Domain Theory In Logical Form

Samson Abramsky

Department of Computing
Imperial College of Science and Technology
180 Queen's Gate
London SW7 2BZ
England

To appear in *Annals of Pure and Applied Logic*

December 12, 1989

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. Finally, Chapter 5 discusses directions for further research.

Contents

Chapter 1

Introduction

Our aim is 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.

The key to our synthesis is the mathematical theory of Stone duality, which 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. As a worked example, we show how Domain Theory can be construed as a logic of observable properties; applications to the study of programming languages will be presented elsewhere [Abr88,Abr87a].

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,Sco80]. 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 [ML83]. His formulation also traces a line of descent from Kreisel's definition of the continuous functionals [Kre59], via [ML70,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 algebra 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

$$B \cong \mathsf{Clop}\ \mathsf{Spec}\ B \qquad b \mapsto \{x \in \mathsf{Spec}\ B : b \in x\} \tag{1.1}$$

$$S \cong \mathsf{Spec}\ \mathsf{Clop}\ S \qquad s \mapsto \{U \in \mathsf{Clop}\ S : s \in U\}. \tag{1.2}$$

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 paper is to elaborate the programme outlined in 1–4 above (applications as in 5 will be presented elsewhere [Abr88,Abr87a]). Chapter 2 is devoted to filling in some background on domains and locales. Then Chapters 3 and 4 present our results. Finally, Chapter 5 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 N" (or just: "is defined to be N"). We shall use ω and $\mathbb{N}$ to denote the natural numbers $\{0, 1, \ldots\}$ (thought of sometimes as an ordinal, and sometimes as just a set). 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 $X \subseteq^{\mathsf{f}} Y$ ($X \subseteq^{\mathsf{fne}} Y$) for the assertion that X is a finite (finite non-empty) 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

$$\begin{array}{lcl}
{\downarrow}(X) & = & \{y \in P : \exists x \in X.\, y \leq x\} \\
{\uparrow}(X) & = & \{y \in P : \exists x \in X.\, x \leq y\} \\
\mathsf{Con}(X) & = & \{y \in P : \exists x, z \in X.\, x \leq y \leq z\} \\
\mathsf{UB}(X) & = & \{y \in P : \forall x \in X.\, x \leq y\} \\
\mathsf{MUB}(X) & = & \{y \in \mathsf{UB}(X) : \forall z \in \mathsf{UB}(X).\, z \leq y \Rightarrow y \leq z\}
\end{array}$$

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 $X = \mathsf{Con}(X)$. $\mathsf{UB}(X)$ is the set of *upper bounds* of X, and $\mathsf{MUB}(X)$ the *minimal upper bounds*. 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:

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

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 main reference for Domain theory. We shall also refer to [Gun85,Gun87].

For the reader's convenience, we briefly review some basic definitions. Let $(P, \sqsubseteq)$ be a poset. A subset $X \subseteq P$ is *directed* if every finite subset of X has an upper bound in X. (Thus a directed subset is non-empty). A *directed-complete partial order* (dcpo) is a poset in which every directed subset X has a least upper bound (written $\bigsqcup X$). A *morphism* of dcpo's is a map $f : D \to E$ satisfying

$$f(\bigsqcup X) = \bigsqcup\{f(x) : x \in X\}$$

for all directed $X \subseteq D$. Let D be a dcpo. The *Scott topology* on D, $\sigma(D)$, is given by all subsets $U \subseteq D$ satisfying

- $U = {\uparrow}(U)$
- $\bigsqcup S \in U$, S directed implies $S \cap U \neq \varnothing$.

Fact 2.2.1 *A function $f : D \to E$ is a morphism if and only if it is continuous with respect to the Scott topology.*

In the light of this fact, morphisms are referred to as continuous maps.

Let $(D, \sqsubseteq)$ be a dcpo. An element $b \in D$ is *finite* if, whenever $S \subseteq D$ is directed and $b \sqsubseteq \bigsqcup S$, $b \sqsubseteq d$ for some $d \in S$. We write $\mathcal{K}(D)$ for the sub-poset of finite elements of D. A subset $B \subseteq D$ is a *basis* for D if for every $d \in D$, $S = B \cap \downarrow(d)$ is directed, and $d = \bigsqcup S$. A dcpo is *algebraic* if $\mathcal{K}(D)$ is a basis, and *ω-algebraic* if $\mathcal{K}(D)$ is also countable. We shall refer to ω-algebraic dcpo's D with least elements (written $\perp_D$) as *(algebraic) domains.* If D is algebraic, the Scott topology has a particularly simple form, namely all sets of the form

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

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

By a *category of domains* we shall mean a sub-category of **DCPO**, the category of dcpo's with least elements and continuous functions[1]. $\mathbf{DCPO}_\perp$ is the sub-category of *strict* functions, i.e. those satisfying $f(\perp) = \perp$.

The properties of **DCPO** 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 dcpo's ([Plo81, Chapter 1 Theorem 1]), and the initial fixed point theorem for continuous functors on **DCPO** ([Plo81, Chapter 5 Theorem 1]). As for (2), the type constructions available over **DCPO** 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 paper, with references to the definitions in [Plo81]:

$A \times B$	product	Ch. 2 p. 2
$(A \to B)$	function space	Ch. 2 p. 9
$A \oplus B$	coalesced sum	Ch. 3 p. 6
$(A)_\perp$	lifting	Ch. 3 p. 9
$(A \to_\perp B)$	strict function space	Ch. 1 p. 13
$\mathcal{P}_l(A)$	lower (Hoare) powerdomain	Ch. 8 p. 14
$\mathcal{P}_u(A)$	upper (Smyth) powerdomain	Ch. 8 p. 45
$\mathcal{P}(A)$	convex (Plotkin) powerdomain	Ch. 8 p. 28

(Note that *separated sum* $A + B$ can be defined by: $A + B \equiv (A)_\perp \oplus (B)_\perp$.) Most of these constructions have simple concrete descriptions as operations on dcpo's. Thus $A \times B$ is

[1] Plotkin uses a slightly different category **CPO** in [Plo81]. The difference is marginal, and for the ω-algebraic case we are mainly concerned with the two notions in fact coincide.

Cartesian product, ordered componentwise; $A \to B$ is the set of continuous maps, ordered pointwise; $A \oplus B$ is the disjoint union, with bottom elements identified; $(A)_\perp$ is A with a new bottom element adjoined; $A \to_\perp B$ is the set of strict continuous maps, ordered pointwise. The description of the powerdomains makes some use of topological notions. The Smyth powerdomain is given as all non-empty Scott-compact upper-closed subsets, ordered by superset; the Hoare powerdomain as all non-empty Scott-closed subsets, ordered by subset inclusion; the Plotkin powerdomain as all non-empty Scott-compact subsets S satisfying

$$S = \uparrow(S) \cap \bar{S}$$

(where $\bar{S}$ is the Scott-closure of S), with the Egli-Milner ordering[2].

In this paper, we shall be concerned with sub-categories of ω**ALG**, the category of algebraic domains and continuous maps. In particular, we shall be concerned with the following full sub-category of ω**ALG**:

- The category **SFP** of *strongly algebraic* domains [Plo76,Plo81,Gun85,Gun87].

The name is an acronym for "Sequences of Finite Posets", which arises from one of the main descriptions of **SFP** domains, as bilimits (i.e. direct limits of embeddings, or, equivalently by the "limit-colimit coincidence" [SP82], inverse limits of projections) of sequences of finite posets. The other main description of **SFP** is "intrinsic", i.e. expressed in terms of conditions on the poset of finite elements.

Definition 2.2.2 Let D be an algebraic domain. We say that D is:

- *coherent algebraic* if $\mathcal{K}(D)$ satisfies "property M" [Gun85,Gun87]: for every finite $u \subseteq \mathcal{K}(D)$, $\mathsf{MUB}(u)$ is finite, and moreover *complete* in the sense that

$$\forall x \in \mathsf{UB}(u). \exists y \in \mathsf{MUB}(u).\, y \sqsubseteq x.$$

- *SFP* if it is coherent algebraic, and moreover for every finite $u \subseteq \mathcal{K}(D)$, $\mathcal{U}^\star(u)$ is finite, where

$$\begin{aligned} \mathcal{U}^\star(u) &= \textstyle\bigcup_{k \in \omega} \mathcal{U}^k(u) \\ \mathcal{U}^0(u) &= u \\ \mathcal{U}^{k+1}(u) &= \textstyle\bigcup \{\mathsf{MUB}(v) : v \subseteq \mathcal{U}^k(u)\} \end{aligned}$$

The justification for studying this category comes from the fact that **SFP** is closed under all the type constructions listed above. In particular, it is cartesian closed; indeed, **SFP** is the *largest* cartesian closed full sub-category of ω**ALG** [Smy83a]. Moreover, it admits initial solutions of domain equations built from these constructions. 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: [Abr83], [Ole85], [Plo82]).

Now algebraic domains are *freely constructed* from their bases, i.e.

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

[2] These descriptions are strictly speaking only valid for algebraic domains, but this will be the only case considered in this paper.

where $\mathsf{Idl}(P)$, for any poset P, is the ideal completion formed by taking all directed, left-closed subsets of P, ordered by inclusion [Plo81, Chapter 6 p. 5]. Thus we can in fact completely describe such categories as **SFP** in an elementary fashion in terms of the bases; various ways of doing this for a certain sub-category of **SFP** (the "Scott domains") 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.

$$\{\perp\} \cup (\{0\} \times (\mathcal{K}(A) \setminus \{\perp_A\})) \cup (\{1\} \times (\mathcal{K}(B) \setminus \{\perp_B\}))$$

with the ordering defined by

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

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

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

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

All these definitions are valid for *any* algebraic domain. Since ω**ALG** is not cartesian closed, we shall describe the function space construction for its largest cartesian closed sub-category, **SFP**. Our description will follow [Gun85,Gun87].

Definition 2.2.3 (i) ([Plo81, Chapter 6 p. 1]). Let A, B be algebraic domains. For $a \in \mathcal{K}(A)$, $b \in \mathcal{K}(B)$,

$$(a \searrow b) : A \to B$$

is the *one-step* function defined by

$$(a \searrow b)d = \begin{cases} b & \text{if } a \sqsubseteq d \\ \perp & \text{otherwise} \end{cases}$$

(ii) ([Gun85,Gun87]). $u \subseteq^{\mathsf{fne}} \mathcal{K}(A) \times \mathcal{K}(B)$ is *joinable* (notation: $\triangle(u)$) iff for all $a \in \mathcal{K}(A)$, $\{(a', b') \in u : a' \sqsubseteq a\}$ has a maximum in $\mathcal{K}(A) \times \mathcal{K}(B)$.

(iii) Given $\triangle(u)$, define $\lceil u \rceil : A \to B$ by

$$\lceil u \rceil(d) = \max\{b : \exists a.\, (a,b) \in u \;\&\; a \sqsubseteq d\}.$$

Note that Plotkin writes $(a \Rightarrow b)$ for $(a \searrow b)$.

Proposition 2.2.4 *([Gun87, Theorem 13]). Let A, B be* **SFP** *domains, and $u \subseteq^{\mathsf{fne}} \mathcal{K}(A) \times \mathcal{K}(B)$ such that $\triangle(u)$.*
(i) $\lceil u \rceil$ is a continuous function, and a finite element of $(A \to B)$. In fact, we have

$$\lceil u \rceil = \bigsqcup \{(a \searrow b) : (a, b) \in u\}.$$

(ii) $\lceil u \rceil \sqsubseteq \lceil v \rceil \iff \forall (a, b) \in u. \exists (a', b') \in v. \, a' \sqsubseteq a \;\&\; b \sqsubseteq b'$.
(iii) $(A \to B)$ is algebraic, and in fact an **SFP** *domain, with basis given by*

$$\mathcal{K}(A \to B) = \{\lceil u \rceil : u \subseteq^{\mathsf{fne}} \mathcal{K}(A) \times \mathcal{K}(B) \;\&\; \triangle(u)\}.$$

The definition of $\triangle(u)$ has the unwelcome feature of a universal quantification over $\mathcal{K}(A)$. The following Proposition (which is new) gives an alternative, more "local" (and hence more easily effectivized) description.

Notation. Given a set X of ordered pairs, we write $(X)_1$ ($(X)_2$) for the set of first (second) components of elements of X.

Proposition 2.2.5 *Let A, B be coherent algebraic domains, $u \subseteq^{\mathsf{fne}} \mathcal{K}(A) \times \mathcal{K}(B)$. Then $\triangle(u)$ iff*

$$\forall v \subseteq u. \, \exists w \subseteq u. \, (w)_1 = \mathsf{MUB}((v)_1) \;\&\; [\forall b \in (v)_2, b' \in (w)_2. \, b \sqsubseteq b']. \tag{2.1}$$

PROOF. $(\Rightarrow)$. Consider $v \subseteq u$. For each $m \in \mathsf{MUB}((v)_1)$, $\triangle(u)$ implies that

$$z_m = \{(a', b') \in u : a' \sqsubseteq m\}$$

has a maximum element (a_m, b_m). Let $w = \{(a_m, b_m) : m \in \mathsf{MUB}((v)_1)\}$. Since $v \subseteq z_m$, $a_m \in \mathsf{UB}((v)_1)$ for each m. Also, $a_m \sqsubseteq m$; hence by minimality of m, $a_m = m$, and $(w)_1 = \mathsf{MUB}((v)_1)$. Finally, $v \subseteq z_m$ implies $b_m \in \mathsf{UB}((v)_2)$ for each m.

$(\Leftarrow)$. Given $a \in \mathcal{K}(A)$, consider the set

$$v = \{(a', b') \in u : a' \sqsubseteq a\}.$$

For some $w \subseteq u$, $(w)_1 = \mathsf{MUB}((v)_1)$. Since $a \in \mathsf{UB}((v)_1)$, for some $(a', b') \in w$, $a' \sqsubseteq a$. Hence $(a', b') \in v$. Moreover, $b' \in \mathsf{UB}((v)_2)$. Thus (a', b') is the required maximum for v. ∎

We conclude this section by reviewing some notions on *embeddings*, which play an important rôle in the standard category-theoretic account 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

$$p \circ e = \mathsf{id}_D, \quad e \circ p \sqsubseteq \mathsf{id}_E.$$

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. It is standard that embeddings are strict, order-reflecting (i.e. $e(x) \sqsubseteq e(y) \Rightarrow x \sqsubseteq y$), and carry finite elements to finite elements. Moreover, since they are left adjoints they preserve all joins. The following Proposition (which I have not found in the literature) is simple but useful; it says that the formation of bases for the various type constructions commutes with embeddings.

Proposition 2.2.6 *Let $e_i : A_i \to B_i$ $(i = 1, 2)$, $e : A \to B$ be embeddings. Then:*

$$
\begin{array}{rrcl}
(i) & (e_1 \times e_2)((a, b)) & = & (e_1(a), e_2(b)) \\
(ii) & (e_1 \to e_2)(\lceil u \rceil) & = & \lceil \{(e_1(a), e_2(b)) : (a, b) \in u\} \rceil \\
(iii) & \mathcal{P}(e)(\mathsf{Con}(\{x_1, \ldots, x_n\})) & = & \mathsf{Con}(\{e(x_1), \ldots, e(x_n)\}) \\
(iv) & (e_1 \oplus e_2)((i, d)) & = & (i, e_{i+1}(d)) \ (i = 0, 1) \\
(v) & (e)_\perp((0, a)) & = & (0, e(a))
\end{array}
$$

PROOF. We shall verify (ii). Firstly, since embeddings preserve joins, by Proposition 2.2.4 it suffices to verify that $(e_1 \to e_2)((a \searrow b)) = (e_1(a) \searrow e_2(b))$. By definition ([SP82]), $(e_1 \to e_2)(f) = e_2 \circ f \circ e_1^R$; while e_1 left adjoint to e_1^R means that

$$e_1(a) \sqsubseteq d \iff a \sqsubseteq e_1^R(d).$$

Now

$$
\begin{array}{rcl}
(e_1 \to e_2)((a \searrow b))(d) & = & \left\{ \begin{array}{ll} e_2(b), & a \sqsubseteq e_1^R(d) \\ \perp & \text{otherwise} \end{array} \right. \\
 & = & \left\{ \begin{array}{ll} e_2(b), & e_1(a) \sqsubseteq d \\ \perp & \text{otherwise} \end{array} \right. \\
 & = & (e_1(a) \searrow e_2(b))(d). \quad \blacksquare
\end{array}
$$

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

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

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 $f : A \to B$ in **Loc**, we write $f^\star$ for the corresponding morphism $B \to A$ in **Frm**.

Now we can define a functor

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

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 $f : A \to \mathbf{2}$, 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 $(a \in A)$:

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

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:

$$\frac{A \xleftarrow{f^\star} B}{\mathsf{Pt}(A) \xrightarrow{\mathsf{Pt}(f)} \mathsf{Pt}(B)} \qquad \mathsf{Pt}(f)x = \{b : f^\star b \in x\}.$$

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

$$\eta_X : X \to \mathsf{Pt}(\Omega(X)) \qquad \eta_X(x) = \{U : x \in U\}$$

$$\epsilon_A : \Omega(\mathsf{Pt}(A)) \to A \qquad \epsilon_A^\star(a) = \{x : a \in x\}.$$

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 dis-junctions—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 paper.)*
(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

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

where **AlgLat** *is the full sub-category of* **CohSp** *of algebraic lattices with the Scott topology.*

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 disjunctions. 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[3]. 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

[3]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.

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

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

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,

$$\text{geometric logic} = \text{observational logic}.$$

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. We have already seen that domains can be viewed topologically, via the Scott topology.

