{1}bx. \end{array}$$

(II) and (III) are proved similarly.
(ii). Let $\rho \in Env(\mathcal{A})$, and assume $\mathcal{A}, \rho \models M{\Downarrow}$. We must prove that

$$\mathcal{A}, \rho \models \lambda x.Mx = M.$$

Firstly, note that for any abstraction $\lambda z.P$,

$$\mathcal{A} \models \lambda z.P{\Downarrow}$$

by the definition of $\lambda^* z.P$ and the convergence axioms for an lts. Thus since $x \notin FV(M)$, we can apply (${\Downarrow}$ext) to obtain

$$\mathcal{A}, \rho \models [\forall x.\, (\lambda x.Mx)x = Mx] \;\rightarrow\; \lambda x.Mx = M.$$

It is thus sufficient to show

$$\mathcal{A} \models (\lambda x.Mx)x = Mx.$$

But this is just an instance of (β), which $\mathcal{A}$ satisfies by (i).

(iii). We calculate:

$$\begin{aligned} \lambda\ell \vdash M\Downarrow &\Rightarrow M\Downarrow\lambda x.N \\ &\Rightarrow \lambda \vdash M = \lambda x.N \\ &\Rightarrow \mathcal{A} \models M = \lambda x.N \\ &\Rightarrow \mathcal{A} \models M\Downarrow, \end{aligned}$$

since $\mathcal{A} \models \lambda x.N\Downarrow$, as noted in (ii).
(iv). By (i) and (iii),

$$\mathcal{A} \models \mathbf{YK}\Downarrow \,\&\, \forall x.\, (\mathbf{YK})x = \mathbf{YK}.$$

Hence we can use the same argument as in 6.2.7(iv) to prove that

$$\mathcal{A} \models x \sqsubseteq \mathbf{YK}.$$

(v). This assertion amounts to the same list of properties as Proposition 6.2.6, but with respect to $\Im(\mathcal{A})$. The only difference in the proof is that 6.2.6(vii) follows immediately from 6.3.5 and the fact that $\mathcal{A}$ is an ats, and can then be used to prove 6.2.6(iv) by induction on P. ∎

Part (iii) of the Theorem tells us that all the closed terms which we expect to converge must do so in any lts. What of the converse? For example, do we have

$$\mathcal{A} \models \mathbf{\Omega}\Uparrow$$

in every lts? This is evidently not the case, since we have not imposed any axioms which require *anything* to be divergent.

Observation 6.3.12 *Let $\mathcal{A} = (A, ev)$ be an ats in which ev is* total*, i.e.* dom $ev = A$*. Then $\Im(\mathcal{A})$ is* inconsistent*, in the sense that*

$$\mathcal{A} \models x = y.$$

This is of course because the distinctions made by applicative bisimulation are based on divergence.

In the light of this observation and 6.3.11, it is natural to make the following definition in analogy with that in [Bar84]:

Definition 6.3.13 An lts $\mathcal{A}$ is *sensible* if the converse to 6.3.11(iii) holds, i.e. for all $M \in \Lambda^0$:

$$\mathcal{A} \models M\Downarrow \iff \lambda\ell \vdash M\Downarrow \iff \exists x, N.\ \lambda \vdash M = \lambda x.N.$$

(The second equivalence is justified by an appeal to the Standardisation Theorem [Bar84].)

6.4 A Domain Equation for Applicative Bisimulation

We now embark on the same programme as in the previous Chapter; to obtain a domain-theoretic analysis of our computational notions, based on a suitable domain equation. What this should be is readily elicited from the definition of ats. The structure map

$$ev : A \rightharpoonup (A \rightarrow A)$$

is *partial*; the standard approach to partial maps in domain theory (*pace* Plotkin's recent work on predomains [Plo85]) is to make them into total ones by sending undefined arguments to a "bottom" element, i.e. changing the type of ev to

$$A \rightarrow (A \rightarrow A)_{\perp}.$$

This suggests the domain equation

$$D = (D \rightarrow D)_{\perp}$$

i.e. the denotation of the type expression $\mathsf{rec}\, t.(t \rightarrow t)_{\perp}$. This equation is composed from the function space and lifting constructions. Since **SDom** is closed under these constructions, D is a Scott domain. Indeed, by the same reasoning it is an algebraic lattice. The crucial point is that this equation has a *non-trivial initial solution*, and thus there is a good candidate for a canonical model. To see this, consider the "approximants" D_k, with $D_0 \equiv \mathbf{1}$, $D_{k+1} \equiv (D_k \rightarrow D_k)_{\perp}$. Then

$$\begin{array}{rcl} D_1 & = & (\mathbf{1} \rightarrow \mathbf{1})_{\perp} \cong (\mathbf{1})_{\perp} \cong \mathbb{O} \\ D_2 & \cong & (\mathbb{O} \rightarrow \mathbb{O})_{\perp}, \quad \text{with four elements} \\ & \vdots & \end{array}$$

etc. We now unpack the structure of D. Our treatment will be rather cursory, as it proceeds along similar lines to our work in the previous Chapter. Firstly, there is an isomorphism pair

$$\mathsf{unfold} : D \rightarrow (D \rightarrow D)_{\perp},$$

$$\mathsf{fold} : (D \to D)_\perp \to D.$$

Next, we recall the categorical description of lifting, as the left adjoint to the forgetful functor

$$U : \mathbf{Dom}_\perp \to \mathbf{Dom}$$

where $\mathbf{Dom}_\perp$ is the sub-category of strict functions. Thus we have:

- A natural transformation $\mathsf{up} : I_{\mathbf{Dom}} \to U \circ (\cdot)_\perp$.
- For each continuous map $f : D \to UE$ its adjoint

$$\mathsf{lift}(f) : (D)_\perp \to_\perp E.$$

Concretely, we can take

$$\begin{array}{rcl} (D)_\perp & \equiv & \{\perp\} \cup \{<0, d> \mid d \in D\} \\ x \sqsubseteq y & \equiv & x = \perp \\ & & \text{or } x = <0, d> \; \& \; y = <0, d'> \; \& \; d \sqsubseteq_D d' \\ \mathsf{up}_D(d) & \equiv & <0, d> \\ \mathsf{lift}(f)(\perp) & \equiv & \perp_E \\ \mathsf{lift}(f)<0, d> & \equiv & f(d). \end{array}$$

We can now define

$$ev : D \rightharpoonup (D \to D)$$

by

$$ev(d) = \begin{cases} f, & \mathsf{unfold}(d) = <0, f> \\ \text{undefined} & \mathsf{unfold}(d) = \perp. \end{cases}$$

Thus (D, ev) is a quasi-ats, and we write $d \Downarrow f$, $d \Uparrow$ etc. Note that we can recover d from $ev(d)$ by

$$d = \begin{cases} \mathsf{fold}(<0, f>), & d \Downarrow f \\ \perp_D & d \Uparrow. \end{cases}$$

The final ingredient in the definition of D is initiality. The only direct consequence of this which we will use is contained in

Theorem 6.4.1 *D is internally fully abstract, i.e.*

$$\forall d, d' \in D.\, d \sqsubseteq d' \iff d \lesssim^B d'.$$

PROOF. Unpacking the definitions, we see that for all $d, d' \in D$:

$$d \sqsubseteq d' \iff d \Downarrow f \Rightarrow d' \Downarrow g \;\&\; \forall d'' \in D.\, f(d'') \sqsubseteq g(d'').$$

Thus the domain ordering is an applicative bisimulation, and so is included in $\sqsubseteq^B$. For the converse, we need some additional notions. We define d_k, f_k for $d \in D$, $f \in [D \to D]$, $k \in \omega$ by:

$$d_0 \Uparrow$$

$$d \Uparrow \Rightarrow d_k \Uparrow$$

$$d \Downarrow f \Rightarrow d_{k+1} \Downarrow f_k$$

$$f_k : d \mapsto (fd)_k.$$

We can use standard techniques to prove, from the initiality of D:

- $\forall d \in D.\, d = \bigsqcup_{k \in \omega} d_k.$

The proof is completed with a routine induction to show that:

$$\forall k \in \omega.\, d \lesssim_k d' \Rightarrow d_k \sqsubseteq d'_k. \quad \blacksquare$$

As an immediate corollary of this result, we see that D is an ats. We thus have an interpretation function

$$[\![\cdot]\!]^D : CL(D) \to Env(D) \to\!\!\to D.$$

We extend this to $\Lambda(D)$ by:

$$[\![\lambda x.M]\!]^D_\rho = \mathsf{fold}(\mathsf{up}(\lambda d \in D. [\![M]\!]^D_{\rho[x \mapsto d]})).$$

Note that the application induced from (D, ev) can be described by

$$d \cdot d' = \mathsf{lift}(Ap)\ \mathsf{unfold}(d)\ d'$$

where

$$Ap : [D \to D] \to D \to D$$

is the standard application function; and is therefore continuous. This together with standard arguments about environment semantics guarantees that our extension of $[\![]\!]^D$ is well-defined. Note also that $[\![\lambda x.M]\!]^D_\rho \neq \bot_D$, as expected.

We can now define

$$k \equiv [\![\lambda x.\lambda y.x]\!]^D_\rho,$$

$$s \equiv [\![\lambda x.\lambda y.\lambda z.(xz)(yz)]\!]^D_\rho$$

for D. It is straightforward to verify

Proposition 6.4.2 *D is an lts.* ∎

Thus far, we have merely used our domain equation to construct a particular lts D. However, its "categorical" or "absolute" nature should lead us to suspect that we can use D to study the whole class of lts. The medium we will use for this purpose is once again a suitable domain logic.

6.5 A Domain Logic for Applicative Transition Systems

Definition 6.5.1 The syntax of our domain logic $\mathcal{L}$ is defined by

$$\phi \ ::= \ t \mid \phi \wedge \psi \mid (\phi \to \psi)_\perp$$

Definition 6.5.2 (Semantics of $\mathcal{L}$) Given a quasi ats $\mathcal{A}$, we define the satisfaction relation $\models_\mathcal{A} \subseteq \mathcal{A} \times \mathcal{L}$:

$$a \models_\mathcal{A} t \text{ always}$$

$$a \models_\mathcal{A} \phi \wedge \psi \equiv a \models_\mathcal{A} \phi \,\&\, a \models_\mathcal{A} \psi$$

$$a \models_\mathcal{A} (\phi \to \psi)_\perp \equiv a{\Downarrow}f \,\&\, \forall b \in A.\, b \models_\mathcal{A} \phi \Rightarrow f(b) \models_\mathcal{A} \psi.$$

Notation:

$$\begin{aligned}
\mathcal{L}(a) &\equiv \{\phi \in \mathcal{L} : a \models_\mathcal{A} \phi\} \\
\mathcal{A} \models \phi \leq \psi &\equiv \forall a \in A.\, a \models_\mathcal{A} \phi \implies a \models_\mathcal{A} \psi \\
\mathcal{A} \models \phi = \psi &\equiv \forall a \in A.\, a \models_\mathcal{A} \phi \iff a \models_\mathcal{A} \psi \\
\models \phi \leq \psi &\equiv \forall \mathcal{A}.\, \mathcal{A} \models \phi \leq \psi \\
\lambda &\equiv (t \to t)_\perp \\
a \sqsubseteq^\mathcal{L} b &\equiv \mathcal{L}(a) \subseteq \mathcal{L}(b).
\end{aligned}$$

Note that: $\forall a \in A.\, a{\Downarrow} \iff a \models_\mathcal{A} \lambda$.

Lemma 6.5.3 *Let $\mathcal{A}$ be a quasi ats. Then*

$$\forall a, b \in A.\, a \sqsubseteq^B b \implies a \sqsubseteq^\mathcal{L} b.$$

PROOF. We assume $a \sqsubseteq^B b$ and prove $\forall \phi \in \mathcal{L}.\, a \models_\mathcal{A} \phi \Rightarrow b \models_\mathcal{A} \phi$ by induction on ϕ. The non-trivial case is $(\phi \to \psi)_\perp$.

- $a \models_\mathcal{A} (\phi \to \psi)_\perp$

$\implies a{\Downarrow}f$

$\implies b{\Downarrow}g \,\&\, \forall c.\, f(c) \sqsubseteq^B g(c)$

$\implies \forall c.\, c \models_\mathcal{A} \phi \implies f(c) \sqsubseteq^B g(c) \,\&\, f(c) \models_\mathcal{A} \psi$

$\implies \forall c.\, c \models_\mathcal{A} \phi \Rightarrow g(c) \models_\mathcal{A} \psi$ ind. hyp.

$\implies b \models_\mathcal{A} (\phi \to \psi)_\perp.$ ∎

To get a converse to this result, we need a condition on $\mathcal{A}$.

Definition 6.5.4 A quasi ats A is *approximable* iff

$$\forall a, b_1, \ldots, b_n \in A.\, ab_1 \ldots b_n\Downarrow \;\Rightarrow\; \exists \phi_1, \cdots, \phi_n.$$
$$a \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_n \to \lambda)_\perp \cdots)_\perp \;\&\; b_i \models_{\mathcal{A}} \phi_i, \;\; 1 \leq i \leq n.$$

This is a natural condition, which says that convergence of a function application is caused by some finite amount of information (observable properties) of its arguments.

As expected, we have

Theorem 6.5.5 (Characterisation Theorem) *Let $\mathcal{A}$ be an approximable quasi ats. Then*

$$\lesssim^B = \lesssim^{\mathcal{L}}.$$

Proof. By 5.3, $\lesssim^B \subseteq \lesssim^{\mathcal{L}}$. For the converse, suppose $a \not\lesssim^B b$. Then for some k, $a \not\lesssim^B_k b$, and so for some $c_1, \cdots, c_k \in A$:

$$ac_1 \cdots c_k\Downarrow \;\&\; bc_1 \cdots c_k\Uparrow.$$

By approximability, for some $\phi_1, \cdots, \phi_k \in \mathcal{L}$,

$$a \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \;\&\; b_i \models_{\mathcal{A}} \phi_i, \;\; 1 \leq i \leq k.$$

Clearly $b \not\models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$, and so $a \not\lesssim^{\mathcal{L}} b$. ∎

As a further consequence of approximability, we have:

Proposition 6.5.6 *An approximable quasi ats is an ats.*

Proof. Suppose $a\Downarrow f$ and $b \lesssim^B c$. We must show $f(b) \lesssim^B f(c)$. It is sufficient to show that for all $k \in \omega$, $d_1, \ldots, d_k \in A$:

$$f(b)d_1 \ldots d_k\Downarrow \;\;\Rightarrow\;\; f(c)d_1 \ldots d_k\Downarrow.$$

Now $f(b)d_1 \ldots d_k\Downarrow$ implies $abd_1 \ldots d_k\Downarrow$; hence by approximability, for some $\phi, \phi_1, \ldots \phi_k \in \mathcal{L}$:

$$a \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$$

and

$$b \models_{\mathcal{A}} \phi, \;\; b_i \models_{\mathcal{A}} \phi_i, \;\; 1 \leq i \leq k.$$

By 5.5, $c \models_{\mathcal{A}} \phi$, and so $abd_1 \ldots d_k \models_{\mathcal{A}} \lambda$, and $f(c)d_1 \ldots d_k\Downarrow$ as required. ∎

We now introduce a proof system for assertions of the form $\phi \leq \psi$, $\phi = \psi$ $(\phi, \psi \in \mathcal{L})$.