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 2.4.1 *([Plo81, Chapter 1 p. 16]). Let D be a dcpo. 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.
We now relate domains to coherent spaces.

Theorem 2.4.2 (The 2/3 SFP Theorem) *([Plo81, Chapter 8 p. 41], [Gun85, Theorem 4.19]). An algebraic domain is coherent as a space iff it is "2/3 SFP" in the terminology of (loc. cit.), i.e. satisfies property M as defined in Section 2.2. By Stone duality for coherent spaces (Theorem 2.3.3), any such domain D satisfies*

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

This justifies our terminology for such domains as *coherent algebraic*. Thus **SFP** is a category 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 *coprime* if $a \leq \bigvee_{i \in I} b_i$, I finite, implies $a \leq b_i$ for some $i \in I$. Writing $cpr(A)$ for the set of coprimes of A, we have

$$\mathcal{K}(D) = (cpr(A))^{\mathsf{op}}, \quad A \cong \mathsf{L}((\mathcal{K}(D))^{\mathsf{op}}). \tag{2.2}$$

(The fact that the latter construction produces a distributive lattice even though $\mathcal{K}(D)$ is not a meet-semilattice follows from the property M axiom for coherent algebraic domains).

Proposition 2.4.3 *([Gun85, Lemma 4.18]). Suppose X, Y are subsets of a poset P. Then Y is a complete set of upper bounds for X iff:*

$$\bigcap_{x \in X} \uparrow(x) = \bigcup_{y \in Y} \uparrow(y). \tag{2.3}$$

Moreover, if Y is finite and (2.3) holds, then it contains a complete set of minimal upper bounds for X.

PROOF. ($\Rightarrow$). If $d \in \bigcap_{x \in X} \uparrow(x)$ then $d \in \mathsf{UB}(X)$, and so for some $y \in Y$, $y \sqsubseteq d$, and $d \in \bigcup_{y \in Y} \uparrow(y)$. If $d \sqsupseteq y$, $y \in Y$, then d is an upper bound for X, hence $d \in \bigcap_{x \in X} \uparrow(x)$.

($\Leftarrow$). Firstly, $\bigcup_{y \in Y} \uparrow(y) \subseteq \bigcap_{x \in X} \uparrow(x)$ implies $Y \subseteq \mathsf{UB}(X)$, while the converse inclusion implies that Y contains a complete set of upper bounds for X. If Y is finite, it must contain a complete minimal subset, which will be a complete set of minimal upper bounds for X. ∎

The significance of this proposition is that it shows how to turn the MUB axioms for coherent algebraic and SFP domains into logical axioms about their lattices of compact-open subsets. It also shows how to translate the description of $\triangle(u)$ given in Proposition 2.2.5 into an appropriate axiom in our logical treatment of function spaces.

Definition 2.4.4 ([Gun85, Theorem 4.19]). We say that a subset $X \subseteq A$ of a lattice is *quasi-conjunctively closed* if for every $u \subseteq^{\mathsf{f}} X$, for some $v \subseteq^{\mathsf{f}} X$,

$$\bigwedge u = \bigvee v.$$

X is a *quasi-conjunctive closure* of Y, $X, Y \subseteq A$, if $Y \subseteq X$ and X is quasi-conjunctively closed.

The following result gives a purely lattice-theoretic description of those distributive lattices which arise as the compact opens of **SFP** domains.

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

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

Moreover, $\mathsf{Spec}(A)$ is **SFP** *iff it also satisfies:*

$$(3)\quad \forall u \subseteq^{\mathsf{f}} cpr(A). \exists v \subseteq^{\mathsf{f}} cpr(A).\, u \subseteq v \;\&\; v \textit{ is quasi-conjunctively closed.}$$

Of these, (1) ensures the existence of a bottom point, and (2) says "there are enough coprimes". (3) translates the SFP property of finite MUB-closure into finite quasi-conjunctive closure as indicated by Proposition 2.3.

PROOF. We shall show that $\mathsf{Spec}(A)$ is an algebraic domain under the specialisation ordering (which taking filters concretely as subsets of A is just subset inclusion); and moreover that the Scott topology on this domain is precisely $\Omega(\mathsf{Spec}(A))$. The remainder of the Proposition then follows from [Gun85, Theorem 4.19].

Consider the poset $P = (\mathsf{Spec}(A), \subseteq)$. It is easily verified that prime filters are closed under directed unions, so P is a dcpo. Moreover, for each coprime $a \in A$, the principal filter $\uparrow_A(a)$ is prime (justifying our terminology, since $\uparrow(\cdot)$ is contravariant), and hence in $\mathsf{Spec}(A)$. It follows from our description of directed joins that each such principal filter is a finite element of P; in particular, since 1_A is coprime, $\uparrow(1_A)$ is a bottom element for P. Denote the set of all such prime principal filters by B. Now given $x \in \mathsf{Spec}(A)$, consider $S = B \cap \downarrow_P(x)$. Firstly, S is directed: Given $\uparrow(a), \uparrow(b) \in S$, we have $a \wedge b = \bigvee_{i \in I} c_i$ for some finite disjunction of coprimes c_i. Now

$$\begin{array}{rcll}
\uparrow(a), \uparrow(b) \subseteq x & \Rightarrow & a, b \in x & \\
& \Rightarrow & a \wedge b \in x & x \text{ is a filter} \\
& \Rightarrow & \exists i \in I.\, c_i \in x & x \text{ is prime} \\
& \Rightarrow & \uparrow(c_i) \in S &
\end{array}$$

and $c_i \leq \bigvee_{i \in I} c_i = a \wedge b \leq a, b$, so $\uparrow(a) \subseteq \uparrow(c_i) \supseteq \uparrow(b)$. Also, $x = \bigcup S$, since given $a \in x$, $a = \bigvee_{i \in I} a_i$ with a_i coprime, and so since x is prime, for some i $a_i \in x$, and $a \in \uparrow(a_i) \in S$. Thus B is a basis for P. It follows that *all* finite elements of P are in B, since given finite $x \in P$, $x = \bigcup S$, where $S = B \cap \downarrow(x)$; since x is finite, $x \subseteq \uparrow(a)$ for some $\uparrow(a) \in S$; but $\uparrow(a) \subseteq x$, and so $x = \uparrow(a)$.

We now show that $\Omega(\mathsf{Spec}(A)) = \sigma(P)$, the Scott topology on P. We know that basic open sets in $\sigma(P)$ have the form $\uparrow_P(\uparrow_A(a))$, for coprime $a \in A$; but clearly $\uparrow(a) \subseteq x \Leftrightarrow a \in x$, and so $\uparrow_P(\uparrow_A(a))$ is the basic open set U_a in $\Omega(\mathsf{Spec}(A))$. Conversely, given a basic open set U_a in $\Omega(\mathsf{Spec}(A))$, we have $a = \bigvee_{i \in I} a_i$, a_i coprime, and

$$U_a = U_{\bigvee_{i \in I} a_i} = \bigcup_{i \in I} U_{a_i} = \bigcup_{i \in I} \uparrow_P(\uparrow_A(a_i)),$$

and so each U_a is in $\sigma(P)$. Thus the two topologies coincide. ∎

Chapter 3

Domain Pre-Locales

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 **SFP**. 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 algebraic prelocale* is a structure

$$A = (|A|, \leq_A, =_A, 0_A, \vee_A, 1_A, \wedge_A, \mathsf{C}_A, \mathsf{T}_A)$$

where

- $|A|$ is a countable 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 $|A|$
- C_A, T_A are unary predicates on $|A|$

(Notation: $\mathsf{C}(A) \equiv \{a \in |A| : \mathsf{C}_A(a)\}$)

subject to the following axioms (subscripts omitted):

$$(d1) \quad a \leq a \quad \frac{a \leq b \;\; b \leq c}{a \leq c} \quad \frac{a \leq b \;\; b \leq a}{a = b} \quad \frac{a = b}{a \leq b \;\; b \leq a}$$

$$(d2) \quad 0 \leq a \quad \frac{a \leq c \;\; b \leq c}{a \vee b \leq c} \quad a \leq a \vee b \quad b \leq a \vee b$$

$$(d3) \quad a \leq 1 \quad \frac{a \leq b \;\; a \leq c}{a \leq b \wedge c} \quad a \wedge b \leq a \quad a \wedge b \leq b$$

$$(d4) \quad a \wedge (b \vee c) \leq (a \wedge b) \vee (a \wedge c)$$

$$(p1) \quad \frac{\mathsf{C}(a) \;\; a = b}{\mathsf{C}(b)} \quad \mathsf{C}(1)$$

$$(p2) \quad [\mathsf{C}(a) \;\&\; a \leq \bigvee_{i \in I} b_i] \;\Rightarrow\; \exists i \in I.\, a \leq b_i$$

$$(p3) \quad \forall a \in |A|. \exists b_1, \ldots, b_n \in \mathsf{C}(A).\, a =_A \bigvee_{i=1}^{n} b_i$$

$$(t1) \quad \frac{\mathsf{T}(a) \;\; b \leq a}{\mathsf{T}(b)} \quad \frac{\mathsf{T}(a_j) \;\; (j \in I)}{\mathsf{T}(\bigwedge_{i \in I} a_i)} \quad \frac{\{\mathsf{T}(a_i)\}_{i \in I}}{\mathsf{T}(\bigvee_{i \in I} a_i)}$$

$$(t2) \quad \mathsf{T}(a) \iff \neg(a =_A 1)$$

A is an *SFP prelocale* if it also satisfies:

$$(p4) \quad \forall u \subseteq^{\mathsf{f}} \mathsf{C}(A). \exists v \subseteq^{\mathsf{f}} \mathsf{C}(A).\, u \subseteq v \;\&\; [\forall w \subseteq v. \exists z \subseteq v.\, \bigwedge w = \bigvee z]$$

These notions arise naturally from Theorem 2.4.5. We are axiomatizing *pre-orders* rather than partial orders to capture the notion of a logical theory, in which we have syntactically distinct but logically equivalent formulae. Evidently, the quotient structure ("Lindenbaum algebra")

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

is a distributive lattice, with meet and join given by $\wedge/{=_A}$, $\vee/{=_A}$. As we saw in Theorem 2.4.5, the coprimes play a crucial role in coherent algebraic prelocales, and we make them part of the structure. The T predicate is used to facilitate working with bottom elements; its role will not become apparent until we introduce the coalesced sum construction.

Note that our axiomatization is not elementary (first-order); in particular, $(p2)$–$(p4)$ use quantification over finite subsets of $|A|$ (i.e. "weak second order logic" in the terminology of [Bar77]). This is in fact inevitable; the results of [Gun86] can be adapted to show that the lattices of compact-open subsets arising from SFP domains are not first-order definable.

Proposition 3.2.2 *Let A be a coherent algebraic prelocale, $a \in |A|$. Then*

$$[\forall u \subseteq^{\mathsf{f}} |A|.\, a \leq \bigvee u \;\Rightarrow\; \exists b \in u.\, a \leq b] \;\Rightarrow\; \mathsf{C}_A(a).$$

PROOF. By $(p3)$, $a =_A \bigvee_{i\in I} b_i$ $(\mathsf{C}_A(b_i), i \in I)$. Under the hypothesis on a, $a \leq b_i$ for some $i \in I$, and also $b_i \leq a$. Thus $a =_A b_i$ and $\mathsf{C}_A(b_i)$, so by $(p1)$ $\mathsf{C}_A(a)$, as required. ∎

This proposition shows that the C predicate captures exactly the "semantic" notion of coprimeness.

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

Definition 3.2.3 Let A, B, be domain prelocales. An *approximable mapping* $R : A \to B$ is a relation $R \subseteq |A| \times |B|$ satisfying

$$\begin{array}{ll}
(r1) & [\forall i \in I.\, aRb_i] \;\Rightarrow\; aR \bigwedge_{i\in I} b_i \\
(r2) & [\forall i \in I.\, a_iRb] \;\Rightarrow\; \bigvee_{i\in I} a_iRb \\
(r3) & a \leq a'Rb' \leq b \;\Rightarrow\; aRb \\
(r4) & \mathsf{C}_A(a)\,\&\, aR\bigvee_{i\in I} b_i \;\Rightarrow\; \exists i \in I.\, aRb_i
\end{array}$$

Approximable mappimgs are closed under relational composition. We verify the least trivial closure condition, $(r4)$. Suppose $R : A \to B$, $S : B \to C$, $\mathsf{C}_A(a)$ and $a(R;S)\bigvee_{j\in J} c_j$. For some $b \in |B|$, aRb and $bS\bigvee_{j\in J} c_j$. By $(p3)$,

$$b =_B \bigvee_{i\in I} b_i \;\; (\mathsf{C}_A(b_i), i \in I).$$

By $(r4)$, aRb_i for some $i \in I$; and by $(r3)$, $b_iS\bigvee_{j\in J} c_j$. By $(r4)$ again, b_iSc_j for some $j \in J$. Hence $a(R;S)c_j$, as required.

Identities with respect to this composition are given by

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

Hence we can define a category **DPL** ("domain prelocales") of SFP pre-locales and approximable mappings.

Definition 3.2.4 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 3.2.5 *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 3.2.6 **DPL** *is equivalent to* **SFP**.

PROOF. We define functors

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

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

as follows:

$$F(D) = (K\Omega(D), \subseteq, =, \varnothing, \cup, D, \cap, \{\uparrow(b) : b \in \mathcal{K}(D)\}, K\Omega(D) \setminus \{\uparrow(\bot_D)\})$$

i.e. the distributive lattice of compact-open subsets of D, with the expected interpretations of coprimeness and termination.

$$F(f) = R_f,$$

where

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

The verification that F is well-defined on objects is routine in the light of Theorem 2.4.5. To verify $(r4)$ for R_f, note that, for $u \in \mathcal{K}(D)$:

$$\begin{aligned}
\uparrow(u) \subseteq f^{-1}(\bigvee_{i\in I} b_i) \quad &\Leftrightarrow \quad u \in f^{-1}(\bigvee_{i\in I} b_i) \\
&\Leftrightarrow \quad f(u) \in \bigvee_{i\in I} b_i \\
&\Leftrightarrow \quad \exists i \in I.\, f(u) \in b_i \\
&\Leftrightarrow \quad \exists i \in I.\, \uparrow(u) \subseteq f^{-1}(b_i).
\end{aligned}$$

Next, we define

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

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

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

$\hat{A}$ 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,

$$\hat{A} \;\cong\; \mathsf{Spec}(\tilde{A}).$$

Finally, we define

$$G(R) = f_R,$$

where

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

Well-definedness of G on objects again follows directly from Theorem 2.4.5. The fact that $f_R(x)$ is a filter follows from $(r1)$ and $(r3)$. To verify that $f_R(x)$ is prime, suppose $\bigvee_{j\in J} b_j \in f_R(x)$. For some $a \in x$, $aR \bigvee_{j\in J} b_j$. By $(p3)$, $a =_A \bigvee_{i\in I} a_i$ $(\mathsf{C}_A(a_i), i \in I)$. Since x is prime, $a_i \in x$ for some $i \in I$. By $(r3)$, $a_i R \bigvee_{j\in I} b_j$; by $(r4)$, $a_i R b_j$ for some $j \in J$; so $b_j \in f_R(x)$, as required. Directed joins in $\hat{B}$ are just unions, so continuity of $f_R(x)$ is trivial.

The remainder of the verification that F and G are functors is routine.

We now define natural transformations

$$\eta : I_{\mathbf{SFP}} \to GF \qquad \epsilon : I_{\mathbf{DPL}} \to FG$$

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

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

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

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

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 $(|B|, 0_B, \vee_B, 1_B, \wedge_B)$
- $\leq_A \subseteq \leq_B$
- $\mathsf{C}_A \subseteq \mathsf{C}_B$
- $\mathsf{T}_A \subseteq \mathsf{T}_B$.

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

$$\begin{array}{ll} (s1) & A \Subset B \\ (s2) & \leq_A = \leq_B \cap |A|^2 \\ (s3) & \mathsf{C}_A = \mathsf{C}_B \cap |A| \\ (s4) & \mathsf{T}_A = \mathsf{T}_B \cap |A| \end{array}$$

Note that 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 $(p3)$ for illustration.

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

$$a =_{A_n} \bigvee_{i\in I} a_i, \;\; (\mathsf{C}_{A_n}(a_i), i \in I).$$

Clearly $a =_{A_\infty} \bigvee_{i\in I} a_i$; furthermore, $\mathsf{C}(A_n) \subseteq \mathsf{C}(A_\infty)$, hence for all $i \in I$, $\mathsf{C}_{A_\infty}(a_i)$, as required. ∎

Recall that a *cpo* [Plo81] is an ω-chain complete partial order with a least element. 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:

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

Finally, $\leq_{\mathbf{1}}$, $=_{\mathbf{1}}$, $\mathsf{C}_{\mathbf{1}}$, $\mathsf{T}_{\mathbf{1}}$ are defined inductively as the least relations satisfying $(d1)$–$(d4)$, $(p1)$ and $(t1)$. It is easy to see that $\tilde{\mathbf{1}}$ is the two-point lattice, $\mathsf{C}_{\mathbf{1}} = \{t\}$, $\mathsf{T}_{\mathbf{1}} = \{f\}$; 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 embeddings used in the standard category-theoretic treatment of the solution of domain equations [SP82]. (See Chapter 2 for definitions).

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 3.2.6).