Proof System For $\mathcal{L}$

(REF) $\phi \leq \phi$

(TRANS) $\dfrac{\phi \leq \psi \quad \psi \leq \xi}{\phi \leq \xi}$

$(= -I)$ $\dfrac{\phi \leq \psi \quad \psi \leq \phi}{\phi = \psi}$

$(= -E)$ $\dfrac{\phi = \psi}{\phi \leq \psi \quad \psi \leq \phi}$

$(t \ - I)$ $\phi \leq t$

$(\wedge - I)$ $\dfrac{\phi \leq \phi_1 \quad \phi \leq \psi_2}{\phi \leq \phi_1 \wedge \phi_2}$

$(\wedge - E)$ $\phi \wedge \psi \leq \phi \quad \phi \wedge \psi \leq \psi$

$((\rightarrow)_\perp - \leq)$ $\dfrac{\phi_2 \leq \phi_1 \quad \psi_1 \leq \psi_2}{(\phi_1 \rightarrow \psi_1)_\perp \leq (\phi_2 \rightarrow \psi_2)_\perp}$

$((\rightarrow)_\perp - \wedge)$ $(\phi \rightarrow \psi_1 \wedge \psi_2)_\perp = (\phi \rightarrow \psi_1)_\perp \wedge (\phi \rightarrow \psi_2)_\perp$

$((\rightarrow)_\perp - t)$ $(\phi \rightarrow t)_\perp \leq (t \rightarrow t)_\perp$.

We write $\mathcal{L} \vdash A$ or just $\vdash A$ to indicate that an assertion A is derivable from these axioms and rules. Note that the converse of $((\rightarrow)_\perp - t)$ is derivable from $(t - I)$ and $((\rightarrow)_\perp - \leq)$; by abuse of notation we refer to the corresponding equation by the same name.

Theorem 6.5.7 (Soundness Theorem) $\vdash \phi \leq \psi \implies \models \phi \leq \psi$.

Proof. By a routine induction on the length of proofs. ▮

So far, our logic has been presented in a syntax-free fashion so far as the elements of the ats are concerned. Now suppose we have an lts $\mathcal{A}$. λ-terms can be interpreted in $\mathcal{A}$, and for $M \in \mathbf{\Lambda}^0$, $\rho \in Env(\mathcal{A})$, we can define:

$$M, \rho \models_{\mathcal{A}} \phi \equiv [\![M]\!]^{\mathcal{A}}_{\rho} \models_{\mathcal{A}} \phi.$$

We can extend this to arbitrary terms $M \in \mathbf{\Lambda}$ in the presence of *assumptions* $\Gamma : \mathsf{Var} \to \mathcal{L}$ on the variables:

$$M, \Gamma \models_{\mathcal{A}} \phi \quad \equiv \quad \forall \rho \in Env(\mathcal{A}).\, \rho \models_{\mathcal{A}} \Gamma \quad \Rightarrow \quad [\![M]\!]^{\mathcal{A}}_{\rho} \models_{\mathcal{A}} \phi$$

where

$$\rho \models_{\mathcal{A}} \Gamma \quad \equiv \quad \forall x \in \mathsf{Var}.\, \rho x \models_{\mathcal{A}} \Gamma x.$$

We write

$$M, \Gamma \models \phi \quad \equiv \quad \forall \mathcal{A}.\, M, \Gamma \models_{\mathcal{A}} \phi.$$

We now introduce a proof system for assertions of the form $M, \Gamma \vdash \phi$.

Proof System For Program Logic

$$(TR) \quad M, \Gamma \vdash t$$

$$(AND) \quad \frac{M, \Gamma \vdash \phi \quad M, \Gamma \vdash \psi}{M, \Gamma \vdash \phi \wedge \psi}$$

$$(LEQ) \quad \frac{\Gamma \leq \Delta \quad M, \Delta \vdash \phi \quad \phi \leq \psi}{M, \Gamma \vdash \psi}$$

$$(VAR) \quad x, \Gamma[x \mapsto \phi] \vdash \phi$$

$$(ABS) \quad \frac{M, \Gamma[x \mapsto \phi] \vdash \psi}{\lambda x.M, \Gamma \vdash (\phi \to \psi)_{\perp}}$$

$$(APP) \quad \frac{M, \Gamma \vdash (\phi \to \psi)_{\perp} \quad N, \Gamma \vdash \phi}{MN, \Gamma \vdash \psi}.$$

Theorem 6.5.8 (Soundness of Program Logic) *For all M, Γ, ϕ:*

$$M, \Gamma \vdash \phi \implies M, \Gamma \models \phi. \quad \blacksquare$$

The proof is again routine. Note the striking similarity of our program logic with type inference, in particular with the intersection type discipline and Extended Applicative Type Structures of [CDHL84]. The crucial *difference* lies in the entailment relation $\leq$, and in particular the fact that their axiom (in our notation)

$$t \leq (t \rightarrow t)_{\perp}$$

is *not* a theorem in our logic; instead, we have the weaker $((\rightarrow)_{\perp})$. This reflects a different notion of "function space"; we discuss this further in section 7.

We now come to the expected connection between the domain logic $\mathcal{L}$ and the domain D. Once again, the connecting link is the domain equation used to define D, and from which $\mathcal{L}$ is derived. Since this equation corresponds to the type expression $\sigma \equiv \mathsf{rec}\, t.(t \rightarrow t)_{\perp}$, it falls within the scope of the general theory developed in Chapter 4. The logic $\mathcal{L}$ presented in this section is a streamlined version of $\mathcal{L}(\sigma)$ as defined in Chapter 4. Once we have shown that $\mathcal{L}$ is equivalent to $\mathcal{L}(\sigma)$, we can apply the results of Chapter 4 to obtain the desired relationships between $\mathcal{L} \simeq \mathcal{L}(\sigma)$ and $D \simeq D(\sigma)$.

Firstly, note that $\mathcal{L}$ as presented contains no disjunctive structure, while the constructs $\rightarrow$, $(\cdot)_{\perp}$ appearing in σ generate no inconsistencies according to the definition of C in Chapter 4. Thus (the Lindenbaum algebra of) $\mathcal{L}_{\wedge}(\sigma)$, the purely conjunctive part of $\mathcal{L}(\sigma)$, is a meet-semilattice, and applying Theorem 2.3.4, we obtain

$$\mathsf{Spec}\ (\mathcal{L}(\sigma)/{=_{\sigma}}, \leq_{\sigma}/{=_{\sigma}}) \;\cong\; \mathsf{Filt}(\mathcal{L}_{\wedge}(\sigma)/{=_{\sigma}}, \leq_{\sigma}/{=_{\sigma}}).$$

It remains to show that $\mathcal{L}$ is pre-isomorphic to $\mathcal{L}_{\wedge}(\sigma)$. We can describe the syntax of $\mathcal{L}_{\wedge}(\sigma)$ as follows:

- $L_{\wedge}(\sigma)$:

$$\phi \;\;::=\;\; t \mid \phi \wedge \psi \mid (\phi)_{\perp}\;\; (\phi \in L(\sigma \rightarrow \sigma))$$

- $L_{\wedge}(\sigma \rightarrow \sigma)$:

$$\phi \;\;::=\;\; t \mid \phi \wedge \psi \mid (\phi \rightarrow \psi)\;\; (\phi, \psi \in L(\sigma)).$$

Using $(()_\bot - \wedge)$ and $(\to -t)$ (i.e. the nullary instances of $(\to -\wedge)$) from Chapter 4, we obtain the following normal forms for $L_\wedge(\sigma)$:

$$\phi \ ::= \ t \mid \phi \wedge \psi \mid (\phi \to \psi)_\bot.$$

In this way we see that $L \subseteq L_\wedge(\sigma)$, and that each $\phi \in L_\wedge(\sigma)$ is equivalent to one in L. Moreover, the axioms and rules of $\mathcal{L}$ are easily seen to be derivable in $\mathcal{L}_\wedge(\sigma)$. For example, $((\to)_\bot - t)$ is derivable, since

$$\mathcal{L}_\wedge(\sigma) \vdash (\phi \to \psi)_\bot = (t)_\bot = (t \to t)_\bot.$$

It remains to show the converse, i.e. that for $\phi, \psi \in \mathcal{L}$:

$$\mathcal{L}_\wedge(\sigma) \vdash \phi \leq \psi \implies \mathcal{L} \vdash \phi \leq \psi.$$

For this purpose, we use $((\to)_\bot - \wedge)$ and $((\to)_\bot - t)$ to get normal forms for $\mathcal{L}$.

Lemma 6.5.9 (Normal Forms) *Every formula in $\mathcal{L}$ is equivalent to one in $N\mathcal{L}$, where:*

- $N\mathcal{L} = \{\bigwedge_{i \in I} \phi_i : I \text{ finite}, \ \phi_i \in SN\mathcal{L}, \ i \in I\}$
- $SN\mathcal{L} = \{(\phi_1 \to \cdots (\phi_k \to \lambda)_\bot \cdots)_\bot : k \geq 0, \phi_i \in N\mathcal{L}, \ 1 \leq i \leq k\}$. ∎

Now by the semantic arguments of Chapter 3, we have

Lemma 6.5.10 *For ϕ, ψ with*

$$\phi \ \equiv \ \bigwedge_{i \in I} (\phi_i \to \phi'_i)_\bot,$$

$$\psi \ \equiv \ \bigwedge_{j \in J} (\psi_j \to \psi'_j)_\bot :$$

$$\mathcal{L}(\sigma) \vdash \phi \leq \psi \iff \forall j \in J.\, \mathcal{L}(\sigma) \vdash \bigwedge\{\phi'_i \ : \ \mathcal{L}(\sigma) \vdash \psi_j \leq \phi_i\} \leq \psi'_j.$$

Proposition 6.5.11 *For $\phi, \psi \in N\mathcal{L}$, if $\mathcal{L}(\sigma) \vdash \phi \leq \psi$ then there is a proof of $\phi \leq \psi$ using only the meet-semilattice laws and the derived rule $((\to)_\bot)$.*

PROOF. By induction on the complexity of ϕ and ψ, and the preceding Lemma. ∎

We have thus shown that

$$\mathcal{L}(\sigma) \;\cong\; \mathcal{L}_{\wedge}(\sigma) \;\cong\; \mathcal{L},$$

and we can apply the Duality Theorem of Chapter 4 to obtain

Theorem 6.5.12 (Stone Duality) *$\mathcal{L}$ is the Stone dual of $\mathcal{D}$:*

$$(i) \quad \mathcal{D} \;\cong\; \mathsf{Filt}\,\mathcal{L}$$

$$(ii) \quad (K(\mathcal{D}))^{op} \;\cong\; (L/{=}, \leq/{=}).$$

Corollary 6.5.13 $\mathcal{D} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi.$

We can now deal with the program logic over λ-terms in a similar fashion. The denotational semantics for $\mathbf{\Lambda}$ in $\mathcal{D}$ given in the precious section can be used to define a translation map

$$(\cdot)^* : \mathbf{\Lambda} \to \mathbf{\Lambda}(\sigma).$$

The logic presented in this section is equivalent to the endogenous logic of Chapter 4 in the sense that

$$M, \Gamma \vdash \phi \iff M^*, \Gamma \vdash \phi$$

where $M \in \mathbf{\Lambda}$, $\Gamma : \mathsf{Var} \to L$, $\phi \in L \subseteq L(\sigma)$. We omit the details, which by now should be routine. As a consequence of this result, we can apply the Completeness Theorem for Endogenous Logic from Chapter 4, to obtain:

Theorem 6.5.14 *$\mathcal{D}$ is $\mathcal{L}$-complete, i.e. for all $M \in \mathbf{\Lambda}$, $\Gamma : \mathsf{Var} \to L$, $\phi \in L \subseteq L(\sigma)$:*

$$M, \Gamma \vdash \phi \iff M, \Gamma \models_{\mathcal{L}} \phi.$$

In the previous section, we defined an lts over $\mathcal{D}$; and we have now shown that $\mathcal{D}$ is isomorphic to $\mathsf{Filt}\,\mathcal{L}$. We can in fact describe the lts structure over $\mathsf{Filt}\,\mathcal{L}$ directly; and this will show how $\mathcal{D}$, defined by a domain equation reminiscent of the D_∞ construction, can also be viewed as a graph model or "PSE algebra" in the terminology of [Lon83].

Notation. For $X \subseteq L$, $X^\dagger$ is the filter generated by X. This can be defined inductively by:

- $X \subseteq X^\dagger$
- $t \in X^\dagger$
- $\phi, \psi \in X^\dagger \;\Rightarrow\; \phi \wedge \psi \in X^\dagger$
- $\phi \in X^\dagger,\ \mathcal{L} \vdash \phi \leq \psi \;\;\Rightarrow\;\; \psi \in X^\dagger$.

Definition 6.5.15 The quasi-applicative structure with divergence

$$(\mathsf{Filt}\,\mathcal{L}, \cdot\ , \Uparrow)$$

is defined as follows:

- $x\Uparrow \;\;\equiv\;\; x = \{t\}$
- $x \cdot y \;\;\equiv\;\; \{\psi : \exists\phi.\,(\phi \to \psi)_\perp \in x \,\&\, \phi \in y\} \cup \{t\}.$

It is easily verified that in this structure

$$x \lesssim^B y \iff x \subseteq y,$$

and hence that application is monotone in each argument, and $\mathsf{Filt}\,\mathcal{L}$ is an ats. Thus we have an interpretation function

$$[\![\cdot]\!]^{\mathsf{Filt}\,\mathcal{L}} : CL(\mathsf{Filt}\,\mathcal{L}) \to Env(\mathsf{Filt}\,\mathcal{L}) \to \mathsf{Filt}\,\mathcal{L}$$

which is extended to $\mathbf{\Lambda}(\mathsf{Filt}\,\mathcal{L})$ by

$$[\![\lambda x.M]\!]^{\mathsf{Filt}\,\mathcal{L}}_\rho = \{(\phi \to \psi)_\perp : \psi \in [\![M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\rho[x \mapsto \uparrow\psi]}\}^\dagger.$$

We then define

Definition 6.5.16

$$\begin{array}{lcl} s & \equiv & [\![\lambda x.\lambda y.\lambda z.(xz)(yz)]\!]^{\mathsf{Filt}\,\mathcal{L}} \\ k & \equiv & [\![\lambda x.\lambda y.x]\!]^{\mathsf{Filt}\,\mathcal{L}}. \end{array}$$

Proposition 6.5.17 $\mathsf{Filt}\,\mathcal{L}$ *is an lts. Moreover,* $\mathsf{Filt}\,\mathcal{L}$ *and* $\mathcal{D}$ *are isomorphic as combinatory algebras.*

PROOF. It is sufficient to show that the isomorphism of the Duality Theorem preserves application, divergence and the denotation of λ-terms, since it then preserves s and k and so is a combinatory isomorphism, and $\mathsf{Filt}\,\mathcal{L}$ is an lts, since $\mathcal{D}$ is.

Firstly, we show that application is preserved, i.e. for $d_1, d_2 \in \mathcal{D}$:

$$(\star)\ \ \mathcal{L}(d_1 \cdot\, d_2) = \mathcal{L}(d_1)\cdot\, \mathcal{L}(d_2)$$

The right to left inclusion follows by the same argument as the soundness of (APP) in 6.5.7. For the converse, suppose $\psi \in \mathcal{L}(d_1 \cdot\, d_2)$, $\mathcal{L}\ \nvdash\ \psi = t$. By the Duality Theorem, each ψ in $\mathcal{L}$ corresponds to a unique $c \in K(\mathcal{D}$ with $\mathcal{L}(c) = \uparrow\psi$. Since application is continuous in $\mathcal{D}$, $c \sqsubseteq d_1 \cdot\, d_2$, $c \neq \bot$ implies that for some $b \in K(\mathcal{D})$, $\mathsf{fold}(<0, [b, c]>) \sqsubseteq d_1$ and $b \sqsubseteq d_2$. Let $\mathcal{L}(b) = \uparrow\phi$, then $(\phi \to \psi)_\bot \in \mathcal{L}(d_1)$ and $\phi \in \mathcal{L}(d_2)$, as required.

Next, we show that denotations of λ-terms are preserved, i.e. for all $M \in \mathbf{\Lambda}$, $\rho \in Env(\mathcal{D})$:

$$(\star\star)\ \ \mathcal{L}([\![M]\!]^{\mathcal{D}}_{\rho}) = [\![M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\mathcal{L}\circ\rho}.$$

This is proved by induction on M. The case when M is a variable is trivial; the case for application uses $(\star)$. For abstraction, we argue by structural induction over $\mathcal{L}$. We show the non-trivial case. Let ϕ, b be paired in the isomorphism of the Duality Theorem. Then

$$\begin{array}{lll} & \lambda x.M,\, \rho\ \models_{\mathcal{D}}\ (\phi \to \psi)_\bot & \\ \iff & M,\, \rho[x \mapsto b]\ \models_{\mathcal{D}}\ \psi & \\ \iff & M,\, \mathcal{L}() \circ (\rho[x \mapsto b])\ \models_{\mathsf{Filt}\,\mathcal{L}}\ \psi & \text{ind. hyp.} \\ \iff & M,\, (\mathcal{L}() \circ \rho)[x \mapsto \uparrow\phi]\ \models_{\mathsf{Filt}\,\mathcal{L}}\ \psi & \\ \iff & \lambda x.M,\, \mathcal{L}() \circ \rho\ \models_{\mathsf{Filt}\,\mathcal{L}}\ (\phi \to \psi)_\bot. & \end{array}$$

Finally, divergence is trivially preserved, since the only divergent elements in $\mathcal{D}$, $\mathsf{Filt}\,\mathcal{L}$ are $\bot$, $\{t\}$, are these are in bi-unique correspondence under the isomorphism of the Duality Theorem. ∎

We can now proceed in exact analogy to Chapter 5, and use Stone Duality to convert the Characterisation Theorem into a Final Algebra Theorem.

Definition 6.5.18 We define a number of categories of transition systems:

ATS Objects: applicative transition systems; morphisms $\mathcal{A} \to \mathcal{B}$: maps $f : A \to B$ satisfying

$$a \models_{\mathcal{A}} \phi \iff f(a) \models_{\mathcal{B}} \phi.$$

LTS The subcategory of **ATS** of lts and morphisms which preserve application, s and k.

CLTS The full subcategory of **LTS** of those $\mathcal{A}$ satisfying *continuity*:

$$\psi \neq t,\ ab \models_{\mathcal{A}} \psi \implies \exists \phi.\, a \models_{\mathcal{A}} (\phi \to \psi)_{\perp} \,\&\, b \models_{\mathcal{A}} \phi,$$

and also

$$\mathcal{L}(s) = [\![s]\!]^{\mathsf{Filt}\,\mathcal{L}},\ \ \mathcal{L}(k) = [\![k]\!]^{\mathsf{Filt}\,\mathcal{L}}.$$

Note that continuity implies approximability.

Theorem 6.5.19 (Final Algebra) *(i) $\mathcal{D}$ is final in* **ATS**.
(ii) Let $\mathcal{A}$ be an approximable lts. The map

$$\mathsf{t}_{\mathcal{A}} : \mathcal{A} \to \mathcal{D}$$

from (i) is an **LTS** *morphism iff $\mathcal{A}$ is continuous.*
(iii) $\mathcal{D}$ is final in **CLTS**.

Proof. (i). Given $\mathcal{A}$ in **ATS**, define

$$\mathsf{t}_{\mathcal{A}} : \mathcal{A} \to \mathcal{D}$$

by

$$\mathsf{t}_{\mathcal{A}} \ \equiv\ \mathcal{A} \stackrel{\mathcal{L}()}{\to} \mathsf{Filt}\,\mathcal{L} \stackrel{\eta}{\to} \mathcal{D}$$

where η is the isomorphism from the Stone Duality Theorem. For $a \in A$,

$$\mathcal{L}(a) = \mathcal{L} \circ \eta \circ \mathcal{L}(a) = \mathcal{L} \circ \mathsf{t}_{\mathcal{A}}(a),$$

and so $\mathsf{t}_{\mathcal{A}}$ is an **ATS** morphism; moreover, it is unique, since for $d, d' \in D$:

$$\mathcal{L}(d) = \mathcal{L}(d') \ \Rightarrow\ \mathcal{K}(d) = \mathcal{K}(d') \ \Rightarrow\ d = d'.$$

(ii). That $\mathcal{L}()$ is a combinatory morphism iff $\mathcal{A}$ is in **CLTS** is an immediate consequence of the definitions; the result then follows from the fact that η is a combinatory isomorphism.
(iii). Immediate from (ii). ∎

Note that if $\mathcal{A}$ is approximable, we have:

$$a \precsim^B b \iff \mathtt{t}_{\mathcal{A}}(a) \precsim^B \mathtt{t}_{\mathcal{A}}(b).$$

Thus we can regard the Final Algebra Theorem as giving a syntax-free fully abstract semantics for approximable ats. However, from the point of view of applications to programming language semantics, this is not very useful. In the next section, we shall study full abstraction in a syntax-directed framework, using our domain logic as a tool.

6.6 Lambda Transition Systems considered as Programming Languages

The classical discussion of full abstraction in the λ-calculus [Plo77, Mil77] is set in the typed λ-calculus with ground data. As remarked in the Introduction, this material has not to date been transferred successfully to the pure untyped λ-calculus. To see why this is so, let us recall some basic notions from [Plo77, Mil77].

Firstly, there is a natural notion of *program*, namely closed term of ground type. Programs either diverge, or yield a ground constant as result. This provides a natural notion of observable behaviour for programs, and hence an operational order on them. This is extended to arbitrary terms via ground contexts; in other words, the point of view is taken that only program behaviour is directly observable, and the meaning of a higher-type term lies in the observable behaviour of the programs into which it can be embedded. Thus both the presence of ground data, and the fact that terms are typed, enter into the basic definitions of the theory.

By contrast, we have a notion of atomic observation for the lazy λ-calculus in the absence of types or ground data, namely convergence to weak head normal form. This leads to the applicative bisimulation relation, and hence to a natural operational ordering. We can thus develop a theory of full abstraction in the pure untyped λ-calculus. Our results will correspond recognisably to those in [Plo77], although the technical details contain many differences. One feature of our development is that we work axiomatically with classes of lts under various hypotheses, rather than with particular languages. (Note that operational transition systems and "programming languages" such as $\lambda\ell$ actually *are* lts under our definitions.)

Definition 6.6.1 Let $\mathcal{A}$ be an lts. $\mathcal{D}$ is *fully abstract* for $\mathcal{A}$ if $\Im(\mathcal{A}) = \Im(\mathcal{D})$.

This definition is consistent with that in [Plo77, Mil77], provided we accept the applicative bisimulation ordering on $\mathcal{A}$ as the appropriate operational preorder. The argument for doing so is made highly plausible by Proposition 6.2.5, which characterises applicative bisimulation as a contextual preorder analogous to those used in [Plo77, Mil77]. We shall prove 6.2.5 later in this section.

We now turn to the question of conditions under which $\mathcal{D}$ is fully abstract for $\mathcal{A}$. As emerges from [Plo77, Mil77], this is essentially a question of definability.

Definition 6.6.2 An ats $\mathcal{A}$ is $\mathcal{L}$-*expressive* if for all $\phi \in \mathcal{L}$, for some $a \in \mathcal{A}$:

$$\mathcal{L}(a) = \uparrow\phi \;\equiv\; \{\psi \in \mathcal{L} \;:\; \mathcal{L} \vdash \phi \leq \psi\}.$$

In the light of Stone Duality, $\mathcal{L}$-expressiveness can be read as: "all finite elements of $\mathcal{D}$ are definable in $\mathcal{A}$".

Definition 6.6.3 Let $\mathcal{A}$ be an ats.

- *Convergence testing* is definable in $\mathcal{A}$ if for some $c \in A$, $\mathcal{A}$ satisfies:
  - $c\Downarrow$
  - $x\Uparrow \;\Rightarrow\; cx\Uparrow$
  - $x\Downarrow \;\Rightarrow\; cx = \mathbf{I}$.

  In this case, we use C as a constant to denote c.

- *Parallel convergence* is definable in $\mathcal{A}$ if for some $p \in A$, $\mathcal{A}$ satisfies:
  - $p\Downarrow, \;\; px\Downarrow$
  - $x\Downarrow \;\Rightarrow\; pxy\Downarrow$
  - $y\Downarrow \;\Rightarrow\; pxy\Downarrow$
  - $x\Uparrow \;\&\; y\Uparrow \;\Rightarrow\; pxy\Uparrow$.

  In this case, we use P to denote such a p.

Note that if C is definable, it is unique (up to bisimulation); this is not so for P.

The notion of parallel convergence is reminiscent of Plotkin's parallel or, and will play a similar role in our theory. (A sharper comparison will be made later in this section.) The notion of convergence testing is less expected. We can think of the combinator C as a sort of "1-strict" version of $\mathbf{K}$:

$$\mathsf{C}xy = \mathbf{K}xy = y \quad \text{if } x\Downarrow$$

$$\mathsf{C}xy\Uparrow \quad \text{if } x\Uparrow.$$

This 1-strictness allows us to test, sequentially, a number of expressions for convergence. Under the hypothesis that C is definable, we can give a very satisfactory picture of the relationship between all these notions.

Theorem 6.6.4 (Full Abstraction) *Let $\mathcal{A}$ be a sensible, approximable lts in which C is definable. The following conditions are equivalent:*

(i) *Parallel convergence is definable in $\mathcal{A}$.*

(ii) *$\mathcal{A}$ is $\mathcal{L}$-expressive.*

(iii) *$\mathcal{A}$ is $\mathcal{L}$-complete.*

(iv) *$\mathsf{t}_{\mathcal{A}}$ is a combinatory embedding with $K(\mathcal{D}) \subseteq Im\, \mathsf{t}_{\mathcal{A}}$.*

(v) *$\mathcal{D}$ is fully abstract for $\mathcal{A}$.*

Proof. We shall prove a sequence of implications to establish the theorem, indicating in each case which hypotheses on $\mathcal{A}$ are used.
$(i) \implies (ii)$ ($\mathcal{A}$ sensible, C definable).

Since $\mathcal{A}$ is sensible, $\mathbf{\Omega}$ diverges in $\mathcal{A}$.

Notation. Given a set Con of constants, $\mathbf{\Lambda}(\mathsf{Con})$ is the set of λ-terms over Con.

For each $\phi \in N\mathcal{L}$ we shall define terms $M_\phi, T_\phi \in \mathbf{\Lambda}(\{\mathsf{P}, \mathsf{C}\})$ such that:

- $M_\phi \models_{\mathcal{A}} \psi \iff \mathcal{L} \vdash \phi \leq \psi$
- $\forall a \in A. \begin{cases} T_\phi a \Downarrow & \text{if } a \models_{\mathcal{A}} \phi, \\ T_\phi a \Uparrow & \text{otherwise.} \end{cases}$

The definition is by induction on the complexity of

$$\phi \equiv \bigwedge_{i \in I} (\phi_{i,1} \to \cdots (\phi_{i,k_i} \to \lambda)_\perp \cdots)_\perp.$$

If $I = \varnothing$, $M_\phi \equiv \mathbf{\Omega}$. Otherwise, we define $M_\phi \equiv M(\phi, k)$, where $k = \max\{k_i \mid i \in I\}$:

$$\begin{aligned} M(\phi, 0) &\equiv \mathbf{K\Omega} \\ M(\phi, i+1) &\equiv \lambda x_j.\, \mathsf{C} N M(\phi, i) \end{aligned}$$

where

$$\begin{array}{rcl} j & \equiv & k-i \\ N & \equiv & \sum\{N_i : j \leq k_i\} \\ N_i & \equiv & \mathsf{C}(T_{\phi_{i,1}}x_1)(\mathsf{C}(T_{\phi_{i,2}}x_2)(\ldots(\mathsf{C}(T_{\phi_{i,j}}x_j))\ldots)) \\ \sum\varnothing & \equiv & \mathbf{\Omega} \\ \sum\{N\}\cup\Theta & \equiv & \mathsf{P}N(\sum\Theta). \end{array}$$

$$\begin{array}{rcl} T_\phi & \equiv & \lambda x.\prod\{xM_{\phi_{i,1}}\ldots M_{\phi_{i,k_i}} : i \in I\} \\ \prod\varnothing & \equiv & \mathbf{K\Omega} \\ \prod\{N\}\cup\Theta & \equiv & \mathsf{C}N(\prod\Theta). \end{array}$$

We must show that these definitions have the required properties. Firstly, we prove for all $\phi \in N\mathcal{L}$:

(1) $M_\phi \models_{\mathcal{A}} \phi$

(2) $a \models_{\mathcal{A}} \phi \;\Rightarrow\; T_\phi a \Downarrow$

by induction on ϕ:

- $\forall i \in I.\, a_j \models_{\mathcal{A}} \phi_{i,j} \;\; (1 \leq j \leq k_i)$

$\Rightarrow$ $M_\phi a_1 \ldots a_{k_i} \Downarrow$ by induction hypothesis (2),

$\therefore$ $M_\phi \vdash_{\mathcal{A}} \phi$.