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). The argument that e is well defined is similar to that for f_R in the proof of Theorem 3.2.6. 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 : **DPL1** $\rightarrow$ **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

$$B \equiv \bigsqcup_n f(A_n), \qquad C \equiv f(\bigsqcup_n A_n).$$

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

- $G(T(A, B)) \subseteq |T(A, B)|$
- $t, f \in |T(A, B)|$
- $a, b \in |T(A, B)| \;\Rightarrow\; a \wedge b, a \vee b \in |T(A, B)|$

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 $\leq_{T(A,B)}$, $=_{T(A,B)}$, $\mathsf{C}_{T(A,B)}$, $\mathsf{T}_{T(A,B)}$ are defined inductively as the least satisfying $(d1)$–$(d4)$, $(p1)$ and $(t1)$, plus specific axioms on the generators. (Note that our definition of $\mathbf{1}$ in the previous section is the special case of this scheme where the set of generators is empty.)

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 $(d1)$–$(d4)$, $(p1)$, and $(t1)$
- $\mathbf{1} \trianglelefteq T(A, B)$
- $T(A, B) \Subset T(A', B')$
- T is continuous on carriers.

We are left to focus our attention on proving that:

- $T(A, B)$ satisfies $(p2)$–$(p4)$ and $(t2)$
- conditions $(s2)$–$(s4)$ 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 **SFP** corresponding to T, and show that $[\![\cdot]\!]_{T(A,B)}$ is a (pre)isomor-phism; 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: Normal Forms.

$$\forall a \in |T(A,B)|.\, \exists b_1, \ldots, b_n \in \mathsf{C}(T(A,B)).\, a =_{T(A,B)} \bigvee_{i=1}^{n} b_i.$$

T2: Soundness. *For all $a, b \in |T(A,B)|$:*

- $a \leq_{T(A,B)} b \;\Rightarrow\; [\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)}$
- $\mathsf{C}_{T(A,B)}(a) \;\Rightarrow\; \exists b \in \mathcal{K}(F_T(\hat{A},\hat{B})).\, [\![a]\!]_{T(A,B)} = \uparrow(b)$
- $\mathsf{T}_{T(A,B)}(a) \;\Rightarrow\; \perp_{F_T(\hat{A},\hat{B})} \not\in [\![a]\!]_{T(A,B)}$.

T3: Coprime Completeness. *For all $a, b \in \mathsf{C}(T(A,B))$:*

- $[\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)} \;\Rightarrow\; a \leq_{T(A,B)} b$
- $\perp_{F_T(\hat{A},\hat{B})} \not\in [\![a]\!]_{T(A,B)} \;\Rightarrow\; \mathsf{T}_{T(A,B)}(a)$.

T4: Definability.

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

T5: 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 $a \in \mathsf{C}(T(A,B))$, let*

$$[\![a]\!]_{T(A,B)} = \uparrow(u), \quad [\![a]\!]_{T(A',B')} = \uparrow(v), \qquad (u \in \mathcal{K}(F_T(\hat{A},\hat{B})), v \in \mathcal{K}(F_T(\hat{A}',\hat{B}'))).$$

(This is well-defined by (T2)). Then:

$$F_T(e_1, e_2)(u) = v.$$

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

T6: 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 (T1),

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

with $a_i, b_j \in \mathsf{C}(T(A,B))$ $(i \in I, j \in J)$. By (T2),

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

where

$$[\![a_i]\!]_{T(A,B)} = \uparrow(u_i), \;\; [\![b_j]\!]_{T(A,B)} = \uparrow(v_j) \quad (u_i, v_j \in \mathcal{K}(F_T(\hat{A},\hat{B})) \quad (i \in I, j \in J).$$

Now,

$$\begin{array}{lll}
\bullet & [\![a]\!]_{T(A,B)} \subseteq [\![b]\!]_{T(A,B)} & \\
\Longrightarrow & \bigcup_{i \in I} \uparrow(u_i) \subseteq \bigcup_{j \in J} \uparrow(v_j) & \\
\Longrightarrow & \forall i \in I.\, \exists j \in J.\, \uparrow(u_i) \subseteq \uparrow(v_j) & \\
\Longrightarrow & \forall i \in I.\, \exists j \in J.\, a_i \leq_{T(A,B)} b_j & \text{by (T3)} \\
\Longrightarrow & \bigvee_{i \in I} a_i \leq_{T(A,B)} \bigvee_{j \in J} b_j & \text{by (d2)} \\
\Longrightarrow & a \leq_{T(A,B)} b & \text{by (d1).} \blacksquare
\end{array}$$

(T7): Stone Duality. *$T(A,B)$ is the Stone dual of $F_T(\hat{A},\hat{B})$, i.e.*

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

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

PROOF. (i) and (ii) are equivalent since SFP domains are coherent spaces. (ii) is an immediate consequence of (T2), (T4) and (T6). ∎

(T8). *T is a well defined, $\trianglelefteq$-monotonic and continuous operation on* **DPL1**.

PROOF. T(A,B) is an SFP prelocale by (T7), since $K\Omega(F_T(\hat{A},\hat{B}))$ is. Given $A \trianglelefteq A'$, $B \trianglelefteq B'$, with corresponding embeddings e_1, e_2, we must verify $(s2)$–$(s4)$ to show that $T(A,B) \trianglelefteq T(A',B')$.

To verify $(s2)$, it is sufficient to show that for $a, b \in \mathsf{C}(T(A,B)$:

$$a \leq_{T(A',B')} b \;\Rightarrow\; a \leq_{T(A,B)} b.$$

Let

$$[\![a]\!]_{T(A,B)} = \uparrow(u), \;\; [\![b]\!]_{T(A,B)} = \uparrow(v), \;\; [\![a]\!]_{T(A',B')} = \uparrow(u'), \;\; [\![b]\!]_{T(A',B')} = \uparrow(v').$$

By (T5),

$$u' = F_T(e_1,e_2)(u), \quad v' = F_T(e_1,e_2)(v).$$

Now $a \leq_{T(A',B')} b$ implies $v' \sqsubseteq u'$; since embeddings are order-reflecting, this implies $v \sqsubseteq u$, and hence $a \leq_{T(A,B)} b$.

To verify $(s3)$, we use $(s2)$ and Proposition 3.2.2. Finally, to verify $(s4)$ we use $(t2)$.

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

(T9). *With notation as in (T5), let $C = T(A,B)$, $C' = T(A',B')$, and e be the embedding induced by $T(A,B) \trianglelefteq T(A',B')$ (this is well-defined by (T8)). By (T4), we have isomorphisms*

$$C \cong K\Omega(F_T(\hat{A},\hat{B})), \quad C' \cong K\Omega(F_T(\hat{A}',\hat{B}'))$$

which are carried under the functor G of Theorem 3.2.6 into isomorphisms

$$\hat{C} \cong \mathsf{Spec}(K\Omega(F_T(\hat{A},\hat{B}))), \quad \hat{C}' \cong \mathsf{Spec}(K\Omega(F_T(\hat{A},\hat{B}))).$$

Moreover, by Stone duality we have isomorphisms

$$\mathsf{Spec}(K\Omega(F_T(\hat{A},\hat{B}))) \cong F_T(\hat{A},\hat{B}), \quad \mathsf{Spec}(K\Omega(F_T(\hat{A}',\hat{B}'))) \cong F_T(\hat{A}',\hat{B}').$$

Composition yields isomorphisms

$$\eta_{T(A,B)} : \hat{C} \cong F_T(\hat{A},\hat{B}), \quad \eta_{T(A',B')} : \hat{C}' \cong F_T(\hat{A}',\hat{B}').$$

These isomorphisms are natural *in A and B:*

$$F_T(e_1,e_2) \circ \eta_{T(A,B)} = \eta_{T(A',B')} \circ e. \tag{3.1}$$

PROOF. It suffices to show that the two functions agree on finite elements of $\hat{C}$. These have the form $\uparrow_C(a)$, $\mathsf{C}_{T(A,B)}(a)$. Let

$$[\![a]\!]_{T(A,B)} = \uparrow(u), \;\; [\![a]\!]_{T(A',B')} = \uparrow(v) \quad (u \in \mathcal{K}(F_T(\hat{A},\hat{B})), v \in \mathcal{K}(F_T(\hat{A}',\hat{B}'))).$$

Then $\eta_{T(A,B)}(\uparrow_C(a)) = u$, $\eta_{T(A',B')}(\uparrow_{C'}(a)) = v$, and (3.1) follows from (T5). ∎

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 3.2.6.

Definition 3.4.1 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 relations $\leq_{A\to B}$, $=_{A\to B}$, $\mathsf{C}_{A\to B}$, $\mathsf{T}_{A\to B}$ are then defined inductively by the following axioms and rules in addition to $(d1)$–$(d4)$, $(p1)$ and $(t1)$ (subscripts omitted):

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

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

$$(\to - \vee - R) \qquad \frac{\mathsf{C}_A(a)}{(a \to \bigvee_{i\in I} b_i) = \bigvee_{i\in I}(a \to b_i)}$$

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

$$(\mathsf{C} - \to) \qquad \frac{\{\mathsf{C}_A(a_i)\}_{i\in I} \quad \{\mathsf{C}_B(b_i)\}_{i\in I} \quad \forall J \subseteq I.\, \exists K \subseteq I.\, [\bigwedge_{j\in J} a_j =_A \bigvee_{k\in K} a_k \,\&\, [\forall j \in J, k \in K.\, b_k \leq_B b_j]]}{\mathsf{C}(\bigwedge_{i\in I}(a_i \to b_i))}$$

$$(\mathsf{T} - \to) \qquad \frac{\mathsf{C}_A(a') \quad a' \leq a \quad \mathsf{T}_B(b)}{\mathsf{T}(a \to b)}$$

(iii) 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 calculations needed to show that $[\![(a \to b)]\!]_{A\to B}$ is compact can be found in the proofs of the following Propositions). 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)–(T5) in a sequence of propositions.

Proposition 3.4.2 (T1)

$$\forall a \in |A \to B|.\, \exists b_1, \ldots, b_n \in \mathsf{C}(A \to B).\, a =_{A \to B} \bigvee_{i=1}^{n} b_i.$$

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

$$\bigvee_{i \in I_0} \bigwedge_{j \in J_i} (a_{ij} \to b_{ij}).$$

By $(p3)$, each a_{ij} is equal to

$$\bigvee_{k \in K_{ij}} c_k, \quad (\mathsf{C}_A(c_k), k \in K_{ij}),$$

and each b_{ij} is equal to

$$\bigvee_{l \in L_{ij}} d_l, \quad (\mathsf{C}_B(d_l), l \in L_{ij}).$$

Now

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

Using the distributive lattice laws again, we obtain a disjunction of terms of the form:

$$\bigwedge_{i \in I} (a_i \to b_i) \quad (\mathsf{C}_A(a_i), \mathsf{C}_B(b_i), i \in I). \tag{3.2}$$

We are only prevented from inferring $\mathsf{C}_{A \to B}$ for such an expression by failure of the final premise of the rule $(\mathsf{C} - \to)$:

$$\forall J \subseteq I.\, \exists K \subseteq I.\, [\bigwedge_{j \in J} a_j =_A \bigvee_{k \in K} a_k \,\&\, [\forall j \in J, k \in K.\, b_k \leq_B b_j]]. \tag{3.3}$$

We shall give an algorithm for progressively removing failures of (3.3). At each stage k, we have a disjunction θ_k of terms of the form (3.2). If (3.3) does not fail for any of these terms, the algorithm terminates. Otherwise, we select a disjunct (3.2) of θ_k, and $J \subseteq I$, such that (3.3) fails, and replace the disjunct by $c \equiv \bigvee_{l \in L} c_l$ such that

$$\bigwedge_{i \in I} (a_i \to b_i) =_{A \to B} c \tag{3.4}$$

to form θ_{k+1}. When the algorithm terminates, we have the required normal form. Thus we have to do two things:

1. specify c (given a term (3.2) and $J \subseteq I$ as above), and verify (3.4)

2. prove termination.

1. Firstly, we have $\bigwedge_{j\in J} a_j =_A \bigvee_{m\in M} d_m$, for some $\{d_m\}_{m\in M}$ with $\mathsf{C}_A(d_m)$, $(m \in M)$. Similarly, $\bigwedge_{j\in J} b_j =_B \bigvee_{n\in N} e_n$. Now,

$$\begin{array}{rcll}
\bigwedge_{i\in I}(a_i \to b_i) & = & \bigwedge_{i\in I}(a_i \to b_i) \wedge \bigwedge_{j\in J}(\bigwedge_{j\in J} a_j \to b_j) & (\to - \leq) \\
& = & \bigwedge_{i\in I}(a_i \to b_i) \wedge (\bigwedge_{j\in J} a_j \to \bigwedge_{j\in J} b_j) & (\to - \wedge) \\
& = & \bigwedge_{i\in I}(a_i \to b_i) \wedge (\bigvee_{m\in M} d_m \to \bigvee_{n\in N} e_n) & (\to - \leq) \\
& = & \bigwedge_{i\in I}(a_i \to b_i) \wedge \bigwedge_{m\in M}(d_m \to \bigvee_{n\in N} e_n) & (\to - \vee - L) \\
& = & \bigwedge_{i\in I}(a_i \to b_i) \wedge \bigwedge_{m\in M} \bigvee_{n\in N}(d_m \to e_n) & (\to - \vee - R) \\
& = & \bigvee_{f:M\to N}(\bigwedge_{i\in I}(a_i \to b_i) \wedge \bigwedge_{m\in M}(d_m \to e_{f(m)})) & \text{distributive laws}
\end{array}$$

We take the final term in this derivation as c to form θ_{k+1}.

2. To prove termination, we firstly assign a measure $\|\theta\|$ to "states" of the algorithm as follows: $\|\theta\| = (n, m)$, where n is the minimum number of distinct conjuncts in any disjunct (3.2) of θ which does not satisfy (3.3), and m is the number of disjuncts attaining this minimum. (If there are no such disjuncts then we are in a final state). We order $\omega \times \omega$ by:

$$(n, m) \succ (n', m') \equiv n < n' \text{ or } n = n' \;\&\; m > m'.$$

Clearly $\succ$ is irreflexive and transitive. Next, we note that, if the term (3.2) is chosen minimal in forming θ_{k+1}, then

$$\theta_k \to \theta_{k+1} \implies \|\theta_k\| \succ \|\theta_{k+1}\|$$

since each c_l in c has a greater number of conjuncts than the formula (3.2) it is replacing, as M must be non-empty if $J \subseteq I$ violates (3.3). It follows that states are never repeated as we execute the algorithm.

We complete the argument by showing that the algorithm can be executed in such a way that the possible state-space for a given initial state θ_0 is *finite*. Given the set G of all generators $(a \to b)$ occurring in disjuncts (3.2) of θ_0, let U be the set of all a with $(a \to b) \in G$, and V the set of all b with $(a \to b) \in G$. Since A and B are SFP prelocales, by $(p4)$ we can find finite quasi-conjunctive closures $U' \supseteq U$, $V' \supseteq V$. The states θ_k can then be constructed as disjunctions of conjunctions of generators $(a \to b)$ with $a \in U'$ and $b \in V'$. ∎

It may be noted that the last step in this proof is the only place where the SFP property $(p4)$ is used. The SFP axiom has always been hard to justify conceptually, and its rather natural appearance in the termination argument of a normal form algorithm yields a novel perpective on it. (Note that the algorithm makes sense, and the partial correctness argument (1) is valid, for coherent algebraic prelocales).

Proposition 3.4.3 (T2) *For all $a, b \in |A \to B|$:*

- $a \leq_{A\to B} b \;\Rightarrow\; [\![a]\!]_{A\to B} \subseteq [\![b]\!]_{A\to B}$
- $\mathsf{C}_{A\to B}(a) \;\Rightarrow\; \exists b \in \mathcal{K}(\hat{A} \to \hat{B}).\, [\![a]\!]_{A\to B} = \uparrow(b)$
- $\mathsf{T}_{A\to B}(a) \;\Rightarrow\; \perp_{\hat{A}\to\hat{B}} \notin [\![a]\!]_{A\to B}$.

PROOF. $[\![\cdot]\!]_{A \to B}$ preserves meets and joins by definition, and $(d1)$–$(d4)$, $(t1)$, $(p1)$ are valid in any coherent algebraic prelocale. $(\mathsf{C} - \to)$ is sound by Proposition 2.2.5. 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 which validate $(\to - \leq)$, $(\to - \wedge)$ and $(\to - \vee - L)$. Next, suppose $\mathsf{C}_A(a)$. Then $[\![a]\!]_A = \uparrow(u)$ with $u \in \mathcal{K}(\hat{A})$, 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. Finally, assume the premises of $(\mathsf{T} - \to)$. Then $[\![a]\!]_A \neq \varnothing$, $\perp \notin [\![b]\!]_B$, and so $\perp_{\hat{A} \to \hat{B}} \notin [\![(a \to b)]\!]_{A \to B}$. ∎

Proposition 3.4.4 (T3) *For all $a, b \in \mathsf{C}(A \to B)$:*

- $[\![a]\!]_{A \to B} \subseteq [\![b]\!]_{A \to B} \implies a \leq_{A \to B} b$
- $\perp_{\hat{A} \to \hat{B}} \notin [\![a]\!]_{A \to B} \implies \mathsf{T}_{A \to B}(a)$.

PROOF. By induction on the proofs that $\mathsf{C}_{A \to B}(a)$, $\mathsf{C}_{A \to B}(b)$. The non-trivial case is when these are inferred by $(\mathsf{C} - \to)$, with $a \equiv \bigwedge_{i \in I} (a_i \to b_i)$, $b \equiv \bigwedge_{j \in J} (a_j \to b_j)$.