- $a \models_{\mathcal{A}} \phi$ by induction hypothesis (1)

$\Rightarrow$ $T_\phi a \Downarrow$.

We complete the argument by proving, for all $\phi, \psi \in N\mathcal{L}$:

(3) $M_\phi \models_{\mathcal{A}} \psi \;\Rightarrow\; \mathcal{L} \vdash \phi \leq \psi$

(4) $M_\psi \models_{\mathcal{A}} \phi \;\Rightarrow\; \mathcal{L} \vdash \psi \leq \phi$

(5) $T_\phi M_\psi \Downarrow \;\Rightarrow\; M_\psi \models_{\mathcal{A}} \phi$

(6) $T_\psi M_\phi \Downarrow \;\Rightarrow\; M_\phi \models_{\mathcal{A}} \psi$.

The proof is by induction on $n + m$, where n, m are the number of subformulae of ϕ, ψ respectively. Let

$$\phi \;\equiv\; \bigwedge_{i\in I}(\phi_{i,1} \to \cdots (\phi_{i,k_i} \to \lambda)_\perp \cdots)_\perp,$$

$$\psi \;\equiv\; \bigwedge_{j\in J}(\psi_{j,1} \to \cdots (\psi_{j,k_j} \to \lambda)_\perp \cdots)_\perp.$$

(3):

- $M_\phi \;\models_{\mathcal{A}}\; \psi$

$\Rightarrow$ $\forall j \in J.\, M_\phi M_{\psi_{j,1}} \ldots M_{\psi_{j,k_j}} \Downarrow$ by (1) ,

$\Rightarrow$ $\forall j \in J.\, \exists i \in I.\, k_j \leq k_i \;\&\; T_{\phi_{i,l}} M_{\psi_{j,l}} \Downarrow, \;\; 1 \leq l \leq k_j$

$\Rightarrow$ $M_{\psi_{j,l}} \;\models_{\mathcal{A}}\; \phi_{i,l}, \;\; 1 \leq l \leq k_j$ ind. hyp. (5)

$\Rightarrow$ $\mathcal{L} \;\vdash\; \psi_{j,l} \leq \phi_{i,l}, \;\; 1 \leq l \leq k_j$ ind. hyp. (4)

$\Rightarrow$ $\mathcal{L} \;\vdash\; \phi \leq \psi.$

(4): Symmetrical to (3).
(5):

- $T_\phi M_\psi \Downarrow$

$\Rightarrow$ $\forall i \in I.\, M_\psi M_{\phi_{i,1}} \ldots M_{\phi_{i,k_i}} \Downarrow$

$\Rightarrow$ $\forall i \in I.\, \exists j \in J.\, k_i \leq k_j \;\&\; T_{\psi_{j,l}} M_{\phi_{i,l}} \Downarrow, \;\; 1 \leq l \leq k_i$

$\Rightarrow$ $M_{\phi_{i,l}} \;\models_{\mathcal{A}}\; \psi_{j,l}, \;\; 1 \leq l \leq k_i$ ind. hyp. (6)

$\Rightarrow$ $\mathcal{L} \;\vdash\; \phi_{i,l} \leq \psi_{j,l}, \;\; 1 \leq l \leq k_i$ ind. hyp. (3)

$\Rightarrow$ $\mathcal{L} \;\vdash\; \psi \leq \phi$

$\Rightarrow$ $M_\psi \;\models_{\mathcal{A}}\; \phi$ by (1).

(6): Symmetrical to (5).
$(ii) \implies (iii)$ ($\mathcal{A}$ approximable).
Notation. For each $\phi \in \mathcal{L}$, $a_\phi \in A$ is the element representing ϕ. Given $\Gamma : \mathsf{Var} \to \mathcal{L}$, $\rho_\Gamma \in Env(\mathcal{A})$ is defined by

$$\rho_\Gamma x = a_{\Gamma x}.$$

Finally, $\Gamma_t : \mathsf{Var} \to \mathcal{L}$ is the constant map $x \mapsto t$.

We begin with some preliminary results.

(1) $\mathcal{A} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi$.

One half is the Soundness Theorem for $\mathcal{L}$. For the converse, note that

$$\begin{aligned} \mathcal{A} \models \phi \leq \psi &\Rightarrow a_\phi \models_{\mathcal{A}} \psi \\ &\Rightarrow \mathcal{L} \vdash \phi \leq \psi. \end{aligned}$$

(2) $\forall \psi \in N\mathcal{L}.\, \psi \neq t \,\&\, ab \models_{\mathcal{A}} \psi \Rightarrow \exists \phi.\, a \models_{\mathcal{A}} (\phi \to \psi)_\perp \,\&\, b \models_{\mathcal{A}} \phi$.

This is shown by induction on ψ.

- $ab \models_{\mathcal{A}} \bigwedge_{i \in I} \psi_i \;\; (I \neq \varnothing)$

$\Rightarrow$ $\forall i \in I.\, ab \models_{\mathcal{A}} \psi_i$

$\Rightarrow$ $\forall i \in I.\, \exists \phi_i.\, a \models_{\mathcal{A}} (\phi_i \to \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \phi_i$ by ind. hyp.

$\Rightarrow$ $\forall i \in I.\, a \models_{\mathcal{A}} (\bigwedge_{i \in I} \phi_i \to \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \bigwedge_{i \in I} \phi_i$

$\Rightarrow$ $a \models_{\mathcal{A}} (\bigwedge_{i \in I} \phi_i \to \bigwedge_{i \in I} \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \bigwedge_{i \in I} \phi_i$.

- $ab \models_{\mathcal{A}} (\psi_1 \to \cdots (\psi_k \to \lambda)_\perp \cdots)_\perp$

$\Rightarrow$ $aba_{\psi_1} \dots a_{\psi_k}\Downarrow$

$\Rightarrow$ $\exists \phi, \phi_1, \dots, \phi_k.\, b \models_{\mathcal{A}} \phi \,\&\, a_{\psi_i} \models_{\mathcal{A}} \phi_i \; (1 \leq i \leq k)$

$\& \, a \models_{\mathcal{A}} (\phi \to (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$,

since A is approximable

$\Rightarrow$ $\mathcal{L} \vdash \psi_i \leq \phi_i \; (1 \leq i \leq k)$

$\Rightarrow$ $\mathcal{L} \vdash (\phi \to (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$

$\leq (\phi \to (\psi_1 \to \cdots (\psi_k \to \lambda)_\perp \cdots)_\perp$

$\Rightarrow$ $a \models_{\mathcal{A}} (\phi \to \psi)_\perp \,\&\, b \models_{\mathcal{A}} \phi$.

(3) $\forall M \in \mathbf{\Lambda}.\, M, \Gamma \models_{\mathcal{A}} \phi \iff M, \rho_\Gamma \models_{\mathcal{A}} \phi$.

The right to left implication is clear, since $\rho_\Gamma \models_\mathcal{A} \Gamma$. We prove the converse by induction on M.

$$\begin{aligned} x, \Gamma \models_\mathcal{A} \phi &\iff \mathcal{A} \models \Gamma x \leq \phi \\ &\iff \mathcal{L} \vdash \Gamma x \leq \phi \ \text{ by(1)} \\ &\iff a_{\Gamma x} \models_\mathcal{A} \phi \\ &\iff x, \rho_\Gamma \models_\mathcal{A} \phi. \end{aligned}$$

The case for $\lambda x.M$ is proved by induction on ϕ. We show the non-trivial case.

$$\begin{aligned} \bullet \quad & \lambda x.M, \rho_\Gamma \models_\mathcal{A} (\phi \to \psi)_\perp \\ \Longrightarrow \quad & M, \rho_\Gamma[x \mapsto a_\phi] \models_\mathcal{A} \psi \\ \Longrightarrow \quad & M, \Gamma[x \mapsto \phi] \models_\mathcal{A} \psi \qquad \text{by (outer) induction hypothesis} \\ \Longrightarrow \quad & \lambda x.M, \Gamma \models_\mathcal{A} (\phi \to \psi)_\perp. \end{aligned}$$

$$\begin{aligned} \bullet \quad & MN, \rho_\Gamma \models_\mathcal{A} \psi \\ \Longrightarrow \quad & [\![M]\!]^\mathcal{A}_{\rho_\Gamma} [\![N]\!]^\mathcal{A}_{\rho_\Gamma} \models_\mathcal{A} \psi \\ \Longrightarrow \quad & \exists \phi.\, [\![M]\!]^\mathcal{A}_{\rho_\Gamma} \models_\mathcal{A} (\phi \to \psi)_\perp \mathbin{\&} [\![N]\!]^\mathcal{A}_{\rho_\Gamma} \models_\mathcal{A} \phi \qquad \text{by (2)} \\ \Longrightarrow \quad & M, \Gamma \models_\mathcal{A} (\phi \to \psi)_\perp \mathbin{\&} N, \Gamma \models_\mathcal{A} \phi \qquad \text{ind. hyp.} \\ \Longrightarrow \quad & MN, \Gamma \models_\mathcal{A} \psi. \end{aligned}$$

(4):

$$\begin{aligned} (i) \qquad & x, \Gamma[x \mapsto \phi] \models_\mathcal{A} \psi \iff \mathcal{L} \vdash \phi \leq \psi \\ (ii) \qquad & \lambda x.M, \Gamma \models_\mathcal{A} (\phi \to \psi)_\perp \iff M, \Gamma[x \mapsto \phi] \models_\mathcal{A} \psi \\ (iii) \qquad & MN, \Gamma \models_\mathcal{A} \psi \iff \exists \phi.\, M, \Gamma \models_\mathcal{A} (\phi \to \psi)_\perp \\ & \qquad\qquad \mathbin{\&} N, \Gamma \models_\mathcal{A} \phi. \end{aligned}$$

4(i) is proved using (1).

4(ii):

- $\lambda x.M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_{\perp}$

$\Rightarrow$ $\forall \rho, a.\, \rho \models_{\mathcal{A}} \Gamma \,\&\, a \models_{\mathcal{A}} \phi \;\Rightarrow\; [\![\lambda x.M]\!]^{\mathcal{A}}_{\rho}.a \models_{\mathcal{A}} \psi$

$\Rightarrow$ $\forall \rho.\, \rho \models_{\mathcal{A}} \Gamma[x \mapsto \phi] \;\Rightarrow\; M, \rho \models_{\mathcal{A}} \psi$

since $[\![\lambda x\cdot\, M]\!]^{\mathcal{A}}_{\rho}.a = [\![M]\!]^{\mathcal{A}}_{\rho[x \mapsto a]}$,

$\Rightarrow$ $M, \Gamma[x \mapsto \phi] \models_{\mathcal{A}} \psi$.

The converse follows from the soundness of $\mathcal{L}$.

4(iii):

$$\begin{array}{rcll} MN, \Gamma \models_{\mathcal{A}} \psi & \iff & MN, \rho_{\Gamma} \models_{\mathcal{A}} \psi & \text{by (3)} \\ & \iff & [\![M]\!]^{\mathcal{A}}_{\rho_{\Gamma}} [\![N]\!]^{\mathcal{A}}_{\rho_{\Gamma}} \models_{\mathcal{A}} \psi & \\ & \iff & \exists \phi.\, [\![M]\!]^{\mathcal{A}}_{\rho_{\Gamma}} \models_{\mathcal{A}} (\phi \to \psi)_{\perp} \,\&\, [\![N]\!]^{\mathcal{A}}_{\rho_{\Gamma}} \models_{\mathcal{A}} \phi & \text{by (2)} \\ & \iff & \exists \phi.\, M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_{\perp} \,\&\, N, \Gamma \models_{\mathcal{A}} \phi & \text{by (3)} \end{array}$$

We can now prove

$$M, \Gamma \models_{\mathcal{A}} \phi \;\Rightarrow\; M, \Gamma \vdash \phi$$

by induction on M, using (4).

$(iii) \implies (i)$.

Firstly, note that (iii) implies

$$\mathcal{A} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

One half is the Soundness Theorem. For the converse, suppose $\mathcal{A} \models \phi \leq \psi$ and $\mathcal{L} \nvdash \phi \leq \psi$. Then $\mathbf{I} \models_{\mathcal{A}} (\phi \to \psi)_{\perp}$ but $\mathbf{I} \nvdash (\phi \to \psi)_{\perp}$, and so $\mathcal{A}$ is not $\mathcal{L}$-complete.

Now suppose that P is not definable in $\mathcal{A}$, and consider

$$\phi \equiv (\lambda \to (t \to \lambda)_{\perp})_{\perp} \wedge (t \to (\lambda \to \lambda)_{\perp})_{\perp},$$

$$\psi \equiv (t \to (t \to \lambda)_{\perp})_{\perp}.$$

Clearly, $\mathcal{L} \nvdash \phi \leq \psi$. However, for $a \in \mathcal{A}$, if $a \models_{\mathcal{A}} \phi$, then $x\Downarrow$ or $y\Downarrow$ implies $axy\Downarrow$; since P is not definable in $\mathcal{A}$, and in particular, a does not define P,

we must have $axy\Downarrow$ even if $x\Uparrow$ and $y\Uparrow$, and hence $a \models_{\mathcal{A}} \psi$. Thus $\mathcal{A} \models \phi \leq \psi$ and so by our opening remark, $\mathcal{A}$ is not $\mathcal{L}$-complete.
$(ii) \implies (iv)$ ($\mathcal{A}$ approximable).