Let $[\![a_i]\!]_A = \uparrow(u_i)$, $[\![b_i]\!]_B = \uparrow(v_i)$, $[\![a_j]\!]_A = \uparrow(u_j)$, $[\![b_j]\!]_B = \uparrow(v_j)$. Let $u = \{(u_i, v_i) : i \in I\}$, $v = \{(u_j, v_j) : j \in J\}$. Now

$$\begin{aligned}
\bullet\quad & [\![a]\!]_{A \to B} \subseteq [\![b]\!]_{A \to B} \\
\Rightarrow\quad & \lceil v \rceil \sqsubseteq \lceil u \rceil \\
\Rightarrow\quad & \forall j \in J. \exists i \in I. u_i \sqsubseteq u_j \;\&\; v_j \sqsubseteq v_i \\
\Rightarrow\quad & \forall j \in J. \exists i \in I. a_j \leq_A a_i \;\&\; b_i \leq_B b_j \\
\Rightarrow\quad & \forall j \in J. \exists i \in I. (a_i \to b_i) \leq_{A \to B} (a_j \to b_j) \quad \text{by } (\to - \leq) \\
\Rightarrow\quad & a \leq_{A \to B} b \qquad\qquad\qquad\qquad\qquad\qquad \text{by } (d3)
\end{aligned}$$

Again,

$$\begin{aligned}
\perp_{\hat{A} \to \hat{B}} \notin [\![a]\!]_{A \to B} &\Rightarrow \exists i \in I. \perp_{\hat{B}} \notin [\![b_i]\!]_{\hat{B}} \\
&\Rightarrow \mathsf{T}_{A \to B}(a)
\end{aligned}$$

by $(\mathsf{T}-\rightarrow)$ and $(\mathsf{T}-\wedge)$; the first two premises of $(\mathsf{T}-\rightarrow)$ are trivially verified, since $\mathsf{C}_A(a_i)$ by assumption. ∎

Proposition 3.4.5 (T4) $\forall u \in \mathcal{K}([\hat{A} \rightarrow \hat{B}]). \exists a \in |A \rightarrow B|. [\![a]\!]_{A\rightarrow B} = \uparrow(u)$.

PROOF. By Proposition 2.2.4, finite elements of $[\hat{A} \rightarrow \hat{B}]$ have the form $\lceil u \rceil$, where $u = \{(u_i, v_i) : i \in I\}$. There are $a_i \in |A|$, $b_i \in |B|$ with $[\![a_i]\!]_A = \uparrow(u_i)$, $[\![b_i]\!]_B = \uparrow(v_i)$, $i \in I$. Now

$$\begin{array}{rcl} [\![\bigwedge_{i\in I}(a_i \rightarrow b_i)]\!]_{A\rightarrow B} & = & \bigcap_{i\in I}[\![(a_i \rightarrow b_i)]\!]_{A\rightarrow B} \\ & = & \bigcap_{i\in I}\uparrow((u_i \searrow v_i)) \\ & = & \uparrow(\bigsqcup_{i\in I}(u_i \searrow v_i)) \\ & = & \uparrow(\lceil u \rceil). \end{array}$$ ∎

Proposition 3.4.6 (T5) *Given $A \trianglelefteq A'$, $B \trianglelefteq B'$, let $e_1 : \hat{A} \rightarrow \hat{A}'$, $e_2 : \hat{B} \rightarrow \hat{B}'$ be the corresponding embeddings. Given $a \in \mathsf{C}(A \rightarrow B)$, let $[\![a]\!]_{A\rightarrow B} = \uparrow(u)$, $[\![a]\!]_{A\rightarrow B} = \uparrow(v)$. Then*

$$(e_1 \rightarrow e_2)(u) = v.$$

PROOF. By induction on the proof that $\mathsf{C}_{A\rightarrow B}(a)$. The non-trivial case is when $a \equiv \bigwedge_{i\in I}(a_i \rightarrow b_i)$, and $\mathsf{C}_{A\rightarrow B}(a)$ is inferred by $(\mathsf{C}-\rightarrow)$. Let $[\![a_i]\!]_A = \uparrow(u_i)$, $[\![b_i]\!]_B = \uparrow(v_i)$, $[\![a_i]\!]_{A'} = \uparrow(u'_i)$, $[\![b_i]\!]_{B'} = \uparrow(v'_i)$, $(i \in I)$. Note that $e_1(u_i) = u'_i$, $e_2(v_i) = v'_i$, $(i \in I)$. Now

$$\begin{array}{rcll} (e_1 \rightarrow e_2)(u) & = & \bigsqcup_{i\in I}(e_1(u_i) \searrow e_2(v_i)) & \text{Proposition 2.2.6} \\ & = & \bigsqcup_{i\in I}(u'_i \searrow v'_i) & \\ & = & v. & \end{array}$$ ∎

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

Definition 3.4.7 The *Plotkin powerdomain* $\mathcal{P}(A)$.
(i) The generators:

$$G(\mathcal{P}(A)) \equiv \{\Box a : a \in |A|\} \cup \{\Diamond a : a \in |A|\}$$

(ii) Axioms in addition to $(d1)$ – $(d4)$, $(p1)$ and $(t1)$:

$$\begin{array}{ll} (\Box - \wedge) & \Box \bigwedge_{i\in I} a_i = \bigwedge_{i\in I} \Box a_i \\ (\Diamond - \vee) & \Diamond \bigvee_{i\in I} a_i = \bigvee_{i\in I} \Diamond a_i \\ (\Box - \vee) & \Box(a \vee b) \leq \Box a \vee \Diamond b \\ (\Diamond - \wedge) & \Box a \wedge \Diamond b \leq \Diamond(a \wedge b) \\ (\Box - 0) & \Box 0 = 0 \end{array}$$

Rules:

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

$$(\mathsf{C} - \Box - \Diamond) \quad \frac{\{\mathsf{C}_A(a_i)\}_{i \in I} \quad (I \neq \varnothing)}{\mathsf{C}(\Box \bigvee_{i \in I} a_i \wedge \bigwedge_{i \in I} \Diamond a_i)}$$

$$(\mathsf{T} - \Box) \quad \frac{\mathsf{T}_A(a)}{\mathsf{T}(\Box a)} \qquad (\mathsf{T} - \Diamond) \quad \frac{\mathsf{T}_A(a)}{\mathsf{T}(\Diamond a)}$$

(iii) The semantic function:

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

$$\begin{aligned} [\![\Box a]\!]_{\mathcal{P}(A)} &= \{S \in \mathcal{P}(\hat{A}) : S \subseteq [\![a]\!]_A\} \\ [\![\Diamond a]\!]_{\mathcal{P}(A)} &= \{S \in \mathcal{P}(\hat{A}) : S \cap [\![a]\!]_A \neq \varnothing\} \end{aligned}$$

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

The following equations can be derived from the above axiomatization:

$$\begin{array}{llcl} (\Diamond - 1) & \Diamond 1 & = & 1 \\ (D1) & \Box(a \vee b) & = & \Box a \vee (\Box(a \vee b) \wedge \Diamond b) \\ (D2) & \Box a \wedge \Diamond b & = & \Box a \wedge \Diamond(a \wedge b) \end{array}$$

Proposition 3.4.8 (T1) $\forall a \in |\mathcal{P}(A)|. \exists b_1, \ldots, b_n \in \mathsf{C}(\mathcal{P}(A)). a =_{\mathcal{P}(A)} \bigvee_{i=1}^{n} b_i$.

Proof. In order to keep the notation bearable, we shall omit subscripts, writing e.g. $\bigvee\{a\}$.

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

$$\bigvee\{\bigwedge\{\Box a\} \wedge \bigwedge\{\Diamond b\}\}$$

Applying $(\Box - \wedge)$ and $(p3)$ to each sub-expression $\bigwedge\{\Box a\}$, we obtain:

$$\bigvee\{\Box\{\bigvee c\} \wedge \bigwedge\{\Diamond b\}\}$$

where each c is coprime; applying $(p3)$ to each b, then $(\Diamond - \vee)$ and the distributive laws, we obtain:

$$\bigvee\{\Box \bigvee\{c\} \wedge \bigwedge\{\Diamond d\}\} \tag{3.5}$$

where each c and d is coprime. We now aim to transform (3.5) into an equivalent expression of the same form satisfying, for each disjunct

$$\Box \bigvee\{c\} \wedge \bigwedge\{\Diamond d\} \tag{3.6}$$

1. For each c, for some d, $d \leq_A c$.
2. For each d, for some c, $d \leq_A c$.

(Note the resemblance to the Egli-Milner ordering). Our strategy is to use the derived equations $(D1)$, $(D2)$ to remove failures of (1) and (2) respectively.

Firstly, we show that (3.5) can be transformed into an equivalent expression satisfying (1), by induction on (n, m), where:

- n is the maximum number of c occurring in some disjunct (3.6) such that there is no d with $d \leq_A c$
- m is the number of disjuncts attaining this maximum.

If $n = 0$, (3.5) satisfies (1). Otherwise, choose such a c in one of the maximal disjuncts (3.6). We can apply $(D1)$ to $\Box(\bigvee\{c'\} \vee c)$ to obtain

$$\Box \bigvee\{c'\} \vee [\Box(\bigvee\{c'\} \vee c) \wedge \Diamond c] \tag{3.7}$$

We can then use the distributive laws to obtain a formula of the form (3.5) to which the induction hypothesis can be applied, since the first disjunct in (3.7) has jettisoned c, while the second evidently contains a $\Diamond d$ such that $d \leq_A c$, namely $\Diamond c$.

Now we remove failures of (2). We argue by induction in the same way as for the previous step. Suppose we are given a d in a disjunct (3.6) such that there is no c with $d \leq_A c$. Using $(D2)$, we obtain

$$\Box \bigvee\{c\} \wedge \Diamond[d \wedge \bigvee\{c\}] \wedge \bigwedge\{\Diamond d'\} \tag{3.8}$$

By the distributive law and $(p3)$, $d \wedge \bigvee\{c\} = \bigvee\{\bigvee\{e\}\}$, where each e is coprime. Applying $(\Diamond - \vee)$ and the distributive law again, we replace (3.6) by a disjunction of formulae

$$\Box \bigvee\{c\} \wedge \Diamond e \wedge \bigwedge\{\Diamond d'\}$$

to obtain a new expression θ of the form (3.5). Since $e \leq \bigvee\{e\} = d \wedge c \leq c$ for some c, we can apply the induction hypothesis to θ.

At this point, we have

$$a =_{\mathcal{P}(A)} \bigvee\{\Box \bigvee\{c\} \wedge \bigwedge\{\Diamond d\}\}$$

where each c and d is coprime, and each disjunct satisfies (1) and (2). Next we show that, under these conditions:

$$\bigvee\{\Box \bigvee\{c\} \wedge \bigwedge\{\Diamond d\}\} = \bigvee\{\Box(\bigvee\{c\} \vee \bigvee\{d\}) \wedge (\bigwedge\{\Diamond c\} \wedge \bigwedge\{\Diamond d\})\} \tag{3.9}$$

By (1) and $(\Diamond - \leq)$,

$$\bigwedge\{\Diamond d\} = \bigwedge\{\Diamond c\} \wedge \bigwedge\{\Diamond d\}.$$

By (2), $\bigvee\{c\} = \bigvee\{c\} \vee \bigvee\{d\}$; hence, by $(\Box - \leq)$,

$$\Box \bigvee\{c\} = \Box(\bigvee\{c\} \vee \bigvee\{d\}).$$

Combining these two equations yields (3.9).

We now have $a =_{\mathcal{P}(A)} \theta$, where θ is a disjunction of terms of the form

$$\Box \bigvee\{e\} \wedge \bigwedge\{\Diamond e\} \tag{3.10}$$

where each e is coprime. Such a term is coprime by $(\mathsf{C} - \Box - \Diamond)$, unless the disjunction $\bigvee\{e\}$ is empty. In this case, we can use $(\Box - 0)$ to delete (3.10) from θ; having deleted all such empty disjuncts, we finally obtain the required normal form. ∎

Proposition 3.4.9 (T2) *For all $a, b \in |\mathcal{P}(A)|$:*

- $a \leq_{\mathcal{P}(A)} b \implies [\![a]\!]_{\mathcal{P}(A)} \subseteq [\![b]\!]_{\mathcal{P}(A)}$
- $\mathsf{C}_{\mathcal{P}(A)}(a) \implies \exists b \in \mathcal{K}(\mathcal{P}(\hat{A})).\, [\![a]\!]_{\mathcal{P}(A)} = \uparrow(b)$
- $\mathsf{T}_{\mathcal{P}(A)}(a) \implies \perp_{\mathcal{P}(\hat{A})} \notin [\![a]\!]_{\mathcal{P}(A)}$.

PROOF. We give two cases for illustration. To validate $(\mathsf{C} - \Box - \Diamond)$, consider $\{a_i\}_{i \in I}$ with $\mathsf{C}_A(a_i)$, $i \in I \neq \varnothing$. Let $[\![a_i]\!]_A = \uparrow(u_i)$, $i \in I$. Then

$$
\begin{array}{ll}
\bullet & S \in [\![\Box \bigvee_{i \in I} a_i \wedge \bigwedge_{i \in I} a_i]\!]_{\mathcal{P}(A)} \\
\Leftrightarrow & S \subseteq \bigcup_{i \in I} \uparrow(u_i) \;\&\; \forall i \in I.\, S \cap \uparrow(v_i) \neq \varnothing \\
\Leftrightarrow & \forall x \in S.\, \exists i \in I.\, u_i \sqsubseteq x \;\&\; \forall i \in I.\, \exists x \in S.\, u_i \sqsubseteq x \\
\Leftrightarrow & \{u_i\}_{i \in I} \sqsubseteq_{EM} S \\
\Leftrightarrow & S \in \uparrow_{\mathcal{P}(\hat{A})}(\mathsf{Con}(\{u_i\}_{i \in I})).
\end{array}
$$

For $(\Box - \vee)$,

$$
\begin{array}{rcl}
S \in [\![\Box(a \vee b)]\!]_{\mathcal{P}(A)} & \Rightarrow & S \subseteq ([\![a]\!]_A \cup [\![b]\!]_A) \\
& \Rightarrow & S \subseteq [\![a]\!]_A \text{ or } S \cap [\![b]\!]_A \neq \varnothing \\
& \Rightarrow & S \in [\![\Box a \vee \Diamond b]\!]_{\mathcal{P}(A)}. \quad \blacksquare
\end{array}
$$

Note that $(\Box - 0)$ is valid because the empty set is excluded from $\mathcal{P}(\hat{A})$. (In fact, dropping $(\Box - 0)$ and removing the side-condition $I \neq \varnothing$ in $(\mathsf{C} - \Box - \Diamond)$ corresponds exactly to retaining the empty set. For further discussion, see [Abr87a].)

Proposition 3.4.10 (T3) *For all $a, b \in \mathsf{C}(\mathcal{P}(A))$:*

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

PROOF. By induction on the proofs of $\mathsf{C}_{\mathcal{P}(A)}(a)$, $\mathsf{C}_{\mathcal{P}(A)}(b)$. The non-trivial case is when these are both inferred by $(\mathsf{C} - \Box - \Diamond)$, with $a = \Box \bigvee_{i \in I} a_i \wedge \bigwedge_{i \in I} \Diamond a_i$, $b = \Box \bigvee_{j \in J} b_j \wedge \bigwedge_{j \in J} \Diamond b_j$. In this case, let $[\![a_i]\!]_A = \uparrow(u_i)$, $[\![b_j]\!]_A = \uparrow(v_j)$, $(i \in I, j \in J)$. Then

$$
\begin{array}{rcl}
[\![a]\!]_{\mathcal{P}(A)} \subseteq [\![b]\!]_{\mathcal{P}(A)} & \Rightarrow & \{v_j\} \sqsubseteq_{EM} \{u_i\} \\
& \Rightarrow & \forall j.\, \exists i.\, a_i \leq_A b_j \;\;\&\;\; \forall i.\, \exists j.\, a_i \leq_A b_j \\
& \Rightarrow & \bigwedge_{i \in I} \Diamond a_i \leq_{\mathcal{P}(A)} \bigwedge_{j \in J} \Diamond b_j \;\;\&\;\; \Box \bigvee_{i \in I} a_i \leq_{\mathcal{P}(A)} \Box \bigvee_{j \in J} b_j \\
& \Rightarrow & a \leq_{\mathcal{P}(A)} b. \quad \blacksquare
\end{array}
$$

Proposition 3.4.11 (T4) *For all $u \in \mathcal{K}(\mathcal{P}(\hat{A}))$, for some $a \in |\mathcal{P}(A)|$:*

$$[\![a]\!]_{\mathcal{P}(A)} = \uparrow(u).$$

PROOF. A finite element u has the form $\mathsf{Con}(\{u_i\}_{i\in I})$, with $u_i \in \mathcal{K}(\hat{A})$. Take $a_i \in |A|$ with $[\![a_i]\!]_A = \uparrow(u_i)$, $(i \in I)$. Then

$$[\![\Box \bigvee_{i\in I} a_i \wedge \bigwedge_{i\in I} \Diamond a_i]\!]_{\mathcal{P}(A)} = \uparrow(u),$$

as required. ∎

Proposition 3.4.12 (T5) *Let $A \trianglelefteq B$, with $e : \hat{A} \to \hat{B}$ the corresponding projection. For $a \in \mathsf{C}(\mathcal{P}(A))$, with $[\![a]\!]_{\mathcal{P}(A)} = \uparrow(u)$, $[\![a]\!]_{\mathcal{P}(B)} = \uparrow(v)$,*

$$\mathcal{P}(e)(u) = v.$$

PROOF. By induction on the proof that $\mathsf{C}_{\mathcal{P}(A)}(a)$. The non-trivial case is when this is inferred by $(\mathsf{C} - \Box - \Diamond)$, with $a = \Box \bigvee_{i\in I} a_i \wedge \bigwedge_{i\in I} \Diamond a_i$. Let $[\![a_i]\!]_A = \uparrow(u_i)$, $[\![a_i]\!]_B = \uparrow(v_i)$, $(i \in I)$. Then we have:

$$\begin{aligned} e(u_i) &= v_i \quad (i \in I) \\ u &= \mathsf{Con}(\{u_i\}) \\ v &= \mathsf{Con}(\{v_i\}) \end{aligned}$$

Now $\mathcal{P}(e)(u) = \mathsf{Con}(\{e(u_i)\}) = \mathsf{Con}(\{v_i\}) = v$. ∎

Definition 3.4.13 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) Axioms:

$$\begin{array}{lll} (\oplus - \wedge) & \bigwedge_{i\in I}(a_i \oplus f) = (\bigwedge_{i\in I} a_i \oplus f) & \bigwedge_{i\in I}(f \oplus a_i) = (f \oplus \bigwedge_{i\in I} a_i) \\ (\oplus - \vee) & \bigvee_{i\in I}(a_i \oplus f) = (\bigvee_{i\in I} a_i \oplus f) & \bigvee_{i\in I}(f \oplus a_i) = (f \oplus \bigvee_{i\in I} a_i) \end{array}$$

Rules:

$$(\mathsf{C} - \oplus) \quad \frac{\mathsf{C}_A(a)}{\mathsf{C}(a \oplus f)} \qquad \frac{\mathsf{C}_B(b)}{\mathsf{C}(f \oplus b)}$$

$$(\mathsf{T} - \oplus) \quad \frac{\mathsf{T}_A(a)}{\mathsf{T}(a \oplus f)} \qquad \frac{\mathsf{T}_B(b)}{\mathsf{T}(f \oplus b)}$$

$$(\oplus - \#) \quad \frac{\mathsf{T}_A(a) \quad \mathsf{T}_B(b)}{(a \oplus f) \wedge (f \oplus b) = 0}$$

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

(iii) Semantic function:

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

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

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

We shall only prove (T1) for coalesced sum; the verification of the remaining properties should by now be a rather straightforward exercise for the reader.