Clearly $\mathsf{Im}\, t_{\mathcal{A}} \supseteq \mathcal{K}(D)$, by 5.14(ii). Also, since $\mathcal{A}$ is approximable, we can apply the Characterisation Theorem to deduce that $t_{\mathcal{A}}$ is injective (modulo bisimulation). To show that $t_{\mathcal{A}}$ is a combinatory morphism, we argue as in 6.5.17. Application is preserved by $t_{\mathcal{A}}$ using (2) from the proof of $(ii) \Rightarrow (iii)$ and 6.5.17. The proof is completed by showing that $t_{\mathcal{A}}$ preserves denotations of λ-terms, i.e.

$$\forall M \in \mathbf{\Lambda}, \rho \in Env(\mathcal{A}).\, t_{\mathcal{A}}([\![M]\!]^{\mathcal{A}}_{\rho}) = [\![M]\!]^{D}_{t_{\mathcal{A}} \circ \rho}.$$

The proof is by induction on M. Since it is very similar to the corresponding part of the proof of 6.5.17, we omit it. The only non-trivial point is that in the case for abstraction we need:

$$\forall a \in A.\, a \models_{\mathcal{A}} \phi \implies M, \rho[x \mapsto a] \models_{\mathcal{A}} \psi$$

if and only if

$$M, \rho[x \mapsto a_{\phi}] \models_{\mathcal{A}} \psi,$$

which is proved similarly to (3) in $(ii) \Rightarrow (iii)$.
$(iv) \implies (v)$.

Assuming (iv), $\mathcal{A}$ is isomorphic (modulo bisimulation) to a substructure of D. Since formulas in HF are (equivalent to) universal (Π^0_1) sentences, this yields $\Im(D) \subseteq \Im(\mathcal{A})$. Since $\mathcal{K}(D) \subseteq \mathsf{Im}\, t_{\mathcal{A}}$, to prove the converse it is sufficient to show, for $H \in \mathsf{HF}$:

$$D, \rho \not\models H \implies \exists \rho_0 : \mathsf{Var} \to \mathcal{K}(D).\, D, \rho \not\models H.$$

Let $H \equiv P \Rightarrow F$, where $P \equiv \bigwedge_{i \in I} M_i\Downarrow \wedge \bigwedge_{j \in J} N_j\Uparrow$. There are four cases, corresponding to the form of F.

Case 1: $F \equiv M \sqsubseteq N$. $D, \rho \not\models P \Rightarrow F$ implies $D, \rho \models P$ and $D, \rho \not\models M \sqsubseteq N$. Since D is algebraic, $D, \rho \not\models M \sqsubseteq N$ implies that for some $b \in \mathcal{K}(D)$, $b \sqsubseteq [\![M]\!]^D_{\rho}$ and $b \not\sqsubseteq [\![N]\!]^D_{\rho}$. Since the expression $[\![M]\!]^D_{\rho}$ is continuous in ρ, $b \sqsubseteq [\![M]\!]^D_{\rho}$ implies that for some $\rho_1 : \mathsf{Var} \to \mathcal{K}(D)$, $\rho_1 \sqsubseteq \rho$ and $b \sqsubseteq [\![M]\!]^D_{\rho_1}$. For all ρ' with $\rho_1 \sqsubseteq \rho' \sqsubseteq \rho$, $[\![N]\!]^D_{\rho'} \sqsubseteq [\![N]\!]^D_{\rho}$, and hence $b \not\sqsubseteq [\![N]\!]^D_{\rho'}$. Again, since D is algebraic,

$$D, \rho \models M_i\Downarrow \implies \exists \rho_i : \mathsf{Var} \to \mathcal{K}(D).\, \rho_i \sqsubseteq \rho \,\&\, D, \rho_i \models M_i\Downarrow.$$

Now let $\rho_0 \equiv \bigsqcup_{i \in I} \rho_i \sqcup \rho_1$. This is well-defined since D is a lattice. Moreover, $\rho_0 \sqsubseteq \rho$, and $\rho_0 : \mathsf{Var} \to \mathcal{K}(D)$. Since $\rho_0 \sqsupseteq \rho_i$ $(i \in I)$, $D, \rho_0 \models M_i\Downarrow$; while since $\rho_0 \sqsubseteq \rho$, $D, \rho_0 \models N_j\Uparrow$ $(j \in J)$. Since $\rho_1 \sqsubseteq \rho_0 \sqsubseteq \rho$, $b \sqsubseteq [\![M]\!]^D_{\rho_0}$ and $b \not\sqsubseteq [\![N]\!]^D_{\rho_0}$, and so $D, \rho_0 \not\models M \sqsubseteq N$. Thus $D, \rho_0 \not\models P \Rightarrow F$, as required.

The remaining cases are proved similarly.

$(v) \implies (i)$ ($\mathcal{A}$ sensible).

Consider the formula

$$H \equiv x\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow \wedge\, x(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow \;\Rightarrow\; x\mathbf{\Omega\Omega}\Downarrow.$$

It is easy to see that $\mathcal{A} \models H$ iff P is not definable in $\mathcal{A}$. Since P is definable in D, the result follows. ∎

We now turn to the question of when the bisimulation preorder on an lts can be characterised by means of a contextual equivalence, as in [Bar84, Plo77, Mil77].

Definition 6.6.5 Let $\mathcal{A}$ be an lts, $X, Y \subseteq A$. Then X *separates* Y if:

$$\forall M, N \in \mathbf{\Lambda}^0(Y).\, \mathcal{A} \not\models M \sqsubseteq N \implies$$
$$\exists P_1, \ldots, P_k \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models MP_1 \ldots P_k\Downarrow \;\&\; \mathcal{A} \models NP_1 \ldots P_k\Uparrow.$$

In particular, if X separates A we say that it is a *separating set*. For example, A is always a separating set.

Proposition 6.6.6 *Let $\mathcal{A}$ be an approximable lts, and suppose X separates Y. Then*

$$\forall M, N \in \mathbf{\Lambda}^0(Y).\, \mathcal{A} \models M \sqsubseteq N \iff$$

$$\forall C[\cdot] \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models C[M]\Downarrow \;\Rightarrow\; \mathcal{A} \models C[N]\Downarrow.$$

PROOF. Suppose $\mathcal{A} \not\models M \sqsubseteq N$. Then since X separates Y, for some $P_1, \ldots, P_k \in \mathbf{\Lambda}^0(X)$, $\mathcal{A} \models MP_1 \ldots P_k\Downarrow$ and $\mathcal{A} \models NP_1 \ldots P_k\Uparrow$. Let $C[\cdot] \equiv [\cdot]P_1 \cdots P_k$. For the converse, suppose $\mathcal{A} \models M \sqsubseteq N$ and $\mathcal{A} \models CM\Downarrow$. Since $\mathcal{A}$ is approximable and $\mathcal{A} \models C[M] = \lambda x.C[x]M$, for some ϕ $\lambda x.C[x] \models_{\mathcal{A}} (\phi \to \lambda)_\perp$ and $M \models_{\mathcal{A}} \phi$. Since $\mathcal{A} \models M \sqsubseteq N$, by the Characterisation Theorem $N \models_{\mathcal{A}} \phi$, and so $\mathcal{A} \models C[N]\Downarrow$. ∎

As a first application of this Proposition, we have:

Proposition 6.6.7 *Let $\mathcal{A}$ be a sensible, approximable lts in which* C *and* P *are definable. Then* $\{\mathsf{C}, \mathsf{P}\}$ *is a separating set.*

Proof. By the Full Abstraction Theorem, for each $\phi \in \mathcal{L}$ there is $M_\phi \in \mathbf{\Lambda}^0(\{\mathsf{C}, \mathsf{P}\})$ such that

$$M_\phi \models_{\mathcal{A}} \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

Now

$$\begin{aligned}
\bullet \quad & \mathcal{A} \not\models M \sqsubseteq N \\
\Longrightarrow \quad & \exists \phi.\, M \models_{\mathcal{A}} \phi \,\&\, N \not\models \phi, \quad \text{since A is approximable} \\
\Longrightarrow \quad & \exists \phi_1, \ldots, \phi_k.\, M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
& \quad \& \, N \not\models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
\Longrightarrow \quad & M M_{\phi_1} \ldots M_{\phi_k}\Downarrow \,\&\, N M_{\phi_1} \ldots M_{\phi_k}\Uparrow. \quad \blacksquare
\end{aligned}$$

The hypothesis of approximability has played a major part in out work. We now give a useful sufficient condition.

Definition 6.6.8 Let $\mathcal{A}$ be an lts, $X \subseteq A$. Then $\mathcal{A}$ is X-*sensible* if

$$\forall M \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models M\Downarrow \;\Rightarrow\; D \models M\Downarrow.$$

Here $[\![M]\!]^D$ is the denotation in D obtained by mapping each $a \in X$ to $t_{\mathcal{A}}(a)$. Note that if we extend our endogenous program logic to terms in $\mathbf{\Lambda}^0(X)$, with axioms

$$a, \Gamma \vdash \phi \;\; (\phi \in \mathcal{L}(a)),$$

then the Soundness and Completeness Theorems for D still hold, by a straightforward extension of the arguments used above.

Proposition 6.6.9 *Let $\mathcal{A}$ be an X-sensible lts. Then $\mathcal{A}$ is X-approximable, i.e.*

$$\forall M, N_1, \ldots, N_k \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models M N_1 \ldots N_k\Downarrow \;\Rightarrow\; \exists \phi_1, \ldots, \phi_k.$$

$$M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \,\&\, N_i \models_{\mathcal{A}} \phi_i, \; 1 \leq i \leq k.$$

Proof.

- $\mathcal{A} \models MN_1 \dots N_k \Downarrow$

$\Rightarrow$ $D \models MN_1 \dots N_k \Downarrow$

$\Rightarrow$ $\exists \phi_1, \dots, \phi_k. M \models_{\mathcal{D}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$

$\& \; N_i \models_{\mathcal{D}} \phi_i, \; 1 \le i \le k,$ since D is approximable

$\Rightarrow$ $\exists \phi_1, \dots, \phi_k. M \vdash (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$

$\& \; N_i \vdash \phi_i, \; 1 \le i \le k,$ by extended Completenss

$\Rightarrow$ $\exists \phi_1, \dots, \phi_k. M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$

$\& \; N_i \models_{\mathcal{A}} \phi_i, \; 1 \le i \le k,$ by extended Soundness. ∎

In particular, if X generates $\mathcal{A}$ and $\mathcal{A}$ is X-sensible, then $\mathcal{A}$ is approximable. We now turn to a number of applications of these ideas to syntactically presented lts, i.e. "programming languages".

Firstly, we consider the lts $\ell = (\mathbf{\Lambda}^0, eval)$ defined in section 3 (and studied previously in section 2). Since ℓ is $\varnothing$-sensible by 6.3.11, and it is generated by $\varnothing$, it is approximable by 6.6.9. Since $\varnothing$ is a separating set for $\mathbf{\Lambda}^0$, we can apply 6.6.6 to obtain Theorem 6.2.5.

Next, we consider extensions of ℓ.

Definition 6.6.10 (i) ℓ_{C} is the extension of ℓ defined by

$$\ell_{\mathsf{C}} = (\mathbf{\Lambda}(\{\mathsf{C}\}), \_\Downarrow\_)$$

where $\Downarrow$ is the extension of the relation defined in 6.2.2 with the following rules:

- $\mathsf{C}\Downarrow\mathsf{C}$ • $\dfrac{M\Downarrow}{\mathsf{C}M\Downarrow\mathbf{I}}$

(ii) ℓ_{P} is the extension $(\mathbf{\Lambda}(\{\mathsf{C}\}), \_\Downarrow\_)$ of ℓ with the rules

- $\mathsf{P}\Downarrow\mathsf{P}$ • $\mathsf{P}M\Downarrow\mathsf{P}M$ • $\dfrac{M\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}}$ • $\dfrac{N\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}}$

It is easy to see that the relation $\_\Downarrow\_$ as defined in both ℓ_{C} and ℓ_{P} is a partial function. Moreover, with these definitions the C and P combinators have the properties required by 6.6.3; while C is definable in ℓ_{P}, by

$$\mathsf{C}M \equiv \mathsf{P}MM.$$

Since ℓ_{C} is generated by $\{\mathsf{C}\}$, and ℓ_{P} by $\{\mathsf{P}\}$, these are separating sets. Thus to apply Theorem 6.6.6, we need only check that ℓ_{C} is C-sensible, and ℓ_{P} P-sensible.

To do this for ℓ_{C}, we proceed as follows. Define

$$c \equiv \{(\lambda \to (\phi \to \phi)_\perp)_\perp \mid \phi \in \mathcal{L}\}^\dagger \in \mathsf{Filt}\,\mathcal{L}.$$

Then it is easy to see that $c \subseteq t_{\mathcal{A}}(\mathsf{C})$, and by monotonicity and the Soundness Theorem,

$$[\![M[c/\mathsf{C}]]\!]^D \subseteq [\![M]\!]^D$$

for $M \in \mathbf{\Lambda}^0(\{\mathsf{C}\})$. Thus

$$(\star) \;\; D \models M[c/\mathsf{C}]\Downarrow \implies D \models M\Downarrow.$$

Now we prove

$$\begin{array}{ll}(\star\star) & \forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{C}\}). \\ & M\Downarrow N \implies [\![M[c/\mathsf{C}]]\!]^D = [\![N[c/\mathsf{C}]]\!]^D \;\&\; D \models N[c/\mathsf{C}]\Downarrow,\end{array}$$

which by $(\star)$ yields $\ell_{\mathsf{C}} \models M\Downarrow \Rightarrow D \models M\Downarrow$, as required. $(\star\star)$ is proved by a straightforward induction on the length of the proof that $M\Downarrow N$.