Proposition 3.4.14 (T1) $\forall a \in |A \oplus B|. \exists b_1, \ldots, b_n \in \mathsf{C}(A \oplus B). a =_{A \oplus B} \bigvee_{i=1}^{n} b_i$.

Proof. We use the same notation for meets and joins as in the proof of Proposition 3.4.8.

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

$$\bigvee\{\bigwedge\{(a' \oplus f)\} \wedge \bigwedge\{(f \oplus b')\}\}$$

Applying $(\oplus - \wedge)$, we get an expression of the form

$$\bigvee\{(a'' \oplus f) \wedge (f \oplus a'')\}$$

Applying $(p3)$ to each a'' and b'', and then $(\oplus - \vee)$ and the distributive laws, we get an expression of the form

$$\bigvee\{(c \oplus f) \wedge (f \oplus d)\} \tag{3.11}$$

where each c and d is coprime. For each of the disjuncts $e = (c \oplus f) \wedge (f \oplus d)$ of (3.11):

- If $\mathsf{T}_A(c)$ and $\mathsf{T}_B(d)$, then by $(\oplus - \#)$ we can delete e.
- If $c =_A 1$, then by $(\oplus - \wedge - L)$ we can replace e by $(f \oplus d)$.
- If $d =_B 1$, then by $(\oplus - \wedge - R)$ we can replace e by $(c \oplus f)$.

The resulting expression has the form

$$\bigvee\{(c' \oplus f)\} \vee \bigvee\{(f \oplus d')\}$$

where each c' and d' is coprime, as required. ∎

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)–(T5) (and hence also (T6)–(T9)) from the previous section with respect to a functor $\mathrm{op}^{\mathcal{D}} : \mathbf{SFP}^n \to \mathbf{SFP}$.

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:

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

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

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

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

Denotational Semantics of Type Expressions

Similarly to the logical semantics, we define

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

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

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

and $\mathsf{rec}\, t.\, \sigma$ as the initial fixed point of the endofunctor $\mathbf{SFP}^E \longrightarrow \mathbf{SFP}^E$ induced from $t \mapsto \sigma(t)$. Here $\mathbf{SFP}^E$ is the category of SFP domains and embeddings. 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.*

$$\begin{array}{ll}(i) & \mathcal{D}[\![\sigma]\!]\rho_D \ \cong\ \mathsf{Spec}(\mathcal{LA}[\![\sigma]\!]\rho_L)\\ (ii) & K\Omega(\mathcal{D}[\![\sigma]\!]\rho_D) \ \cong\ \mathcal{LA}[\![\sigma]\!]\rho_L.\end{array}$$

PROOF. Firstly, note that (i) and (ii) are equivalent, since SFP domains are coherent spaces. Thus it suffices to prove (i).

It will be convenient to consider systems of simultaneous domain equations

$$\left.\begin{array}{rcl}\xi_1 & = & \sigma_1(\xi_1,\ldots,\xi_n)\\ & \vdots & \\ \xi_n & = & \sigma_n(\xi_1,\ldots,\xi_n)\end{array}\right\} \tag{3.12}$$

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 (3.12) in **DPL1** and D is the solution in **SFP**. To make this more precise, we need some definitions.

Firstly, we define a diagram Δ^D in $(\mathbf{SFP}^E)^n$ 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 $(\mathbf{SFP}^E)^n$;

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

where $\mathcal{D}_m$ gives the morphism part of the functor corresponding to σ, and $\rho_m^D t = \mathsf{id}_{\rho^D t}$. Now it is standard that the solution of (3.12) in **SFP** is given by

$$\varinjlim \Delta^D.$$

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 $(\mathbf{SFP}^E)^n$, 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 (3.12) 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 $(\mathbf{SFP}^E)^n$. Thus our task reduces to proving

$$\varinjlim \Delta^L \cong \varinjlim \Delta^D,$$

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 $(\mathbf{SFP}^E)^n$;

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

$$\begin{aligned}
D_{\vec{\tau},0} &= (\mathbf{1}^{\mathcal{D}}, \ldots, \mathbf{1}^{\mathcal{D}}); \quad A_{\vec{\tau},0} = (\mathbf{1}^{\mathcal{L}}, \ldots, \mathbf{1}^{\mathcal{L}}) \\
D_{\vec{\tau},k+1} &= (\mathcal{D}[\![\tau_1]\!]\rho^D[\vec{\xi} \mapsto D_{\vec{\sigma},k}], \ldots, \mathcal{D}[\![\tau_n]\!]\rho^D[\vec{\xi} \mapsto D_{\vec{\sigma},k}]) \\
A_{\vec{\tau},k+1} &= (\mathcal{L}[\![\tau_1]\!]\rho^L[\vec{\xi} \mapsto A_{\vec{\sigma},k}], \ldots, \mathcal{L}[\![\tau_n]\!]\rho^L[\vec{\xi} \mapsto A_{\vec{\sigma},k}])
\end{aligned}$$

$f_{\vec{\tau},0}$ 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:

$$\begin{aligned}
\nu_{\xi_i,k+1} &= \nu_{\sigma_i,k} \\
\nu_{t,k+1} &= \hat{\rho}^L t \cong \rho^D t,
\end{aligned}$$

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 (T9) 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 $(\mathbf{SFP}^E)^n$. For the inductive step, we assume:

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

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

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

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

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

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

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

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

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

$$f_{\tau,k+1} = \mathsf{id}_{D_{\tau,k+1}}, \quad e_{\tau,k+1} = \mathsf{id}_{\hat{A}_{\tau,k+1}}, \quad \nu_{\tau,k+1} = \nu_{\tau,k+2} = (\hat{\rho}^L t \cong \rho^D t),$$

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

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

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

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 (T9). This proves (iii). Finally,

$$\begin{aligned}
\nu_{\tau,k+2}\circ e_{\tau,k+1} &= \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2},\ldots,\nu_{\theta_m,k+2})\circ\eta_{\tau,k+2}\circ e_{\tau,k+1} \\
&= \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2},\ldots,\nu_{\theta_m,k+2})\circ\mathrm{op}^{\mathcal{D}}(e_{\theta_1,k+1},\ldots,e_{\theta_m,k+1})\circ\eta_{\tau,k+1} && \text{by (T9)} \\
&= \mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+2}\circ e_{\theta_1,k+1},\ldots,\nu_{\theta_m,k+2}\circ e_{\theta_m,k+1})\circ\eta_{\tau,k+1} \\
&= \mathrm{op}^{\mathcal{D}}(f_{\theta_1,k+2}\circ\nu_{\theta_1,k+1},\ldots,f_{\theta_m,k+2}\circ\nu_{\theta_m,k+1})\circ\eta_{\tau,k+1} && \text{by } (viii) \\
&= \mathrm{op}^{\mathcal{D}}(f_{\theta_1,k+2},\ldots,f_{\theta_m,k+2})\circ\mathrm{op}^{\mathcal{D}}(\nu_{\theta_1,k+1},\ldots,\nu_{\theta_m,k+1})\circ\eta_{\tau,k+1} \\
&= f_{\tau,k+2}\circ\nu_{\tau,k+1},
\end{aligned}$$

which proves (iv). ∎

Chapter 4

Domain Logics

4.1 Introduction

In this Chapter we shall complete the core of our 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 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:

$$\begin{array}{ccc} \text{cartesian closed categories} & & \text{exogenous logic} \\ vs. & \sim & vs. \\ \lambda\text{-calculus} & & \text{endogenous logic.} \end{array}$$

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 outline the two case studies described in [Abr88,Abr87a].

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

$$\sigma \quad ::= \quad \mathbf{1} \mid \sigma \times \tau \mid \sigma \to \tau \mid \sigma \oplus \tau \mid (\sigma)_\perp \mid \mathcal{P}\sigma \mid t \mid \mathsf{rec}\, t.\, \sigma$$

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

The standard way of interpreting these expressions is as objects of **SFP** (more generally as dcpo's, but **SFP** is closed under all the above constructions as a subcategory of **DCPO**). Thus for each type expression σ we define a domain $\mathcal{D}(\sigma) = (D(\sigma), \sqsubseteq_\sigma)$ in **SFP**; $\sigma \times \tau$ is interpreted as product, $\sigma \to \tau$ as function space, $\sigma \oplus \tau$ as coalesced sum, $(\sigma)_\perp$ as lifting, $\mathcal{P}\sigma$ as the Plotkin powerdomain, 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 **SFP**. Other constructions (e.g. strict function space, smash product, the Smyth and Hoare powerdomains) can be added to the list.

So far, all this is standard ([Plo81,Gun85,Gun87,SP82]). Now we begin our alternative approach. For each type expression σ, we shall define a propositional theory (in fact, a *domain prelocale* in the sense of Chapter 3)

$$\mathcal{L}(\sigma) = (L(\sigma), \leq_\sigma, =_\sigma, f_\sigma, \vee_\sigma, t_\sigma, \wedge_\sigma, \mathsf{C}_\sigma, \mathsf{T}_\sigma),$$

where:

- $L(\sigma)$ is a set of *formulae*.
- $\leq_\sigma$, $=_\sigma$ are the relations of logical *entailment* and *equivalence* between formulae.
- $\vee$, $\wedge$ are the logical connectives for *disjunction* and *conjunction*.
- t, f are constants for *truth* and *falsity* (i.e. nullary conjunction and disjunction).
- C, T are predicates for *coprimeness* and *termination*, as motivated in Chapter 3.

$\mathcal{L}(\sigma)$ is defined inductively via formation rules, axioms and inference rules in the usual way.[1]

Formation Rules

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

[1] In fact, for each closed type expression σ and type environment ρ, it will be the case that $\mathcal{L}(\sigma) = \mathcal{L}[\![\sigma]\!]\rho$, where $\mathcal{L}[\![\sigma]\!]\rho$ is the logical semantics defined in Section 3.5. (This justifies our overloading of the the symbol "$\mathcal{L}$"). However, we give a direct definition of $\mathcal{L}(\sigma)$ here, as an explicit syntactic presentation may be helpful to the reader.

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

We should think of $(\phi \to \psi)$, $\Box\phi$ 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
$\mathsf{List(N)}$	$\mathsf{rec}\, t.\, \mathbf{1} + (\mathsf{N} \times t)$	lazy lists of eager numbers
CBN	$\mathsf{rec}\, t.\, \mathsf{N} + (t \to t)$	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	$(f \oplus (\overline{0} \times \mathsf{nil})_\perp)$	List(N)
$\overline{0} :: \perp$	$(f \oplus (\overline{0} \times t)_\perp)$	List(N)
parallel or	$((\mathsf{true} \times t) \to \mathsf{true})$	
	$\wedge\, ((t \times \mathsf{true}) \to \mathsf{true})$	
	$\wedge\, ((\mathsf{false} \times \mathsf{false}) \to \mathsf{false})$	$(\mathsf{B} \times \mathsf{B}) \to \mathsf{B}$

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 a number of 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.

$$
\begin{array}{ll}
(\times - \wedge) & \bigwedge_{i\in I}(\phi_i \times \psi_i) = (\bigwedge_{i\in I}\phi_i \times \bigwedge_{i\in I}\psi_i) \\
(\times - \vee - L) & (\bigvee_{i\in I}\phi_i \times \psi) = \bigvee_{i\in I}(\phi \times \psi) \\
(\times - \vee - R) & (\phi \times \bigvee_{i\in I}\psi_i) = \bigvee_{i\in I}(\phi \times \psi_i) \\
(\rightarrow - \wedge) & (\phi \rightarrow \bigwedge_{i\in I}\psi_i) = \bigwedge_{i\in I}(\phi \rightarrow \psi_i) \\
(\rightarrow - \vee - L) & (\bigvee_{i\in I}\phi_i \rightarrow \psi) = \bigwedge_{i\in I}(\phi_i \rightarrow \psi) \\
(\oplus - \wedge - L) & (\bigwedge_{i\in I}\phi_i \oplus f) = \bigwedge_{i\in I}(\phi_i \oplus f) \\
(\oplus - \wedge - R) & (f \oplus \bigwedge_{i\in I}\psi_i) = \bigwedge_{i\in I}(f \oplus \psi_i) \\
(\oplus - \vee - L) & (\bigvee_{i\in I}\phi_i \oplus f) = \bigvee_{i\in I}(\phi_i \oplus f) \\
(\oplus - \vee - R) & (f \oplus \bigvee_{i\in I}\psi_i) = \bigvee_{i\in I}(f \oplus \psi_i) \\
((\cdot)_\perp - \wedge) & (\phi \wedge \psi)_\perp = (\phi)_\perp \wedge (\psi)_\perp \\
((\cdot)_\perp - \vee) & (\bigvee_{i\in I}\phi_i)_\perp = \bigvee_{i\in I}(\phi_i)_\perp \\
(\Box - \wedge) & \Box \bigwedge_{i\in I}\phi_i = \bigwedge_{i\in I}\Box\phi_i \\
(\Diamond - \vee) & \Diamond \bigvee_{i\in I}\phi_i = \bigvee_{i\in I}\Diamond\phi_i \\
(\Box - \vee) & \Box(\phi \vee \psi) \leq \Box\phi \vee \Diamond\psi \\
(\Diamond - \wedge) & \Box\phi \wedge \Diamond\psi \leq \Diamond(\phi \wedge \psi) \\
(\Box - f) & \Box f = f
\end{array}
$$

Rules

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

$$(\oplus - \#) \quad \frac{\mathsf{T}(\phi) \quad \mathsf{T}(\psi)}{(\phi \oplus f) \wedge (f \oplus \psi) = f}$$

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

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

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

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

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

Generic Axioms for Coprimeness and Termination

$$(\mathsf{C}-=)\quad \frac{\mathsf{C}(\phi),\quad \phi=\psi}{\mathsf{C}(\psi)} \qquad (\mathsf{C}-t)\quad \mathsf{C}(t)$$

$$(\mathsf{T}-\leq)\quad \frac{\mathsf{T}(\phi),\quad \psi\leq\phi}{\mathsf{T}(\psi)} \qquad (\mathsf{T}-\wedge)\quad \frac{\mathsf{T}(\phi_j)\ \ (j\in I)}{\mathsf{T}(\bigwedge_{i\in I}\phi_i)} \qquad (\mathsf{T}-\vee)\quad \frac{\{\mathsf{T}(\phi_i)\}_{i\in I}}{\mathsf{T}(\bigvee_{i\in I}\phi_i)}$$

Type-Specific Axioms for Coprimeness

$$(\mathsf{C}-\times)\quad \frac{\mathsf{C}(\phi)\ \ \mathsf{C}(\psi)}{\mathsf{C}(\phi\times\psi)}$$

$$(\mathsf{C}-\rightarrow)\quad \frac{\{\mathsf{C}(\phi_i)\}_{i\in I}\quad \{\mathsf{C}(\psi_i)\}_{i\in I} \qquad \forall J\subseteq I.\,\exists K\subseteq I.\,[\bigwedge_{j\in J}\phi_j=\bigvee_{k\in K}\phi_k\ \&\ [\forall j\in J,k\in K.\,\psi_k\leq\psi_j]]}{\mathsf{C}(\bigwedge_{i\in I}(\phi_i\rightarrow\psi_i))}$$

$$(\mathsf{C}-\oplus)\quad \frac{\mathsf{C}(\phi)}{\mathsf{C}(\phi\oplus f)} \qquad \frac{\mathsf{C}(\psi)}{\mathsf{C}(f\oplus\psi)} \qquad (\mathsf{C}-(\cdot)_\perp)\quad \frac{\mathsf{C}(\phi)}{\mathsf{C}((\phi)_\perp)}$$

$$(\mathsf{C}-\Box-\Diamond)\quad \frac{\{\mathsf{C}(\phi_i)\}_{i\in I}\ \ (I\neq\varnothing)}{\mathsf{C}(\Box\bigvee_{i\in I}\phi_i\wedge\bigwedge_{i\in I}\Diamond\phi_i)}$$

Type-Specific Axioms for Termination

$$(\mathsf{T}-\times)\quad \frac{\mathsf{T}(\phi)}{\mathsf{T}(\phi\times\psi)} \qquad \frac{\mathsf{T}(\psi)}{\mathsf{T}(\phi\times\psi)}$$

$$(\mathsf{T}-\rightarrow)\quad \frac{\mathsf{C}(\phi')\ \ \phi'\leq\phi\ \ \mathsf{T}(\psi)}{\mathsf{T}(\phi\rightarrow\psi)}$$

$$(\mathsf{T}-\oplus)\quad \frac{\mathsf{T}(\phi)}{\mathsf{T}(\phi\oplus f)} \qquad \frac{\mathsf{T}(\psi)}{\mathsf{T}(f\oplus\psi)}$$

$$(\mathsf{T}-(\cdot)_\perp)\quad \mathsf{T}((\phi)_\perp)$$

$$(\mathsf{T}-\Box)\quad \frac{\mathsf{T}(\phi)}{\mathsf{T}(\Box\phi)} \qquad (\mathsf{T}-\Diamond)\quad \frac{\mathsf{T}(\phi)}{\mathsf{T}(\Diamond\phi)}$$

Remarks on the Axiomatization

The above axioms and rules are to be understood as *schemes*, which are instantiated by uniform substitution for meta-variables. A *literal* is one of the basic assertions $\phi\leq\psi$, $\phi=\psi$, $\mathsf{C}(\phi)$, $\mathsf{T}(\phi)$. All our rules have the form: "from a finite conjunction of literals (empty in the case of an axiom) infer a literal", i.e.

$$\bigwedge_{i\in I} l_i \ \Rightarrow\ l$$

with the single exception of $(\mathsf{C} - \rightarrow)$. However, the quantifications in each instance of $(\mathsf{C} - \rightarrow)$ can be expanded into finite conjunctions and disjunctions; using the distributive laws, each such instance can be put in the form

$$\bigvee_{i \in I} \bigwedge_{j \in J_i} l_{ij} \;\Rightarrow\; l$$

which is equivalent to

$$\bigwedge_{i \in I} [\bigwedge_{j \in J_i} l_{ij} \;\Rightarrow\; l]$$

and hence to a finite set of rules of the standard form. Thus our axiomatization generates a monotone inductive definition in the standard way [Acz77]. Moreover, it is presented by a finite set of schemes, the set of instances of each of which is readily seen to be recursive; thus the set of theorems of the logic is recursively enumerable. In fact, we shall show at the end of this section that the logic is *decidable*.

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. Similar fine-tuning yields strict function space and smash product as variations on the standard function space and product presented above; while the Smyth powerdomain is obtained from the presentation of the Plotkin powerdomain by omitting all rules which refer to $\Diamond$; and the Hoare powerdomain is obtained by omitting all rules which refer to $\Box$.

Semantics

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

$$\begin{array}{lcl}
[\![\phi \wedge \psi]\!]_\sigma & = & [\![\phi]\!]_\sigma \cap [\![\psi]\!]_\sigma \\
[\![t]\!]_\sigma & = & D(\sigma) = 1_{K\Omega(\mathcal{D}(\sigma))} \\
[\![\phi \vee \psi]\!]_\sigma & = & [\![\phi]\!]_\sigma \cup [\![\psi]\!]_\sigma \\
[\![f]\!]_\sigma & = & \varnothing = 0_{K\Omega(\mathcal{D}(\sigma))} \\
[\![(\phi \times \psi)]\!]_{\sigma\times\tau} & = & \{\langle u, v\rangle : u \in [\![\phi]\!]_\sigma,\ v \in [\![\psi]\!]_\tau\} \\
[\![(\phi \rightarrow \psi)]\!]_{\sigma\rightarrow\tau} & = & \{f \in D(\sigma \rightarrow \tau) : f([\![\phi]\!]_\sigma) \subseteq [\![\psi]\!]_\tau\} \\
[\![(\phi \oplus f)]\!]_{\sigma\oplus\tau} & = & \{\langle 0, u\rangle : u \in [\![\phi]\!]_\sigma \setminus \{\perp_\sigma\}\} \\
 & & \cup\, \{d \in D(\sigma \oplus \tau) : \perp_\sigma \in [\![\phi]\!]_\sigma\} \\
[\![(f \oplus \psi)]\!]_{\sigma\oplus\tau} & = & \{\langle 1, v\rangle : v \in [\![\psi]\!]_\tau \setminus \{\perp_\tau\}\} \\
 & & \cup\, \{d \in D(\sigma \oplus \tau) : \perp_\tau \in [\![\psi]\!]_\tau\} \\
[\![(\phi)_\perp]\!]_{(\sigma)_\perp} & = & \{\langle 0, u\rangle : u \in [\![\phi]\!]_\sigma\} \\
[\![\Box\phi]\!]_{\mathcal{P}\sigma} & = & \{S \in D(\mathcal{P}\sigma) : S \subseteq [\![\phi]\!]_\sigma\} \\
[\![\Diamond\phi]\!]_{\mathcal{P}\sigma} & = & \{S \in D(\mathcal{P}\sigma) : S \cap [\![\phi]\!]_\sigma \neq \varnothing\} \\
[\![\phi]\!]_{\mathsf{rec}\, t.\, \sigma} & = & \{\alpha_\sigma(u) : u \in [\![\phi]\!]_{\sigma[\mathsf{rec}\, t.\, \sigma/t]}\}
\end{array}$$

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 domain prelocales in the style of Chapter 3 can be distilled from our definitions for product and lifting. The reader will find no difficulty in carrying out the same programme for these constructions as that shown for function space, Plotkin powerdomain and coalesced sum in Chapter 3. It is immediate from the definitions that, for each closed σ and any $\rho \in \mathsf{LEnv}$:

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

where $\mathcal{L}[\![\sigma]\!]\rho$ is the logical semantics from Section 3.5. The following results are then immediate consequences of our work in Chapter 3.
Notation. $\mathsf{C}(\sigma) \equiv \{\phi \in L(\sigma) : \mathcal{L}(\sigma) \vdash \mathsf{C}(\phi)\}$.

Lemma 4.2.1 (Normal Forms) *For all $\phi \in L(\sigma)$, for some $\psi_i, \ldots, \psi_n \in \mathsf{C}(\sigma)$:*

$$\mathcal{L}(\sigma) \vdash \phi = \bigvee_{i=1}^{n} \psi_i.$$

Now we define a relation $\leftrightsquigarrow \subseteq \mathsf{C}(\sigma) \times K(\mathcal{D}(\sigma))$:

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

Proposition 4.2.2 $\leftrightsquigarrow$ *is a surjective total function.*

Now we come to the main results of the section:

Theorem 4.2.3 (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 4.2.4 (Stone Duality) *$\mathcal{LA}(\sigma)$ is the Stone dual of $\mathcal{D}(\sigma)$, i.e.*

$$(i)\quad \mathcal{D}(\sigma) \;\cong\; \mathsf{Spec}\,\mathcal{LA}(\sigma)$$
$$(ii)\quad K\Omega(\mathcal{D}(\sigma)) \;\cong\; \mathcal{LA}(\sigma).$$

Decidability of $\mathcal{L}$

We define a set $\mathcal{T}$ of formulae in $\mathcal{L}$ by the following syntax:

$$\phi \;::=\; t \mid (\phi \times \psi) \mid \bigwedge_{i\in I}(\phi_i \to \psi_i) \mid (\phi \oplus f) \mid (f \oplus \psi) \mid (\phi)_\perp \mid \Box \bigvee_{i\in I} \phi_i \wedge \bigwedge_{i\in I} \Diamond\phi_i .$$

A formula ϕ is a *rigid coprime* if $\phi \in \mathcal{T}$ and $\mathcal{L} \vdash \mathsf{C}(\phi)$; a formula is in *rigid coprime normal form* if it is a disjunction of rigid coprimes.

We now define predicates LEQ, TERM over rigid coprimes (of the same type in the case of LEQ), by the following inductive definitions:

- $\mathrm{LEQ}(\phi, t) \iff \mathsf{true}$
- $\mathrm{LEQ}(t, \phi) \iff \neg(\mathrm{TERM}(\phi))$
- $\mathrm{LEQ}((\phi \times \psi), (\phi' \times \psi')) \iff \mathrm{LEQ}(\phi, \phi') \;\&\; \mathrm{LEQ}(\psi, \psi')$
- $\mathrm{LEQ}(\bigwedge_{i\in I}(\phi_i \to \psi_i), \bigwedge_{j\in J}(\phi_j \to \psi_j)) \iff \forall j \in J.\, \exists i \in I.$ $\mathrm{LEQ}(\phi_j, \phi_i) \;\&\; \mathrm{LEQ}(\psi_i, \psi_j)$
- $\mathrm{LEQ}((\phi \oplus f), (\phi' \oplus f)) \iff \mathrm{LEQ}(\phi, \phi')$
- $\mathrm{LEQ}((\phi \oplus f), (f \oplus \psi)) \iff \mathrm{LEQ}(t, \psi)$
- $\mathrm{LEQ}((f \oplus \psi), (f \oplus \psi')) \iff \mathrm{LEQ}(\psi, \psi')$
- $\mathrm{LEQ}((f \oplus \psi), (\phi \oplus f)) \iff \mathrm{LEQ}(t, \phi)$
- $\mathrm{LEQ}((\phi)_\perp, (\psi)_\perp) \iff \mathrm{LEQ}(\phi, \psi)$
- $\mathrm{LEQ}(\Box \bigvee_{i\in I} \phi_i \wedge \bigwedge_{i\in I} \Diamond\phi_i,$ $\Box \bigvee_{j\in J} \psi_j \wedge \bigwedge_{j\in J} \Diamond\psi_j) \iff \forall i \in I.\, \exists j \in J.\, \mathrm{LEQ}(\phi_i, \psi_j)$ $\&\; \forall j \in J.\, \exists i \in I.\, \mathrm{LEQ}(\phi_i, \psi_j)$

- $\mathrm{TERM}(t) \iff \mathsf{false}$
- $\mathrm{TERM}((\phi \times \psi)) \iff \mathrm{TERM}(\phi)$ or $\mathrm{TERM}(\psi)$
- $\mathrm{TERM}(\bigwedge_{i\in I}(\phi_i \to \psi_i)) \iff \exists i \in I.\, \mathrm{TERM}(\psi_i)$
- $\mathrm{TERM}((\phi \oplus f)) \iff \mathrm{TERM}(\phi)$
- $\mathrm{TERM}((f \oplus \psi)) \iff \mathrm{TERM}(\psi)$
- $\mathrm{TERM}((\phi)_\perp) \iff \mathsf{true}$
- $\mathrm{TERM}(\Box \bigvee_{i\in I} \phi_i \wedge \bigwedge_{i\in I} \Diamond\phi_i) \iff \exists i \in I.\, \mathrm{TERM}(\phi_i)$

Clearly, LEQ and TERM are recursive. Moreover, we have

Proposition 4.2.5 *For all rigid coprimes ϕ, ψ:*

$$\begin{aligned} \mathit{LEQ}(\phi, \psi) &\iff \mathcal{L} \vdash \phi \leq \psi \\ \mathit{TERM}(\phi) &\iff \mathcal{L} \vdash \mathsf{T}(\phi) \end{aligned}$$

PROOF. Straightforward in the light of our work in Chapter 3, particularly the proofs of the Coprime Completeness property (T3). ∎

We now give a sharpened form of Proposition 4.2.1.

Proposition 4.2.6 *There is an algorithm which for any type σ and formula ϕ of $\mathcal{L}(\sigma)$ produces a formula ψ in rigid coprime normal form such that*

$$\mathcal{L} \vdash \phi = \psi.$$

The proposition is proved by induction on the complexity of σ; recursive types are handled by observing that if $\phi \in L(\mathsf{rec}\, t.\, \sigma)$, ϕ is in some non-recursive finite unfolding $\sigma^{(k)}(\mathbf{1})$. We omit the details, which can be extracted from our proofs of the normal form property (T1) for the various type constructions in Chapter 3. However, note that Proposition 4.2.5 is needed, since the entailment relation is used in the normal form algorithms for function spaces (Proposition 3.4.2) and powerdomains (Proposition 3.4.8); while T is used in the normal form algorithm for coalesced sum (Proposition 3.4.14).

Given rigid coprime normal forms $\bigvee_{i\in I} \phi_i$, $\bigvee_{j\in J} \psi_j$, it is clear from our work in Chapter 3, together with Theorem 4.2.3, that:

- $\mathcal{L} \vdash \bigvee_{i\in I} \phi_i \leq \bigvee_{j\in J} \psi_j \iff \forall i \in I.\, \exists j \in J.\, \mathcal{L} \vdash \phi_i \leq \psi_j$
- $\mathcal{L} \vdash \bigvee_{i\in I} \phi_i = \bigvee_{j\in J} \psi_j \iff \mathcal{L} \vdash \bigvee_{i\in I} \phi_i \leq \bigvee_{j\in J} \psi_j \;\&\; \mathcal{L} \vdash \bigvee_{j\in J} \psi_j \leq \bigvee_{i\in I} \phi_i$
- $\mathcal{L} \vdash \mathsf{C}(\bigvee_{i\in I} \phi_i) \iff \exists i_0 \in I.\, \forall i \in I.\, \mathcal{L} \vdash \phi_i \leq \phi_{i_0}$
- $\mathcal{L} \vdash \mathsf{T}(\bigvee_{i\in I} \phi_i) \iff \forall i \in I.\, \mathcal{L} \vdash \mathsf{T}(\phi_i)$

Thus, to show that $\leq$, $=$, C, T are recursive, it suffices by Proposition 4.2.6 to show that $\leq$, T are recursive over rigid coprimes; i.e. Proposition 4.2.5. Combining these results, we have

Theorem 4.2.7 *$\mathcal{L}$ is decidable.*

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".

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

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

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

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

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

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

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

$$(\mathcal{P} - I)\ \frac{M : \sigma}{\{\!|M|\!\} : \mathcal{P}\sigma} \qquad (\mathcal{P} - E)\ \frac{M : \mathcal{P}\sigma, \quad N : \mathcal{P}\tau}{\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x^\sigma|\!\}.\, N : \mathcal{P}\tau}$$

$$(\mathcal{P} - +)\ \frac{M, N : \mathcal{P}\sigma}{M \uplus N : \mathcal{P}\sigma} \qquad (\mathcal{P} - \otimes)\ \frac{M : \mathcal{P}\sigma, \quad N : \mathcal{P}\tau}{M \otimes N : \mathcal{P}(\sigma \times \tau)}$$

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

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

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 [ML83] 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 $\Gamma \in \prod_\sigma \{\mathsf{Var}(\sigma) \rightarrow L(\sigma)\}$ 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][2] we can get away with it", in the delightful formulation of Barr and Wells [BW84, p. 1].

Axiomatisation

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

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

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

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

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

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

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

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

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

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

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

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

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

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. (In the case of the powerdomain constructions, we have one rule each for box and diamond, and refer e.g. to $(\vdash - \Box - I)$). Note the role of the termination predicate T 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.

[2] Inserted by the author.

Also, note the resemblance of our system to *type inference* (particularly to the conjunctive type discipline of [BCDC83,CDCHL84]); this stands out even more clearly if we use the notation

$$\Gamma \vdash M : \phi$$

for assertions. One can profitably think of properties ϕ as "local types", in a richer type system (powerdomains, lifting, recursive types) than is usually considered.

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*.

$$\begin{array}{lcl}
[\![x]\!]\rho & = & \rho x \\
[\![(M,N)]\!]\rho & = & \langle [\![M]\!]\rho, [\![N]\!]\rho \rangle \\
[\![\mathsf{let}\ M\ \mathsf{be}\ (x,y).\, N]\!]\rho & = & [\![N]\!]\rho[x \mapsto d, y \mapsto e] \\
 & & \text{where} \\
 & & \langle d, e \rangle = [\![M]\!]\rho \\
[\![\lambda x.M]\!]\rho & = & (d \mapsto [\![M]\!]\rho[x \mapsto d]) \\
[\![MN]\!]\rho & = & ([\![M]\!]\rho)([\![N]\!]\rho) \\
[\![\imath(M)]\!]\rho & = & \begin{cases} \langle 0, [\![M]\!]\rho \rangle, & [\![M]\!]\rho \neq \bot \\ \bot & [\![M]\!]\rho = \bot \end{cases} \\
[\![\jmath(N)]\!]\rho & = & \begin{cases} \langle 1, [\![N]\!]\rho \rangle, & [\![N]\!]\rho \neq \bot \\ \bot & [\![N]\!]\rho = \bot \end{cases} \\
[\![\mathsf{cases}\ M\ \mathsf{of} & & \\
\imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2]\!]\rho & = & \begin{cases} [\![N_1]\!]\rho[x \mapsto d], & [\![M]\!]\rho = \langle 0, d \rangle \\ [\![N_2]\!]\rho[y \mapsto e], & [\![M]\!]\rho = \langle 1, e \rangle \\ \bot, & [\![M]\!]\rho = \bot \end{cases} \\
[\![\mathsf{up}(M)]\!]\rho & = & \langle 0, [\![M]\!]\rho \rangle \\
[\![\mathsf{lift}\ M\ \mathsf{to}\ \mathsf{up}(x).\, N]\!]\rho & = & \begin{cases} [\![N]\!]\rho[x \mapsto d], & [\![M]\!]\rho = \langle 0, d \rangle \\ \bot, & [\![M]\!]\rho = \bot \end{cases}
\end{array}$$

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

Here α is the initial algebra isomorphism as in Section 2 page 55.