The argument for ℓ_{P} is similar, using

$$p \equiv \{(\lambda \to (t \to (\phi \to \phi)_\perp)_\perp)_\perp \wedge (t \to (\lambda \to (\psi \to \psi)_\perp)_\perp)_\perp : \phi, \psi \in \mathcal{L}\}^\dagger.$$

Altogether, we have shown

Theorem 6.6.11 (Contextual Equivalence) *(i)* $\forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{C}\})$*:*

$$\ell_{\mathsf{C}} \models M \sqsubseteq N \iff \forall C[\cdot] \in \mathbf{\Lambda}^0(\{\mathsf{C}\}).\, \ell_{\mathsf{C}} \models C[M]\Downarrow \Rightarrow \ell_{\mathsf{C}} \models C[N]\Downarrow.$$

(ii) $\forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{P}\})$*:*

$$\ell_{\mathsf{P}} \models M \sqsubseteq N \iff \forall C[\cdot] \in \mathbf{\Lambda}^0(\{\mathsf{P}\}).\, \ell_{\mathsf{P}} \models C[M]\Downarrow \Rightarrow \ell_{\mathsf{P}} \models C[N]\Downarrow.$$

As a further application of these ideas, we have

Proposition 6.6.12 (Soundness of D) *If $\mathcal{A}$ is X-sensible, and X separates X in $\mathcal{A}$, then:*

$$\Im^0(D, X) \subseteq \Im^0(\mathcal{A}, X).$$

PROOF.

$$\begin{array}{ll} \bullet & D \models M \sqsubseteq N \\ \Longrightarrow & \forall C[\cdot] \in \mathbf{\Lambda}^0(X).\, D \models C[M] \sqsubseteq C[N] \\ \Longrightarrow & D \models C[M]\Downarrow \ \Rightarrow \ D \models C[N]\Downarrow \\ \Longrightarrow & \mathcal{A} \models C[M]\Downarrow \ \Rightarrow \ \mathcal{A} \models C[N]\Downarrow \\ \Longrightarrow & \mathcal{A} \models M \sqsubseteq N. \end{array}$$

The argument for formulae of other forms is similar. ∎

As an immediate corollary of this Proposition,

Proposition 6.6.13 *The denotational semantics of each of our languages is* sound *with respect to the operational semantics:*

$$\begin{array}{rl} (i) & \Im^0(D) \subseteq \Im^0(\ell) \\ (ii) & \Im^0(D, \{\mathsf{C}\}) \subseteq \Im^0(\ell_{\mathsf{C}}, \{\mathsf{C}\}) \\ (iii) & \Im^0(D, \{\mathsf{P}\}) \subseteq \Im^0(\ell_{\mathsf{P}}, \{\mathsf{P}\}). \end{array}$$

We now turn to the question of full abstraction for these languages. Since, as we have seen, ℓ_{P} is P-sensible, and hence sensible and approximable, and C and P are definable, we can apply the Full Abstraction Theorem to obtain

Proposition 6.6.14 *D is fully abstract for ℓ_{P}.*

We now use the sequential nature of ℓ and ℓ_{C} to obtain negative full abstraction results for these languages. This will require a few preliminary notions.

Definition 6.6.15 The *one-step reduction* relation $>$ over terms in $\mathbf{\Lambda}$ is the least satisfying the following axioms and rules:

- $(\lambda x.M)N > M[N/x]$ • $\dfrac{M > M'}{MN > M'N}$

This is then extended to $\mathbf{\Lambda}(\{\mathsf{C}\})$ with the additional rules

- $\mathsf{C}(\lambda x.M) > \mathbf{I}$ • $\mathsf{CC} > \mathbf{I}$ • $\dfrac{M > M'}{\mathsf{C}M > \mathsf{C}M'}$

We then define

- $\gg \;\equiv\;$ the reflexive, transitive closure of $>$
- $M\uparrow \;\equiv\; \exists\{M_n\}.\, M = M_0 \;\&\; \forall n.\, M_n > M_{n+1}$
- $M\not> \;\equiv\; M \notin \mathsf{dom}>$
- $M\downarrow \;\equiv\; M \gg N \;\&\; N \not>$.

It is clear that $>$ is a partial function. Note that these relations are being defined over *all* terms, not just closed ones. For closed terms, these new notions are related to the evaluation predicate $\_\Downarrow\_$ as follows:

Proposition 6.6.16 *For* $M, N \in \mathbf{\Lambda}^0$ $(\mathbf{\Lambda}^0(\{\mathsf{C}\})$*):*

$$(i)\quad M\Downarrow N \iff M\downarrow N$$
$$(ii)\quad M\Uparrow \implies M\uparrow.$$

We omit the straightforward proof. The following proposition is basic; it says that "reduction commutes with substitution".

Proposition 6.6.17 $M \gg N \;\Rightarrow\; M[P/x] \gg N[P/x]$.

Proof. Clearly, it is sufficient to show:

$$M > N \;\Rightarrow\; M[P/x] > N[P/x].$$

This is proved by induction on M, and cases on why $M > N$. We give one case for illustration:

$$M \equiv (\lambda y.M_1)M_2 > N \equiv M_1[M_2/y].$$

We assume $x \neq y$; the other sub-case is simpler.

$$\begin{array}{rcll} M[P/x] & = & (\lambda y.M_1[P/x])M_2[P/x] & \\ & > & M_1[P/x][M_2[P/x]/y] & \\ & = & M_1[M_2/y][P/x] & \text{by [Bar84, 2.1.16]} \\ & = & N[P/x]. & \blacksquare \end{array}$$

Now we come to the basic sequentiality property of ℓ from which various non-definability results can be deduced.

Proposition 6.6.18 *For $M \in \mathbf{\Lambda}$, exactly one of the following holds:*

$$\begin{array}{rl} (i) & M\uparrow \\ (ii) & M \gg \lambda x.N \\ (iii) & M \gg xN_1 \ldots N_k \ (k \geq 0). \end{array}$$

Proof. Since $>$ is a partial function, the computation sequence beginning with M is uniquely determined. Either it is infinite, yielding (i); or it terminates in a term N with $N \not>$, which must be in one of the forms (ii) or (iii). ∎

As a consequence of this proposition, we obtain

Theorem 6.6.19 C *is not definable in ℓ. Moreover, D is not fully abstract for ℓ.*

Proof. We shall show that ℓ satisfies

$$(\star) \ \ x = \mathbf{I} \ \text{ or } \ [x\mathbf{\Omega}\Downarrow \iff x(\mathbf{K\Omega})\Downarrow].$$

Indeed, consider any term $M \in \mathbf{\Lambda}^0$. Either $M\Uparrow$, in which case $M\mathbf{\Omega}\Uparrow$ and $M(\mathbf{K\Omega})\Uparrow$, or $M\Downarrow$. In the latter case, by $(\Downarrow\eta)$ we have $\lambda\ell \models M = \lambda x.Mx$. Thus without loss of generality we may take M to be of the form $\lambda x.M'$, with $FV(M) \subseteq \{x\}$. Now applying the three previous propositions to M', we see that in case (i) of 6.6.18, $(\lambda x.M')\mathbf{\Omega}\Uparrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Uparrow$; in case (ii), $(\lambda x.M')\mathbf{\Omega}\Downarrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Downarrow$; finally in case (iii), if $k = 0$, $\lambda x.M' = \mathbf{I}$;

while if $k > 0$, $(\lambda x.M')\mathbf{\Omega}\Uparrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Uparrow$. Since $\mathsf{C} \neq \mathbf{I}$, $\mathsf{C}\mathbf{\Omega}\Uparrow$ and $\mathsf{C}(\mathbf{K\Omega})\Downarrow$, this shows that C is not definable. Moreover, $(\star)$ implies

$$(\star\star)\ \ x\mathbf{\Omega}\Uparrow \,\&\, x(\mathbf{K\Omega})\Downarrow \ \Rightarrow\ x = \mathbf{I}$$

which is not satisfied by D, since C is definable in D, and taking $x = \mathsf{C}$ refutes $(\star\star)$; hence D is not fully abstract for ℓ. ∎

Note that since C is not definable in ℓ, we could not apply the Full Abstraction Theorem. By contrast, to show that D is not fully abstract for ℓ_{C}, it suffices to show that P is not definable. For this purpose, we prove a result analogous to 6.6.18.

Proposition 6.6.20 *For $M \in \mathbf{\Lambda}(\{\mathsf{C}\})$, exactly one of the following conditions holds:*

(i) $M\uparrow$

(ii) $M \gg \lambda x.N$

(iii) $M \gg \mathsf{C}$

(iv) $M \gg \underbrace{\mathsf{C}(\mathsf{C}\ldots(\mathsf{C}}_{n}\, xN_1\ldots N_k)\ldots)P_1\ldots P_m \ \ (n,k,m \geq 0)$

Proof. Similar to 6.6.18. ∎

Theorem 6.6.21 P *is not definable in ℓ_{C}; hence D is not fully abstract for ℓ_{C}.*

Proof. We show that ℓ_{C} satisfies

$$x(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow \,\&\, x\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow \ \Rightarrow\ x\mathbf{\Omega\Omega}\Downarrow,$$

and hence, as in the proof of the Full Abstraction Theorem, P is not definable in ℓ_{C}. As in the proof of 6.6.19, without loss of generality we consider closed terms of the form $\lambda y_1.\lambda y_2.M$. Assume $(\lambda y_1.\lambda y_2.M)(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow$ and $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow$. Applying 6.6.20, we see that case (i) is impossible; cases (ii) and (iii) imply that $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega\Omega}\Downarrow$; while in case (iv), if $x = y_1$, then $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega}(\mathbf{K\Omega})\Uparrow$, *contra hypothesis*; and if $x = y_2$, $(\lambda y_1.\lambda y_2.M)(\mathbf{K\Omega})\mathbf{\Omega}\Uparrow$, also *contra hypothesis*. Thus case (iv) is impossible, and the proof is complete. ∎

For our final non-definability result, we shall consider a different style of extension of ℓ, to incorporate *ground data*. We shall consider the simplest possible such extension, where a single atom is added. This corresponds to the domain equation

$$D_\star = \mathbf{1} + [D_\star \to D_\star]$$

(where + is separated sum), which is indeed an extension of our original domain, in the sense that D is a retract of $D_\star$. $D_\star$ is still a Scott domain (indeed, a coherent algebraic cpo), but it is no longer a lattice; we have introduced *inconsistency* via the sum.

This extension is reflected on the syntactic level by two constants, $\star$ and C. We define

$$\ell_\star = (\mathbf{\Lambda}^0(\{\star, \mathsf{C}\}), \_\Downarrow\_)$$

with $\_\Downarrow\_$ extending the definition for ℓ as follows:

- $\star \Downarrow \star$
- $\mathsf{C} \Downarrow \mathsf{C}$
- $\dfrac{M \Downarrow \lambda x.N}{\mathsf{C}M \Downarrow \mathsf{T}}$ $(\mathsf{T} \equiv \lambda x.\lambda y.x)$
- $\dfrac{M \Downarrow \mathsf{C}}{\mathsf{C}M \Downarrow \mathsf{T}}$
- $\dfrac{M \Downarrow \star}{\mathsf{C}M \Downarrow \mathsf{F}}$ $(\mathsf{F} \equiv \lambda x.\lambda y.y)$

We see that the C combinator introduced here is a natural generalisation (not strictly an extension) of the C defined previously in the pure case. Of course, C corresponds to *case selection*, which in the unary case — lifting being unary separated sum — is just convergence testing.

A theory can be developed for $\ell_\star$ which runs parallel to what we have done for the pure lazy λ-calculus. Some of the technical details are more complicated because of the presence of inconsistency, but the ideas and results are essentially the same. Our reasons for mentioning this extension are twofold:

1. To show how the ideas we have developed can be put in a broader context. In particular, with the extension to $\ell_\star$ the reader should be able to see, at least in outline, how our work can be applied to systems such as Martin-Löf's Type Theory under its Domain Interpretation [DNPS83], and (the analogues of) our results in this section can be used to settle most of the questions and conjectures raised in [DNPS83].

2. To prove an interesting result which clarifies a point about which there seems to be some confusion in the literature; namely, *what is parallel or*?

The *locus classicus* for parallel or in the setting of typed λ-calculus is [Plo77]. But what of untyped λ-calculus? In [Bar84, p. 375], we find the following definition:

$$FMN = \begin{cases} \mathbf{I} & \text{if M or N is solvable,} \\ \text{unsolvable} & \text{otherwise} \end{cases}$$

which (modulo the difference between the standard and lazy theories) corresponds to our parallel convergence combinator P. The point we wish to make is this: in the pure λ-calculus, where (in domain terms) there are no inconsistent data values (since everything is a function), i.e. we have a lattice, parallel convergence does indeed play the role of parallel or, as the Full Abstraction Theorem shows. However, when we introduce ground data, and hence inconsistency, a distinction reappears between parallel convergence and parallel or, and it is definitely *wrong* to conflate them. To substantiate this claim, we shall prove the following result: even if parallel convergence is added to $\ell_\star$, parallel or is still not definable. This result is also of interest from the point of view of the fine structure of definability; it shows that parallelism is not all or nothing even in the simple, deterministic setting of $\ell_\star$.

Definition 6.6.22 $\ell_{\star\mathsf{P}}$ is the extension of $\ell_\star$ with a constant P and the rules

- $\mathsf{P}\Downarrow\mathsf{P}$
- $\mathsf{P}M\Downarrow\mathsf{P}M$
- $\dfrac{M\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}}$
- $\dfrac{N\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}}$

Definition 6.6.23 Let ℓ' be an extension of $\ell_\star$. We say that *parallel or is definable in* ℓ' if for some term M

$$(i) \quad M(\mathbf{K\Omega})\mathbf{\Omega}, M\mathbf{\Omega}(\mathbf{K\Omega}) \text{ converge to abstractions}$$

$$(ii) \quad M \star \star \Downarrow \star \,.$$

Theorem 6.6.24 *Parallel or is not definable in* $\ell_{\star \mathsf{P}}$*.*

Proof. We proceed along similar lines to our previous non-definability results. Firstly, we extend our definition of $>$ as follows:

- $\mathsf{constructor}(M) \equiv M$ is an abstraction, P, C or $\star$
- $\mathsf{constructor}(M) \,\&\, M \neq \star \;\Rightarrow\; \mathsf{C}M > \mathsf{T}$
- $\mathsf{C}\star > \mathsf{F}$
- $\dfrac{M > M'}{\mathsf{C}M > \mathsf{C}M'}$
- $\mathsf{constructor}(M)$ or $\mathsf{constructor}(N) \;\Rightarrow\; \mathsf{P}MN > \mathbf{I}$
- $\dfrac{M > M' \quad N > N'}{\mathsf{P}MN > \mathsf{P}M'N'}$

With these extensions, $>$ is still a partial function, and 6.6.16, 6.6.17 still hold. For each $M \in \mathbf{\Lambda}(\{\star, \mathsf{C}, \mathsf{P}\})$, one of the following two disjoint conditions must hold:

- $M\uparrow$
- $M \gg N \,\&\, N \not> .$

We now define $\mathcal{T}$ to be the set of all terms M in $\mathbf{\Lambda}(\{\star, \mathsf{C}, \mathsf{P}, \bot\})$, where $\bot$ is a new constant, such that:

- $FV(M) \subseteq \{y_1, y_2\}$
- M contains no $>$-redex.

Note that $\mathcal{T}$ is closed under sub-terms.