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: for closed σ and M it says that, for any ρ and Γ:

$$\eta_\sigma^{-1}(\{[\phi]_{=_\sigma} : M,\Gamma\vdash\phi\}) = [\![M]\!]_\sigma\rho,$$

where

$$\eta_\sigma : \mathcal{D}(\sigma) \ \cong\ \mathsf{Spec}(\mathcal{LA}(\sigma))$$

is the component of the natural isomorphism arising from Theorem 4.2.4; 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* Coprime 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 $(\vdash - \to - I)$:

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

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

$$\lambda x.M, \Gamma \models (\phi \to \psi).$$

2. Next we consider $(\vdash - \Box - E)$:

$$\frac{M, \Gamma \vdash \Box\phi \quad N, \Gamma[x \mapsto \phi] \vdash \Box\psi}{\textsf{over}\ M\ \textsf{extend}\ \{\!|x|\!\}.\,N, \Gamma \vdash \Box\psi}$$

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

$$\begin{array}{ll} \bullet & X = \bigcup_{d \in [\![M]\!]\rho} [\![N]\!]\rho[x \mapsto d] \subseteq [\![\psi]\!] \\ \implies & [\![\textsf{over}\ M\ \textsf{extend}\ \{\!|x|\!\}.\,N]\!]\rho \subseteq \uparrow(X) \subseteq [\![\psi]\!] \\ \implies & \textsf{over}\ M\ \textsf{extend}\ \{\!|x|\!\}.\,N, \Gamma \models \Box\psi. \quad \blacksquare \end{array}$$

Next, we shall need a technical lemma which describes our program constructs under the denotational semantics.

Lemma 4.3.3 *For $u \in \mathcal{K}(\mathcal{D}(\sigma))$, $v \in \mathcal{K}(\mathcal{D}(\tau))$, $w \in \mathcal{K}(\mathcal{D}(\upsilon))$, $X \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\sigma)))$, $Y \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\tau)))$, $Z \in \wp_{\mathsf{fne}}(\mathcal{K}(\mathcal{D}(\sigma \times \tau)))$, $w_1 \in \mathcal{K}(\mathcal{D}(\mathsf{rec}\,t.\,\sigma))$, $w_2 \in \mathcal{K}(\mathcal{D}(\sigma[\mathsf{rec}\,t.\,\sigma/t]))$:*

$$\begin{array}{rl}
(i) & (u, v) \sqsubseteq [\![(M, N)]\!]\rho \iff u \sqsubseteq [\![M]\!]\rho \mathbin{\&} v \sqsubseteq [\![N]\!]\rho \\
(ii) & w \sqsubseteq [\![\textsf{let}\ M\ \textsf{be}\ (x, y).\,N]\!]\rho \iff \exists u, v. \\
 & (u, v) \sqsubseteq [\![M]\!]\rho \mathbin{\&} w \sqsubseteq [\![N]\!]\rho[x \mapsto u, y \mapsto v] \\
(iii) & (u \searrow v) \sqsubseteq [\![\lambda x.M]\!]\rho \iff v \sqsubseteq [\![M]\!]\rho[x \mapsto u] \\
(iv) & v \sqsubseteq [\![MN]\!]\rho \iff \exists u.(u \searrow v) \sqsubseteq [\![M]\!]\rho \mathbin{\&} u \sqsubseteq [\![N]\!]\rho \\
(v) & \langle 0, u\rangle \sqsubseteq [\![\imath(M)]\!]\rho \iff u \sqsubseteq [\![M]\!]\rho \\
 & \langle 1, v\rangle \sqsubseteq [\![\jmath(N)]\!]\rho \iff v \sqsubseteq [\![N]\!]\rho \\
(vi) & w \neq \bot \implies w \sqsubseteq [\![\textsf{cases}\ M\ \textsf{of}\ \imath(x).\,N_1\ \textsf{else}\ \jmath(y).\,N_2]\!]\rho \iff \\
 & \exists u \neq \bot.\,\langle 0, u\rangle \sqsubseteq [\![M]\!]\rho \mathbin{\&} w \sqsubseteq [\![N_1]\!]\rho[x \mapsto u] \\
 & or \\
 & \exists v \neq \bot.\,\langle 1, v\rangle \sqsubseteq [\![M]\!]\rho \mathbin{\&} w \sqsubseteq [\![N_2]\!]\rho[x \mapsto v] \\
(vii) & \langle 0, u\rangle \sqsubseteq [\![\textsf{up}(M)]\!]\rho \iff u \sqsubseteq [\![M]\!]\rho \\
(viii) & v \neq \bot \implies v \sqsubseteq [\![\textsf{lift}\ M\ \textsf{to}\ \textsf{up}(x).\,N]\!]\rho \iff \\
 & \exists u.\,\langle 0, u\rangle \sqsubseteq [\![M]\!]\rho \mathbin{\&} v \sqsubseteq [\![N]\!]\rho[x \mapsto u]
\end{array}$$

$$
\begin{array}{rl}
(ix) & \mathsf{Con}(X) \sqsubseteq [\![\{\!|M|\!\}]\!]\rho \;\Leftrightarrow\; \forall x \in X.\, x \sqsubseteq [\![M]\!]\rho \\
(x) & \mathsf{Con}(Y) \sqsubseteq [\![\mathsf{over}\ M\ \mathsf{extend}\ \{\!|x|\!\}.\, N]\!]\rho \;\Leftrightarrow\; \exists X.\, \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho \\
& \;\&\; \mathsf{Con}(Y) \sqsubseteq \mathsf{Con}(\bigcup_{u \in X} [\![N]\!]\rho[x \mapsto u]) \\
(xi) & \mathsf{Con}(X) \sqsubseteq [\![M \uplus N]\!]\rho \;\Leftrightarrow\; \exists Y, Z.\, \mathsf{Con}(X) = \mathsf{Con}(Y \cup Z) \\
& \;\&\; \mathsf{Con}(Y) \sqsubseteq [\![M]\!]\rho \;\&\; \mathsf{Con}(Z) \sqsubseteq [\![N]\!]\rho \\
(xii) & \mathsf{Con}(Z) \sqsubseteq [\![M \otimes N]\!]\rho \;\Leftrightarrow\; \exists X, Y.\, \mathsf{Con}(Z) \sqsubseteq \mathsf{Con}(X) \otimes \mathsf{Con}(Y) \\
& \;\&\; \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho \;\&\; \mathsf{Con}(Y) \sqsubseteq [\![N]\!]\rho \\
(xiii) & w_1 \sqsubseteq [\![\mathsf{fold}(M)]\!]\rho \;\Leftrightarrow\; \alpha^{-1}(w_1) \sqsubseteq [\![M]\!]\rho \\
(xiv) & w_2 \sqsubseteq [\![\mathsf{unfold}(M)]\!]\rho \;\Leftrightarrow\; \alpha(w_2) \sqsubseteq [\![M]\!]\rho \\
(xv) & u \sqsubseteq [\![\mu x.M]\!]\rho \;\Leftrightarrow\; \exists k \in \omega,\, u_0, \ldots, u_k.\, u_0 = \bot \;\&\; u_k = u \\
& \;\&\; \forall i : 0 \leq i < k.\, u_{i+1} \sqsubseteq [\![M]\!]\rho[x \mapsto u_i]
\end{array}
$$

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]. See also [Gun85]. We shall just give a couple of the less familiar cases for illustration.
(xi).

$$
\begin{array}{ll}
\bullet & \mathsf{Con}(X) \sqsubseteq [\![M \uplus N]\!]\rho \\
\Longleftrightarrow & X \sqsubseteq_{EM} ([\![M]\!]\rho \cup [\![N]\!]\rho) \\
\Longleftrightarrow & \mathsf{Con}(Y) \sqsubseteq [\![M]\!]\rho \;\&\; \mathsf{Con}(Z) \sqsubseteq [\![N]\!]\rho
\end{array}
$$

where

$$
\begin{array}{rcl}
Y & = & \{y \in X : \exists x \in [\![M]\!]\rho.\, y \sqsubseteq x\} \\
Z & = & \{z \in X : \exists x \in [\![N]\!]\rho.\, y \sqsubseteq x\}.
\end{array}
$$

(xii).

$$
\begin{array}{ll}
\bullet & \mathsf{Con}(Z) \sqsubseteq [\![M \otimes N]\!]\rho \\
\Longleftrightarrow & \mathsf{Con}(Z) \sqsubseteq \bigsqcup\{\mathsf{Con}(X) \otimes \mathsf{Con}(Y) : \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho \;\&\; \mathsf{Con}(Y) \sqsubseteq [\![N]\!]\rho\} \\
& \text{since } \otimes \text{ is continuous} \\
\Longleftrightarrow & \exists X, Y.\, \mathsf{Con}(Z) \sqsubseteq \mathsf{Con}(X) \otimes \mathsf{Con}(Y) \;\&\; \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho \;\&\; \mathsf{Con}(Y) \sqsubseteq [\![N]\!]\rho \\
& \text{since } \mathsf{Con}(Z) \text{ is finite. } \blacksquare
\end{array}
$$

Now for Coprime Completeness.
Notation. $\mathsf{C}(\Gamma) \equiv \forall x \in \mathsf{Var}.\, \mathsf{C}(\Gamma x)$.

Theorem 4.3.4 (Coprime Completeness) $\mathsf{C}(\Gamma)$ *and* $\mathsf{C}(\phi)$ *imply that*

$$M, \Gamma \models \phi \;\Longrightarrow\; M, \Gamma \vdash \phi$$

PROOF. We begin by establishing some useful notation. Given Γ with $\mathsf{C}(\Gamma)$, we define an environment ρ_Γ by:

$$\forall x \in \mathsf{Var}.\, \Gamma x \leftrightsquigarrow \rho_\Gamma x.$$

This is well-defined by Proposition 4.2.2. 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 the proof that $\mathsf{C}(\phi)$. The non-trivial case is when this is inferred by $(\mathsf{C} - \rightarrow)$, with $\phi = \bigwedge_{i \in I}(\phi_i \rightarrow \psi_i)$. Let $\phi_i \leftrightsquigarrow u_i$, $\psi_i \leftrightsquigarrow v_i$, $i \in I$. Then

$$\begin{array}{lll}
\bullet & \lambda x.M, \Gamma \models \phi & \\
\Rightarrow & \forall i \in I.\, \lambda x.M, \Gamma \models (\phi_i \rightarrow \psi_i) & \\
\Rightarrow & \forall i \in I.\, (u_i \searrow v_i) \sqsubseteq [\![\lambda x.M]\!]\rho_\Gamma & (4.1) \\
\Rightarrow & \forall i \in I.\, v_i \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto u_i] & 4.3.3(\mathrm{iii}) \\
\Rightarrow & \forall i \in I.\, M, \Gamma[x \mapsto \phi_i] \models \psi_i & (4.1) \\
\Rightarrow & \forall i \in I.\, M, \Gamma[x \mapsto \phi_i] \vdash \psi_i & \text{ind. hyp.} \\
\Rightarrow & \forall i \in I.\, \lambda x.M, \Gamma \vdash (\phi_i \rightarrow \psi_i) & (\vdash - \rightarrow - I) \\
\Rightarrow & \lambda x.M, \Gamma \vdash \phi & (\vdash - \wedge).
\end{array}$$

Application.

$$\begin{array}{lll}
\bullet & MN, \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![MN]\!]\rho_\Gamma & (4.1) \\
\Rightarrow & \exists v.\, (v \searrow u) \sqsubseteq [\![M]\!]\rho \;\&\; v \sqsubseteq [\![N]\!]\rho & 4.3.3(\mathrm{iv}) \\
\Rightarrow & M, \Gamma \models (\psi \rightarrow \phi) \;\&\; N, \Gamma \models \psi & (4.1) \\
 & \text{where } \psi \leftrightsquigarrow v & \\
\Rightarrow & M, \Gamma \vdash (\psi \rightarrow \phi) \;\&\; N, \Gamma \vdash \psi & \text{ind. hyp.} \\
\Rightarrow & MN, \Gamma \vdash \phi & (\vdash - \rightarrow - E).
\end{array}$$

Case expression.

$$\begin{array}{ll}
 & \mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \models \phi \\
\Leftrightarrow & u \sqsubseteq [\![\mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2]\!]\rho_\Gamma \quad (4.1).
\end{array}$$

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

$$(i)\ \exists u_1 \neq \bot.\, \langle 0, u_1\rangle \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; u \sqsubseteq [\![N_1]\!]\rho_\Gamma[x \mapsto u_1]$$

or

$$(ii)\ \exists u_2 \neq \bot.\, \langle 1, u_2\rangle \sqsubseteq [\![M]\!]\rho_\Gamma \;\&\; u \sqsubseteq [\![N_2]\!]\rho_\Gamma[x \mapsto u_2].$$

We shall consider sub-case (i); (ii) is entirely similar. Let $\phi_1 \leftrightsquigarrow u_1$. Then

$$\begin{array}{lll}
\bullet & \langle 0, u_1 \rangle \sqsubseteq [\![M]\!]\rho_\Gamma \ \&\ u \sqsubseteq [\![N_1]\!]\rho_\Gamma[x \mapsto u_1] & \\
\Rightarrow & M, \Gamma \models (\phi_1 \oplus f) \ \&\ N_1, \Gamma[x \mapsto \phi_1] \models \phi & (4.1) \\
\Rightarrow & M, \Gamma \vdash (\phi_1 \oplus f) \ \&\ N_1, \Gamma[x \mapsto \phi_1] \vdash \phi & \text{ind. hyp.} \\
\Rightarrow & \mathsf{cases}\ M\ \mathsf{of}\ \imath(x).\, N_1\ \mathsf{else}\ \jmath(y).\, N_2, \Gamma \vdash \phi & \text{by } (\vdash - \oplus - E) \\
& \text{since } u_1 \neq \bot \text{ implies } \mathsf{T}(\phi_1). &
\end{array}$$

Tensor product. By induction on the proof that $\mathsf{C}(\phi)$. The non-trivial case is when this is inferred by $(\mathsf{C} - \Box - \Diamond)$, with

$$\phi = \Box \bigvee_{i \in I} (\phi_i \times \psi_i) \wedge \bigwedge_{i \in I} \Diamond(\phi_i \times \psi_i)$$

with $\mathsf{C}(\phi_i)$, $\mathsf{C}(\psi_i)$, $i \in I$. We define $Z = \{(u_i, v_i) : i \in I\}$, where

$$\phi_i \leftrightsquigarrow u_i, \quad \psi_i \leftrightsquigarrow v_i \quad (i \in I).$$

Now

$$\begin{array}{lll}
\bullet & M \otimes N, \Gamma \models \Box \bigvee_{i \in I}(\phi \times \psi) \wedge \bigwedge_{i \in I} \Diamond(\phi_i \times \psi_i) & \\
\Rightarrow & \mathsf{Con}(Z) \sqsubseteq [\![M \otimes N]\!]\rho_\Gamma & (4.1) \\
\Rightarrow & \exists X, Y.\, \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho_\Gamma \ \&\ \mathsf{Con}(Y) \sqsubseteq [\![N]\!]\rho_\Gamma & \\
& \&\ \mathsf{Con}(Z) \sqsubseteq \mathsf{Con}(X) \otimes \mathsf{Con}(Y) = \mathsf{Con}(X \times Y) & 4.3.3(\text{xii})
\end{array}$$

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

$$\begin{array}{lll}
\bullet & \mathsf{Con}(X) \sqsubseteq [\![M]\!]\rho_\Gamma \ \&\ \mathsf{Con}(Y) \sqsubseteq [\![N]\!]\rho_\Gamma & \\
\Rightarrow & M, \Gamma \models \Box \bigvee_{k \in K} \phi_k \wedge \bigwedge_{k \in K} \Diamond\phi_k \ \ \&\ \ N, \Gamma \models \Box \bigvee_{l \in L} \psi_l \wedge \bigwedge_{l \in L} \Diamond\psi_l & (4.1) \\
\Rightarrow & M, \Gamma \vdash \Box \bigvee_{k \in K} \phi_k \wedge \bigwedge_{k \in K} \Diamond\phi_k \ \ \&\ \ N, \Gamma \vdash \Box \bigvee_{l \in L} \psi_l \wedge \bigwedge_{l \in L} \Diamond\psi_l & \text{ind. hyp.} \\
\Rightarrow & M \otimes N, \Gamma \vdash \Box(\bigvee_{k \in K} \phi_k \times \bigvee_{l \in L} \psi) \wedge \bigwedge_{(k,l) \in K \times L} \Diamond(\phi_k \times \psi_l) & (\vdash - \otimes).
\end{array}$$

Now $\mathsf{Con}(Z) \sqsubseteq \mathsf{Con}(X) \otimes \mathsf{Con}(Y)$ implies

$$\forall (k,l).\, \exists i.\, \mathcal{L} \vdash (\phi_k \times \psi_l) \leq (\phi_i \times \psi_i) \ \ \&\ \ \forall i.\, \exists (k,l).\, \mathcal{L} \vdash (\phi_k \times \psi_l) \leq (\phi_i \times \psi_i).$$

Hence

$$\begin{array}{rcll}
\mathcal{L} \vdash (\bigvee_{k \in K} \phi_k \times \bigvee_{l \in L} \psi_l) & = & \bigvee_{(k,l) \in K \times L} (\phi_k \times \psi_l) & (\times - \vee) \\
& \leq & \bigvee_{i \in I} (\phi_i \times \psi_i) &
\end{array}$$

and so by $(\Box - \leq)$,

$$\mathcal{L} \vdash \Box(\bigvee_{k \in K} \phi_k \times \bigvee_{l \in L} \psi_l) \leq \Box \bigvee_{i \in I} (\phi_i \times \psi_i).$$