Lemma A

For all $M \in \mathcal{T}$:

$$M[\mathbf{K\Omega}/y_1, \mathbf{\Omega}/y_2]{\downarrow}a \ \&\ M[\mathbf{\Omega}/y_1, \mathbf{K\Omega}/y_2]{\downarrow}b \ \&\ M[\star/y_1, \star/y_2]{\downarrow}c$$
$$\Rightarrow\ a = b = c = \star \text{ or } \star \notin \{a, b, c\}.$$

Proof. By induction on M. Since terms in $\mathcal{T}$ contain no $>$-redexes, M must have one of the following forms:

$$
\begin{aligned}
(i) &\quad xN_1 \ldots N_k \ \ (x \in \{y_1, y_2\}, k \geq 0)\\
(ii) &\quad \star N_1 \ldots N_k \ \ (k \geq 0)\\
(iii) &\quad \lambda x.N\\
(iv) &\quad \mathsf{C} \ \ (v)\ \mathsf{P} \ \ (vi)\ \mathsf{P}N\\
(vii) &\quad \mathsf{C}NN_1 \ldots N_k \ \ (k \geq 0)\\
(viii) &\quad \mathsf{P}M_1M_2N_1 \ldots N_k \ \ (k \geq 0)\\
(ix) &\quad \bot N_1 \ldots N_k \ \ (k \geq 0)
\end{aligned}
$$

Most of these cases can be disposed of directly; we deal with the two which use the induction hypothesis.

(vii). Firstly, we can apply the induction hypothesis to N to conclude that $N[c_1/y_1, c_2/y_2]$ converges to the same result (i.e. either an abstraction or $\star$) for all three argument combinations c_1, c_2; we can then apply the induction hypothesis to either $N_1N_3 \ldots N_k$ or $N_2N_3 \ldots N_k$.

$(viii)$. Under the hypothesis of the Lemma, we must have

$$(\mathsf{P}M_1M_2)[c_1/y_1, c_2/y_2]{\Downarrow}\mathbf{I}$$

for all three argument combinations c_1, c_2; hence we can apply the induction hypothesis to $N_1 \ldots N_k$. ∎

Lemma B

Let $M \in \mathbf{\Lambda}\ (\{\star, \mathsf{C}, \mathsf{P}\})$, with $FV(M) \subseteq \{y_1, y_2\}$. Then for some $M' \in \mathcal{T}$, for all $P, Q \in \mathbf{\Lambda}^0(\{\star, \mathsf{C}, \mathsf{P}\})$:

$$M[P/y_1, Q/y_2]{\downarrow}\star \iff M'[P/y_1, Q/y_2]{\downarrow}\star.$$

PROOF. Given M, we obtain M' as follows; working in an inside-out fashion, we replace each sub-term N by:

$$\begin{cases} N' & \text{if } N{\downarrow}N' \\ \bot & \text{if } N{\uparrow}. \end{cases} \blacksquare$$

Now suppose that we are given a putative term in $\mathbf{\Lambda}^0(\{\star, \mathsf{C}, \mathsf{P}\})$ defining parallel or. As in the proof of 6.6.21, we may take this term to have the form $\lambda y_1.\lambda y_2.M$. Applying Lemma B, we can obtain $M' \in \mathcal{T}$ from M; but then applying Lemma A, we see that $\lambda y_1.\lambda y_2.M'$ cannot define parallel or. Applying Lemma B again, we conclude that $\lambda y_1.\lambda y_2.M$ cannot define parallel or either. $\blacksquare$

6.7 Variations

Throughout this Chapter, we have focussed on the lazy λ-calculus. We round off our treatment by briefly considering the varieties of function space.

1. The Scott function space

$[D \to E]$, the standard function space of all continuous functions from D to E, which we treated in Chapters 3 and 4. In terms of our domain logic $\mathcal{L}$, we can obtain this construction by adding the axiom

(1) $t \leq (t \to t)$.

Note that with (1), $\mathcal{L}$ collapses to a single equivalence class (corresponding to the trivial one-point solution of $D = [D \to D]$). For this reason, Coppo *et al.* have to introduce atoms in their work on Extended Applicative Type Structures [CDHL84].

2. The strict function space

$[D \to_{\perp} E]$, all *strict* continuous functions. This satisfies (1), and also

(2) $(t \to_{\perp} \phi) \leq f \quad (\phi\downarrow)$.

3. The lazy function space

$[D \to E]_{\perp}$, which satisfies neither (1) nor (2). This has of course been our object of study in this Chapter.

4. The Landin-Plotkin function space

$[D \to_{\perp} E]_{\perp}$, the lifted strict function space. This satisfies (2) but not (1). The reason for our nomenclature is that this construction in the category of domains and strict continuous functions corresponds to Plotkin's $[D \rightharpoonup E]$ construction in his (equivalent) category of predomains and partial functions [Plo85]. Moreover, this may be regarded as the formalisation of Landin's applicative-order λ-calculus, with abstraction used to protect expressions from evaluation, as illustrated extensively in [Lan64, Lan65, Bur75].

The intriguing point about these four constructions is that (1) and (2) are *mathematically* natural, yielding cartesian closure and monoidal closure in e.g. **CPO** and $\mathbf{CPO}_\perp$ respectively (the latter being analogous to partial functions over sets); while (3) and (4) are *computationally* natural, as argued extensively for (3) in this Chapter, and as demonstrated convincingly for (4) by Plotkin in his work on predomains [Plo85]. Much current work is aimed at providing good categorical descriptions of generalisations of (4) [Ros86, RR87, Mog86, Mog87, Mog]; it remains to be seen if a similar programme can be carried out for (3).

Chapter 7

Further Directions

Our development of the research programme adumbrated in Chapter 1 has been fairly extensive, but certainly not complete. There are many possibilities for extension and generalisation of our results. In this Chapter, we shall try to pick out some of the most promising topics for future research.

1. A first, very basic extension would be to rework the material of Chapters 3 and 4 for **SFP** rather than **SDom**. In terms of the meta-language, the extension would be to incorporate the Plotkin powerdomain and the associated term constructions. Our treatment of the Plotkin powerdomain in a specific instance in Chapter 5 should convey the general flavour of what is involved. The extension to **SFP** is conceptually straightforward; we remain within the sphere of coherent spaces. However, there are some technical intricacies which arise with the meta-predicates, to do with the fact that the identification of primes is more subtle in the **SFP** case; this should be clear from our work on normal forms in Chapter 5 section 4. These intricacies are negotiable, and indeed I claim that all our work in this thesis *does* carry over (a detailed account, taking Chapters 3 and 4 of the present thesis as its starting point, is being worked out by a student of Glynn Winskel's [Zha86]).

2. All our work in this thesis has been based on Domain Theory, simply because this is the best established and most successful foundation for denotational semantics, and a wealth of applications are ready to hand. However, our programme is really much more general than this.

Any category of topological spaces in which a denotational metalanguage can be interpreted, and for which a suitable Stone duality exists, could serve as the setting for the same kind of exercise as we carried out in Chapter 4. As one example of this: the main alternatives to domains in denotational semantics over the past few years have been *compact ultrametric spaces* [Niv81, dBZ82, Mat85]. These spaces in their metric topologies are Stone spaces, and indeed the category of compact ultrametric spaces and continuous maps is *equivalent* to the category of second-countable Stone spaces [Abr]. A restricted denotational metalanguage comprising product, (disjoint) sum and powerdomain (the Vietoris construction [Joh85, Smy83b], which in this context is induced by the Hausdorff metric [Niv81, dBZ82, Mat85]), can be interpreted in **Stone**, together with the corresponding sub-language of terms (with *guarded* recursion, leading to *contracting* maps, and hence unique fixpoints [Niv81, dBZ82, Mat85]). Under the classical Stone duality as expounded in Chapter 1, the corresponding logical structures are Boolean algebras, and a *classical* logic can be presented for this metalanguage in entirely analogous fashion to that of Chapter 4. Since the meta-language is rich enough to express a domain equation for synchronisation trees, a case study along the same lines as that of Chapter 5 can be carried through. Moreover, there is a satisfying relationship between the Stone space of synchronisation trees (which is the metric topology on the ultrametric space constructed in [dBZ82]), and the corresponding domain studied in Chapter 5; namely, the former is the *subspace of maximal elements* of the latter. This is in fact an instance of a general relationship, as set out in [Abr]. The important point here is that our programme is just as applicable to the metric-space approach to denotational semantics as to the domain-theoretic approach.

3. A further kind of generalisation would be to structures other than topological spaces. Many Stone-type dualities in such alternative contexts are known; e.g. Stone-Gelfand-Naimark duality for $C^\star$-algebras, Pontrjagin duality for topological groups, Gabriel-Ulmer duality for locally finitely presented categories, etc. [Joh82]. Particularly promising for Computer Science applications are the measure-theoretic dualities studied by Kozen [Koz83] as a basis for the semantics and logic of

probabilistic programs. A very interesting feature of these dualities is that whereas the purely topological dualities have the Sierpinski space $\mathbb{O}$ as their "schizophrenic object" (see [Joh82, Chapter 6]), i.e. the fundamental relationship $P \models \phi$ takes values in $\{0, 1\}$, the measure-theoretic dualities take their "characters" in the reals; satisfaction of a measurable function by a measure is expressed by *integration* [Koz83]. The richer mathematical structure of these dualities should deepen our understanding of the framework. Furthermore, there are intriguing connections with Lawvere's concept of "generalised logics" [Law73].

4. The logics of compact-open sets considered in this thesis have been very weak in expressive power, and are clearly inadequate as a specification formalism. For example, we cannot specify such properties of a stream computation as "emits an infinite sequence of ones". Thus we need a language, with an accompanying semantic framework, which permits us to go beyond compact-open sets. A first step would be to allow the expression of more general open sets, e.g. by means of a least fixed point operator on formulae $\mu p.\phi$, permitting the finite description of infinite disjunctions $\bigvee_{i \in \omega} \phi^i(f)$. This would have the advantage of not requiring any major extension of our semantics, but would still not be sufficiently expressive for specification purposes, as the above example shows. What is needed is the ability to express infinite *conjunctions*, e.g. by *greatest* fixpoints $\nu p.\phi$, corresponding to $\bigwedge_{i \in \omega} \phi^i(t)$. Such an extension of our logic would necessarily take us beyond open sets. An important topic for further investigation is whether such an extension can be smoothly engineered and given a good conceptual foundation.

   Another reason for extending the logic is the tempting proximity of locale theory to topos theory. Could this be the basis of the junction between topos theory and Computer Science which many researchers have looked for but none has yet convincingly demonstrated? We must leave this point unresolved. If there *is* a natural extension of our work to the level of topos theory, we have not (yet) succeeded in finding it.

5. Another variation is to change the *morphisms* under consideration. Stone dualities relating to the various powerdomain constructions (i.e. dualities for *multi-functions* rather than functions) are interesting for a number of reasons: they generalise *predicate transformers* in the sense

of Dijkstra [Dij76, Smy83b]; dualities for the Vietoris construction provide a natural setting for intuitionistic modal logic, with interesting differences to the approach recently taken by Plotkin and Stirling; while there are some remarkable *self-dualities* arising from the Smyth powerdomain [Vic87]. These turn out, quite unexpectedly, to provide a model for Girard's classical linear logic [Gir87]; more speculatively, they also suggest the possibility of a homogeneous logical framework in which programs and properties are interchangeable. This may turn out to provide the basis for a unified and systematic treatment of a number of existing *ad hoc* formalisms [GS86, Win85].

6. Turning now to the first of our case studies, a number of interesting further developments suggest themselves. Firstly, from the results of Chapter 5, we can define a fully abstract denotational semantics for SCCS in our denotational metalanguage, and faithfully interpret Hennessy-Milner logic into our domain logic. Thus we should *automatically* get a compositional proof theory for HML. It would be particularly worthwhile to demonstrate this in detail, as the construction of compositional proof systems for HML by Stirling [Sti87] and Winskel [Win85] is one of the most impressive examples to date of the exercise of *ad hoc* ingenuity in the design of program logics.

   Other useful extensions of our work would be to equivalences other then bisimulation (hard); and to countable non-determinism, using Plotkin's powerdomain for countable non-determinism [Plo82]. An interesting point about this construction is that we lack a good representation for it, and a logical description might help.

7. Our development of the lazy λ-calculus represents no more than a beginning. An extensive study is being undertaken by Luke Ong; anyone interested in pursuing the subject further is strongly recommended to read his forthcoming thesis (Imperial College, University of London; expected 1988).

8. Some more general points concerning the two case studies. Firstly, the operational models we study—labelled transition systems in Chapter 5 and lambda transition systems in Chapter 6—are *almost* derived in a systematic way from our domain equations. Namely, a labelled

transition system is a map

$$\text{Proc} \longrightarrow \wp((\mathsf{Act} \times \text{Proc}) \cup \{\bot\})$$

i.e. a coalgebra of the functor (on **Set**)

$$X \mapsto \wp((\mathsf{Act} \times X) \cup \{\bot\}).$$

Similarly, an applicative transition system is a coalgebra of the **Set**-functor

$$X \mapsto (X \to X) \cup \{\bot\}.$$

Since $\mathsf{Act} \times \mathcal{D} \cup \{\bot\}$ can be put in natural bijection with $\sum_{a \in \mathsf{Act}} \mathcal{D}$, and $(\mathcal{D} \to \mathcal{D}) \cup \{\bot\}$ with $(\mathcal{D} \to \mathcal{D})_\bot$, we see that our domain equations give rise to essentially the *same* functors, but over domains rather than sets. Moreover, because of the limit-colimit coincidence in Domain theory [SP82], we can take the *initial solution* of a domain equation (with respect to embeddings) as the *final coalgebra* (with respect to projections). Thus our results can in some sense be seen as concerning the interpretation and "best approximation" of **Set**-based structures in topological ones. Clearly some general theory is called for here.

9. Finally, one of our aims in Chapters 5 and 6 was to place the study of functional languages and concurrency on as similar a footing as possible. Much remains to be done here, although we hope to have made a useful first step.

Bibliography

[AB84] D. Austry and G. Boudol. Algèbres de processus et synchronisations. *Theoretical Computer Science*, 30:91–131, 1984.

[Abr] S. Abramsky. Total vs. partial objects in denotational semantics. To appear.

[Abr83a] S. Abramsky. Experiments, powerdomains and fully abstract models for applicative multiprogramming. In M. Karpinski, editor, *Foundations of Computation Theory*, pages 1–13, Springer-Verlag, Berlin, 1983. Lecture Notes in Computer Science Vol. 158.

[Abr83b] S. Abramsky. Semantic foundations for applicative multiprogramming. In J. Diaz, editor, *Automata, Languages and programming*, pages 1–14, Springer-verlag, Berlin, 1983. Lecture Notes in Computer Science Vol. 154.

[Abr87a] S. Abramsky. Domain theory in logical form. In *Symposium on Logic In Computer Science*, pages 47–53, Computer Society Press of the IEEE, 1987.

[Abr87b] S. Abramsky. Observation equivalence as a testing equivalence. *Theoretical Computer Science*, 53, 1987.

[AH87] S. Abramsky and C. L. Hankin, editors. *Abstract Interpretation for Declarative Languages.* Ellis Horwood, 1987.

[Aug84] L. Augustsson. A compiler for lazy ML. In *ACM Symposium on Lisp and Functional Programming, Austin*, pages 218–227, August 1984.

[Bar75] K. J. Barwise. Back and forth through infinitary logic. In M. Morley, editor, *Studies in Model Theory*, Mathematical Association of America, 1975.

[Bar84] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[BC85] G. Berry and P.-L. Curien. Theory and practice of sequential algorithms: the kernel of the applicative language CDS. In J. C. Reynolds and M. Nivat, editors, *Algebraic Semantics*, pages 35–84, Cambridge University Press, 1985.

[BCL85] G. Berry, P.-L. Currien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132, Cambridge University Press, 1985.

[BK84] J. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60:109–137, 1984.

[Bur75] W. H. Burge. *Recursive programming techniques*. Addison Wesley, Reading, Mass., 1975.

[BvL86] H. Barendregt and M. van Leeuwen. *Functional programming and the language TALE*. Technical Report 412, University of Utrecht Dept. of Mathematics, 1986.

[BW84] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer Verlag, Berlin, 1984.

[CDHL84] M. Coppo, M. Dezani-Ciancaglini, Furio Honsell, and G. Longo. Extended type structure and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquim '82*, pages 241–262, Elsevier Science Publishers B.V. (North-Holland), 1984.

[CK73] C. C. Chang and H. J. Keisler. *Model Theory*. North Holland, Amsterdam, 1973.

[dB80] J. W. de Bakker. *Mathematical Theory of Program Correctness.* Prentice Hall International, 1980.

[dBZ82] J. W. de Bakker and J. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54:70–120, 1982.

[Dij76] Edsger W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[DM82] L. Damas and R. Milner. Principal type schemes for functional programs. In *Ninth Annual ACM Symposium on the Principles of Programming Languages*, pages 207–212, ACM, 1982.

[DNPS83] P. Dybjer, B. Nordström, K. Petersson, and J. Smith, editors. *Workshop on Semantics of Programming Languages*, Programming Methodology Group, Chalmers University of Technology, Göteborg, Sweden, August 1983.

[Dug66] J. Dugundji. *Topology.* Allyn and Bacon, 1966.

[EC76] H. Egli and R. Constable. Computability concepts for programming language semantics. *Theoretical Computer Science*, 2:133–145, 1976.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1.* Volume 6 of *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag, Berlin, 1985.

[Ers72] Yu. L. Ershov. Computable functionals of finite types. *Algebra and Logic*, 11(4):367–437, 1972.

[Fai85] J. Fairbairn. *Design and implementation of a simple typed language based on the lambda calculus.* PhD thesis, University of Cambridge, 1985.

[GHK*80] G. K. Gierz, K. H. Hoffmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, Berlin, 1980.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 1987.

[Gor79] M. J. C. Gordon. *The Denotational Description of Programming Languages*. Springer-Verlag, Berlin, 1979.

[GS86] S. Graf and J. Sifakis. A logic for the specification and proof of regular controllable processes of ccs. *Acta Informatica*, 23:507–527, 1986.

[GTW78] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. T. Yeh, editor, *Current Trends in Programming Methodology IV: Data Structuring*, pages 80–144, Prentice Hall, 1978.

[GTWW77] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24:68–95, 1977.

[Gue81] I. Guessarian. *Algebraic Semantics*. Volume 99 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1981.

[Gun85] C. Gunter. *Profinite Solutions for Recursive Domain Equations*. PhD thesis, Carnegie-Mellon University, 1985.

[Gun86] C. Gunter. The largest first-order axiomatizable cartesian closed category of domains. In A. R. Meyer, editor, *Symposium on Logic in Computer Science*, pages 142–148, IEEE Computer Society press, 1986.

[Har79] D. Harel. *First Order Dynamic Logic*. Volume 68 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1979.

[Hen80] P. Henderson. *Functional Programming: Applications and Implementation*. Prentice Hall, 1980.

[Hen81] M. C. B. Hennessy. A term model for synchronous processes. *Information and Control*, 51(1):58–75, 1981.

[HM85] M. C. B. Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *JACM*, 32:137–161, 85.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–580, 1969.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1985.

[HP79] M. C. B. Hennessy and G. D. Plotkin. Full abstraction for a simple parallel programming language. In J. Beçvar, editor, *Mathematical Foundations of Computer Science*, Springer-Verlag, Berlin, 1979. Lecture Notes in Computer Science Vol. 74.

[Hyl81] J. M. Hyland. Function spaces in the category of locales. In *Continuous Lattices*, pages 264–281, 1981. Lecture Notes in Mathematics Vol. 871.

[Joh82] P. T. Johnstone. *Stone Spaces.* Volume 3 of *Cambridge Studies in Advanced Mathematics*, Cambridge University Press, Cambridge, 1982.

[Joh85] P. T. Johnstone. Vietoris locales and localic semi-lattices. In R.-E. Hoffmann and K. H. Hoffmann, editors, *Continuous lattices and their Applications*, pages 155–180, Marcel Dekker, 1985. Pure and Aplied Mathematics Vol. 101.

[Kan79] A. Kanda. Fully effective solutions of recursive domain equations. In J. Beçvar, editor, *Mathematical Foundations of Computer Science*, Springer-Verlag, Berlin, 1979. Lecture Notes in Computer Science Vol. 74.

[Koy82] Christiaan Peter Jozef Koymans. Models of the lambda calculus. *Information and Control*, 52:206–332, 1982.

[Koz83] D. Kozen. A probabilistic pdl. In *15th Annual ACM Symposium on Theory of Computing*, pages 291–297, 1983.

[Kre59] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In *Constructivity in Mathematics*, North Holland, Amsterdam, 1959.

[Lan64] P. J. Landin. The mechanical evaluation of expressions. *Computer Journal*, 6:308–320, 1964.

[Lan65] P. J. Landin. A correspondence between ALGOL 60 and Church's lambda notation. *Communications of the ACM*, 8:89–101,158–165, 1965.

[Law73] F. W. Lawvere. Metric spaces, generalised logic, and closed categories. In *Rend. del Sem. Mat. e Fis. di Milano*, 1973. Vol. XLIII.

[Lon83] Giuseppe Longo. Set-theoretical models of lambda calculus: theories, expansions and isomophisms. *Annals of Pure and Applied Logic*, 24:153–188, 1983.

[LS86] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic. Cambridge Studies in Advanced Mathematics Vol. 7*, Cambridge University Press, 1986.

[LW84] K. G. Larsen and G. Winskel. Using information systems to solve recursive domain equations effectively. In D. B. MacQueen G. Kahn and G. Plotkin, editors, *Semantics of Data Types*, pages 109–130, Springer-Verlag, Berlin, 1984. Lecture Notes in Computer Science Vol. 173.

[Mar70] P. Martin-Löf. *Notes on Constructive Mathematics.* Almqvist and Wiksell, Stockholm, 1970.

[Mar83] P. Martin-Löf. Lecture notes on the domain interpretation of type theory. In Programming Methodology Group, editor, *Workshop on the Semantics of Programming Languages*, Chalmers University of Technology, Göteborg, Sweden, 1983.

[Mat85] S. Matthews. *Metric Domains for Completeness.* PhD thesis, University of Warwick, 1985.

[Mil75] R. Milner. Processes, a mathematical model of computing agents. In *Logic Colloquium, Bristol 1973*, pages 157–174, North Holland, Amsterdam, 1975.

[Mil77] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[Mil80] R. Milner. *A Calculus for Communicating Systems*. Volume 92 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1980.

[Mil81] R. Milner. A modal characterisation of observable machine behaviours. In G. Astesiano and C. Böhm, editors, *CAAP '81*, pages 25–34, Springer-Verlag, Berlin, 1981. Lecture Notes in Computer Science Vol. 112.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil85] R. Milner. Lectures on a calculus for communicating systems. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, pages 197–221, Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science Vol. 197.

[Mil86] R. Milner. Process constructors and interpretations. In H.-J. Kugler, editor, *Information Processing 86*, pages 507–514, Elsevier Science Publishers B. V. (North Holland), 1986.

[ML71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, Berlin, 1971.

[MM79] G. Milne and R. Milner. Concurrent processes and their syntax. *Journal of the ACM*, 26(2):302–321, 1979.

[Mog] Eugenio Moggi. Partial cartesian closed categories of effective objects. To Appear in *Information and Computation*.

[Mog86] Eugenio Moggi. Categories of partial morphisms and the $lambda_p$-calculus. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Category Theory and Computer Programming*, pages 242–251, Springer-Verlag, 1986. *LNCS* Vol. 240.

[Mog87] Eugenio Moggi. *Partial Lambda Calculus*. PhD thesis, University of Edinburgh, 1987.

[Mor68] J. H. Morris. *Lambda Calculus Models of Programming Languages*. PhD thesis, Massachusets Institute of Technology, 1968.

[Mos74] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North Holland, 1974.

[MS76] R. E. Milne and C. Strachey. *A Theory of Programming Language Semantics*. Chapman and Hall, London, 1976.

[Myc81] A. Mycroft. *Abstract Interpretation and Optimising Transformations for Applicative Programs*. PhD thesis, University of Edinburgh, 1981.

[Nie84] F. Nielsen. *Abstract Interpretation Using Domain Theory*. PhD thesis, University of Edinburgh, 1984.

[Niv81] M. Nivat. Infinite words, infinite trees, infinite computations. In J. W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science III part 2*, pages 3–52, Centrum voor Wiskunde en Informatica, Amsterdam, 1981.

[Ole85] F. J. Oles. Type categories, functor categories and block structure. In M. Nivat amd J. C. Reynolds, editor, *Algebraic Semantics*, pages 543–574, Cambridge University Press, 1985.

[Par81] D. M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conference on Theoretical Computer Science*, Springer-Verlag, Berlin, 1981. Lecture Notes in Computer Science Vol. 104.

[Plo76] G. D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5:452–487, 1976.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Plo81] G. D. Plotkin. Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa Notes"). 1981. Dept. of Computer Science, Univ. of Edinburgh.

[Plo82] G. D. Plotkin. A powerdomain for countable non-determinism. In M. Nielsen and E. M. Schmidt, editors, *Automata, Languages and programming*, pages 412–428, EATCS, Springer-Verlag, Berlin, 1982. Lecture Notes in Computer Science Vol. 140.

[Plo85] G. D. Plotkin. Lectures on predomains and partial functions. 1985. Notes for a course given at the Center for the Study of Language and Information, Stanford 1985.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 19th Annual Symposium on the Foundations of Computer Science*, Computer Society Press of the IEEE, 1977.

[Poi86] Axel Poigné. On specifications, theories and models with higher types. *Information and Control*, 68, 1986.

[Pra81] V. R. Pratt. Dynamic logic. In J. W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science III Part 2*, pages 53–84, Centrum voor Wiskunde en Informatica, Amsterdam, 1981.

[Ros86] Giuseppe Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, Carnegie-Mellon University, 1986.

[Rou85] W. C. Rounds. Applications of topology to semantics of communicating processes. In S. D. Brookes, A. W. Roscoe, and G. Winskel, editors, *Seminar on Concurrency*, pages 360–372, Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science Vol. 197.

[RR87] E. Robinson and G. Rosolini. Categories of partial maps. 1987. Computing laboratory and DPMMS, Cambridge University.

[Sch86] D. A. Schmidt. *Denotational Semantics*. Allyn and Bacon, 1986.

[Sco70] D. S. Scott. Outline of a mathematical theory of computation. In *4th Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.

[Sco76] D. S. Scott. Data types as lattices. *SIAM J. Computing*, 5:522–587, 1976.

[Sco80a] D. S. Scott. Lambda calculus: some models, some philosophy. In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, pages 223–265, North-Holland Publishing Company, 1980.

[Sco80b] D. S. Scott. Relating theories of lambda calculus. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 403–450, Academic Press, 1980.

[Sco81] D. S. Scott. *Lectures on a Mathematical Theory of Computation.* Monograph PRG-19, Oxford University Computing Laboratory, Oxford, 1981.

[Sco82] D. S. Scott. Domains for denotational semantics. In M. Nielson and E. M. Schmidt, editors, *Automata, Languages and Programming: Proceedings 1982*, Springer-Verlag, Berlin, 1982. Lecture Notes in Computer Science **140**.

[Smy77] M. B. Smyth. Effectively given domains. *Theoretical Computer Science*, 5:257–274, 1977.

[Smy83a] M. B. Smyth. The largest cartesian closed category of domains. *Theoretical Computer Science*, 27:109–119, 1983.

[Smy83b] M. B. Smyth. Powerdomains and predicate transformers: a topological view. In J. Diaz, editor, *Automata, Languages and Programming*, pages 662–675, Springer-Verlag, Berlin, 1983. Lecture Notes in Computer Science Vol. 154.

[Soa87] R. I. Soare. *Recursively Enumerable Sets and Degrees. Perspectives in Mathematical Logic*, Springer-Verlag, Berlin, 1987.

[SP82] M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Computing*, 11:761–783, 1982.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

[Sto36] M. H. Stone. The theory of representations for Boolean algebras. *Trans. American Math. Soc.*, 37–111, 1936.

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* The MIT Press, 1977. The MIT Press Series in Computer Science.

[Tur85] D. A. Turner. Miranda—a non-strict functional language with polymorphic types. In J.P. Jouannaud, editor, *Functional programming languages and Computer Architectures*, Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science Vol. 201.

[Vic87] S. J. Vickers. USCC. 1987. Draft paper, Imperial College.

[Wad85] P. Wadler. *Introduction to ORWELL.* Technical Report, Oxford University Programming Research Group, 1985.

[WBT85] E. G. Wagner, S. L. Bloom, and J. W. Thatcher. Why algebraic theories? In M. Nivat and J. C. Reynolds, editors, *Algebraic Semantics*, pages 607–634, Cambridge University press, 1985.

[Win80] G. Winskel. *Events in Computation.* PhD thesis, University of Edinburgh, 1980.

[Win83] G. Winskel. Powerdomains and modality. In M. Karpinski, editor, *Foundations of Computation Theory*, pages 505–514, Springer-Verlag, Berlin, 1983. Lecture Notes in Computer Science Vol. 158.

[Win85] G. Winskel. A complete proof system for SCCS with modal assertions. In S. N. Maheshwari, editor, *Foundations of Software technology and Theoretical Computer Science*, pages 392–410, Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science Vol. 206.

[Zha86] Guo Qiang Zhang. A logic for SFP. December 1986. Draft paper, University of Cambridge.