Again, by $(\diamond - \leq)$, for all i, for some (k,l):

$$\mathcal{L} \vdash \diamond(\phi_k \times \psi_l) \leq \diamond(\phi_i \times \psi_i)$$

and so

$$\mathcal{L} \vdash \bigwedge_{(k,l)\in K\times L} \diamond(\phi_k \times \psi_l) \leq \bigwedge_{i\in I} \diamond(\phi_i \times \psi_i).$$

Hence by $(\vdash - \leq)$,

$$M \otimes N, \Gamma \vdash \phi.$$

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,

$$\begin{array}{lll}
\bullet & \mathsf{fold}(M), \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![\mathsf{fold}(M)]\!]\rho_\Gamma & (4.1) \\
\Rightarrow & \alpha^{-1}(u) \sqsubseteq [\![M]\!]\rho_\Gamma & 4.3.3(\text{xiii}) \\
\Rightarrow & M, \Gamma \models \phi & (4.1) \\
\Rightarrow & M, \Gamma \vdash \phi & \text{ind. hyp.} \\
\Rightarrow & \mathsf{fold}(M), \Gamma \vdash \phi & (\vdash - \mathsf{rec} - I)
\end{array}$$

Recursion.

$$\begin{array}{lll}
\bullet & \mu x.M, \Gamma \models \phi & \\
\Rightarrow & u \sqsubseteq [\![\mu x.M]\!]\rho_\Gamma & (4.1) \\
\Rightarrow & \exists k \in \omega, u_0, \ldots, u_k.\ u_0 = \bot \ \&\ u_k = u & \\
 & \&\ \forall i : 0 \leq i < k.\ u_{i+1} \sqsubseteq [\![M]\!]\rho_\Gamma[x \mapsto u_i] & 4.3.3(\text{xv}).
\end{array}$$

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 $\Gamma x \neq f$[3] 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:

$$M, \Gamma \models \phi \quad \Leftrightarrow \quad M, \Gamma_V \models \phi \tag{4.2}$$
$$M, \Gamma \vdash \phi \quad \Leftrightarrow \quad M, \Gamma_V \vdash \phi \tag{4.3}$$

Now by Lemma 4.2.1, we have

$$\mathcal{L} \vdash \phi = \bigvee_{i \in I} \phi_i,$$

and for all $x \in V$,

$$\mathcal{L} \vdash \Gamma x = \bigvee_{j \in J_x} \psi_j,$$

with $\mathsf{C}(\phi_i)$, $\mathsf{C}(\psi_j)$ for each i, j. 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 $\psi_{f(x)}$, $f(x) \in J_x$, for each $x \in V$), we define Γ_f by:

$$\Gamma_f\, x = \begin{cases} \psi_{f(x)}, & x \in V \\ t & \text{otherwise.} \end{cases}$$

Then

- $M, \Gamma \models \phi$

$\Rightarrow$ $M, \Gamma_V \models \phi$ (4.2)

$\Rightarrow$ $\forall f \in \prod_{x \in V} J_x.\, M, \Gamma_f \models \bigvee_{i \in I} \phi_i$ $(\vdash - \leq)$, Soundness

$\Rightarrow$ $\forall f \in \prod_{x \in V} J_x.\, \exists i \in I.\, M, \Gamma_f \models \phi_i$

$\Rightarrow$ $\forall f \in \prod_{x \in V} J_x.\, \exists i \in I.\, M, \Gamma_f \vdash \phi_i$ Coprime 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) ∎

[3] meaning $[\![\Gamma x]\!] \neq \varnothing$, or, equivalently by Theorem 4.2.4, $\mathcal{L} \nvdash \Gamma x = f$

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

- $\mathsf{id} : (\sigma, \sigma)$
- $\dfrac{f : (\sigma, \tau) \quad g : (\tau, \upsilon)}{f; g : (\sigma, \upsilon)}$
- $1 : (\sigma, \mathbf{1})$
- $\dfrac{f : (\upsilon, \sigma) \quad g : (\upsilon, \tau)}{\langle f, g \rangle : (\upsilon, \sigma \times \tau)}$
- $\mathsf{p} : (\sigma \times \tau, \sigma)$
- $\mathsf{q} : (\sigma \times \tau, \tau)$
- $\dfrac{f : (\sigma \times \tau, \upsilon)}{\Lambda(f) : (\sigma, \tau \to \upsilon)}$
- $\mathsf{Ap} : ((\sigma \to \tau) \times \sigma, \tau)$
- $\mathsf{l} : (\sigma, \sigma \oplus \tau)$
- $\mathsf{r} : (\tau, \sigma \oplus \tau)$
- $\dfrac{f : (\sigma, \upsilon) \quad g : (\tau, \upsilon)}{[f, g] : (\sigma \oplus \tau, \upsilon)}$
- $\mathsf{up} : (\sigma, (\sigma)_\perp)$
- $\dfrac{f : (\sigma, \tau)}{\mathsf{lift}(f) : ((\sigma)_\perp, \tau)}$
- $\{\!| \cdot |\!\} : (\sigma, \mathcal{P}\sigma)$
- $\dfrac{f : (\sigma, \mathcal{P}\tau)}{f^\dagger : (\mathcal{P}\sigma, \mathcal{P}\tau)}$
- $+ : (\mathcal{P}\sigma \times \mathcal{P}\sigma, \mathcal{P}\sigma)$
- $\otimes : (\mathcal{P}\sigma \times \mathcal{P}\tau, \mathcal{P}(\sigma \times \tau))$
- $\mathsf{fold} : (\sigma[\mathsf{rec}\, t.\, \sigma/t], \mathsf{rec}\, t.\, \sigma)$
- $\mathsf{unfold} : (\mathsf{rec}\, t.\, \sigma, \sigma[\mathsf{rec}\, t.\, \sigma/t])$
- $\mathsf{Y} : (\sigma \to \sigma, \sigma)$

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 $\mathrm{DDL}(\sigma)$ 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 $\mathcal{L}$.

- $\dfrac{\phi \leq \psi}{[f]\phi \leq [f]\psi}$
- $[f]\bigwedge_{i\in I}\phi_i = \bigwedge_{i\in I}[f]\phi_i$
- $[f]\bigvee_{i\in I}\phi_i = \bigvee_{i\in I}[f]\phi_i$
- $[\mathsf{id}]\phi = \phi$
- $[f;g]\phi = [f][g]\phi$
- $[\langle f,g\rangle](\phi \times \psi) = [f]\phi \wedge [g]\psi$
- $[\mathsf{p}]\phi = (\phi \times t)$
- $[\mathsf{q}]\psi = (t \times \psi)$
- $\dfrac{(\phi \times \psi) \leq [f]\theta}{\phi \leq [\Lambda(f)](\psi \to \theta)}$
- $(\phi \to \psi) \times \phi \leq [\mathsf{Ap}]\psi$
- $[\mathsf{l}](\phi \oplus f) = \phi$
- $\dfrac{\mathsf{T}(\psi)}{[\mathsf{l}](f \oplus \psi) = f}$
- $\dfrac{\mathsf{T}(\phi)}{[\mathsf{r}](\phi \oplus f) = f}$
- $[\mathsf{r}](f \oplus \psi) = \psi$
- $\dfrac{\mathsf{T}(\phi) \quad \phi \leq [f]\psi}{(\phi \oplus f) \leq [[f,g]]\psi}$
- $\dfrac{\mathsf{T}(\phi) \quad \phi \leq [g]\psi}{(f \oplus \phi) \leq [[f,g]]\psi}$
- $[\mathsf{up}](\phi)_\perp = \phi$
- $\dfrac{\phi \leq [f]\psi}{(\phi)_\perp \leq [\mathsf{lift}(f)]\psi}$
- $[\{\!|\cdot|\!\}]\Diamond\phi = \phi$
- $[\{\!|\cdot|\!\}]\Box\phi = \phi$
- $\dfrac{\phi \leq [f]\Diamond\psi}{\Diamond\phi \leq [f^\dagger]\Diamond\psi}$
- $\dfrac{\phi \leq [f]\Box\psi}{\Box\phi \leq [f^\dagger]\Box\psi}$
- $[+]\Diamond\phi = (\Diamond\phi \times t) \vee (t \times \Diamond\phi)$
- $[+]\Box\phi = (\Box\phi \times \Box\phi)$
- $[\otimes]\Diamond(\phi \times \psi) = (\Diamond\phi \times \Diamond\psi)$
- $[\otimes]\Box(\phi \times \psi) = (\Box\phi \times \Box\psi)$
- $[\mathsf{fold}]\phi = \phi$
- $[\mathsf{unfold}]\phi = \phi$
- $\dfrac{\phi \leq [\mathsf{Y}]\psi}{\phi \wedge (\psi \to \theta) \leq [\mathsf{Y}]\theta}$

The rule for the Y combinator is rather subtle, describing recursion as a form of deductive closure; it is best understood by reading the corresponding case in the proof of the Soundness Theorem 4.4.1 below.

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 $(f)^\circ : \sigma \to \tau$.

Translation

$$
\begin{array}{rcl}
(\mathsf{id})^\circ & = & \lambda x.x \\
(f;g)^\circ & = & \lambda x.(g)^\circ((f)^\circ x) \\
(1)^\circ & = & \lambda x.\star \\
(\langle f,g\rangle)^\circ & = & \lambda x.((f)^\circ x,(g)^\circ x) \\
(\mathsf{p})^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \\
(\mathsf{q})^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,y \\
(\Lambda(f))^\circ & = & \lambda x.\lambda y.(f)^\circ(x,y) \\
(\mathsf{Ap})^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (f,x).\,fx \\
(\mathsf{l})^\circ & = & \lambda x.\imath(x) \\
(\mathsf{r})^\circ & = & \lambda y.\jmath(y) \\
([f,g])^\circ & = & \lambda z.\mathsf{cases}\ z\ \mathsf{of}\ \imath(x).\,(f)^\circ x\ \mathsf{else}\ \jmath(y).\,(g)^\circ y \\
(\mathsf{up})^\circ & = & \lambda x.\mathsf{up}(x) \\
(\mathsf{lift}(f))^\circ & = & \lambda y.\mathsf{lift}\ y\ \mathsf{to}\ \mathsf{up}(x).\,(f)^\circ x \\
(\{\!|\cdot|\!\})^\circ & = & \lambda x.\{\!|x|\!\} \\
(f^\dagger)^\circ & = & \lambda z.\mathsf{over}\ z\ \mathsf{extend}\ \{\!|x|\!\}.\,(f)^\circ x \\
(+)^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \uplus y \\
(\otimes)^\circ & = & \lambda z.\mathsf{let}\ z\ \mathsf{be}\ (x,y).\,x \otimes y \\
(\mathsf{fold})^\circ & = & \lambda x.\mathsf{fold}(x) \\
(\mathsf{unfold})^\circ & = & \lambda x.\mathsf{unfold}(x) \\
(\mathsf{Y})^\circ & = & \lambda f.\mu x.fx
\end{array}
$$

Semantics

Let $\mathcal{M}(\sigma,\tau)$ be the set of morphism terms of sort (σ,τ). Since

$$\mathbf{SFP}(\mathcal{D}(\sigma),\mathcal{D}(\tau)) \cong \mathcal{D}(\sigma \to \tau)$$

by cartesian closure, we can get a semantics

$$[\![\cdot]\!]_{\sigma\tau} : \mathcal{M}(\sigma,\tau) \longrightarrow \mathbf{SFP}(\mathcal{D}(\sigma),\mathcal{D}(\tau))$$

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:*

$$\mathcal{DDL} \vdash \phi \leq \psi \quad \Longrightarrow \quad \models \phi \leq \psi$$

Proof. The usual routine induction on the length of proofs. We give a few cases for illustration.

Left injection.

$$\begin{aligned} (i)\ [\![[\mathsf{l}](\phi \oplus f)]\!] &= ([\![\mathsf{l}]\!])^{-1}([\![(\phi \oplus f)]\!]) \\ &= \{d : \langle 0, d\rangle \in [\![(\phi \oplus f)]\!]\} \cup \{\bot : \bot \in [\![(\phi \oplus f)]\!]\} \\ &= [\![\phi]\!]. \end{aligned}$$

$$(ii)\ \mathsf{T}(\psi) \Rightarrow \bot \notin [\![\psi]\!] \Rightarrow ([\![\mathsf{l}]\!])^{-1}([\![(f \oplus \psi)]\!]) = \varnothing.$$

Source tupling. We verify the first rule.

$$\mathsf{T}(\phi) \Rightarrow \bot \notin [\![\phi]\!] \Rightarrow \forall d \in [\![\phi]\!].\ [\![[f,g]]\!](\langle 0, d\rangle) = f(d),$$

which implies

$$[\![\phi]\!] \subseteq [\![[f]\psi]\!] \ \Rightarrow\ [\![(\phi \oplus f)]\!] \subseteq [\![[[f,g]]\psi]\!].$$

Union.

$$\begin{aligned} (i)\quad [\![[+]\Diamond\phi]\!] &= \{(X,Y) : (X \cup Y) \cap [\![\phi]\!] \neq \varnothing\} \\ &= \{(X,Y) : X \cap [\![\phi]\!] \neq \varnothing \text{ or } Y \cap [\![\phi]\!] \neq \varnothing\} \\ &= \{(X,Z) : X \cap [\![\phi]\!] \neq \varnothing\} \\ &\quad \cup \{(Z,Y) : Y \cap [\![\phi]\!] \neq \varnothing\} \\ &= [\![(\Diamond\phi \times t) \vee (t \times \Diamond\phi)]\!] \\ (ii)\quad [\![[+]\Box\phi]\!] &= \{(X,Y) : X \cup Y \subseteq [\![\phi]\!]\} \\ &= \{(X,Y) : X \subseteq [\![\phi]\!] \;\&\; Y \subseteq [\![\phi]\!]\} \\ &= [\![(\Box\phi \times \Box\phi)]\!]. \end{aligned}$$

Recursion.

$$\begin{aligned} \bullet &\quad [\![\phi]\!] \subseteq [\![[\mathsf{Y}]\psi]\!] \\ \Rightarrow &\quad \forall f \in [\![\phi]\!].\ \mathsf{Y}f \in [\![\psi]\!] \\ \Rightarrow &\quad \forall f \in [\![\phi]\!] \cap [\![(\psi \to \theta)]\!].\ \mathsf{Y}f = f(\mathsf{Y}f) \in [\![\theta]\!]. \quad \blacksquare \end{aligned}$$

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*

$$\mathcal{DDL} \vdash \phi \leq [f]\psi \quad \Longleftrightarrow \quad \models \phi \leq [f]\psi.$$

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 [Kan80]. 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). 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, and under finite unions and intersections. 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 $\varphi : \mathbb{N} \to \mathbb{N}$, thought of as a strict continuous function of type $\mathbb{N}_\perp \to \mathbb{N}_\perp$, 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

$$[N]\bar{0} \leq [f][N]\bar{0}$$

where $\bar{0} \equiv ((t)_{\perp} \oplus f)$ (so $[\![\bar{0}]\!] = \{0\}$). ∎

We can summarize the effectivity of our various systems as follows:

- The basic system of Domain Logic presented in Section 2 is recursive (Theorem 4.2.7).
- The system of endogenous logic presented in Section 3, and the equivalent system of Hoare triples described in the present section, are r.e. (since they are recursively axiomatized) but not recursive; in fact, they are Σ^0_1-complete. (This is proved by a similar argument to the above Theorem).
- The full system of exogenous logic described in the present section is Π^0_2-complete.

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. Two substantive case studies of this kind have been carried out, in the areas of concurrency [Abr87a] and the λ-calculus [Abr88].

For example, in [Abr87a] we define a domain equation for synchronisation trees, and generate a logic which can be applied to the whole class of labelled transition systems. This logic subsumes Hennessy-Milner logic [HM85], and can be taken as a rational reconstruction of it. Furthermore, we *automatically* get a compositional proof theory for this logic, along the lines indicated above. Since one can define a denotational semantics for, e.g., SCCS [Mil83] in our denotational metalanguage, we get a compositional proof system along the lines of those developed by Stirling and Winskel [Sti87,Win85]. Moreover, this proof system is guaranteed to be in harmony with our semantics.

Chapter 5

Conclusion

5.1 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. All our work in this paper 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 [Abr87a] 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 [Abr87a]; 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.

2. 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].

3. The logics of compact-open sets considered in this paper are 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.

4. 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 [PS86]; 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].

5. Recent work by Axel Poigné [Poi87] raises the possibility of generalising our work in the setting of indexed category theory; it also suggests links with specification theory.

5.2 Related Work

We have already mentioned work by Smyth [Smy83b], Martin-Löf [ML83] and Plotkin [Plo81, Chapter 8]. Kozen's work on the representation of dynamic algebras [Koz80,Koz81] uses Stone duality ideas, but in a rather different spirit and setting. His work on probabilistic PDL [Koz83] is discussed above. Inspired by [Smy83b], Plotkin developed a predicate-transformer-style denotational metalanguage (for Scott domains), and proved its equivalence with a conventional metalanguage [Plo83]. Robinson has independently formulated some very similar ideas to our own [Rob88], although apparently in a less systematic and developed form.

The forthcoming book by Vickers [Vic88] includes an elegant exposition of much of the material of this paper, cast in a more algebraic style. It also gives a general introduction to localic topology, with motivation from both Mathematics and Computer Science.

5.3 Acknowledgements

My thanks to Per Martin-Löf, Gordon Plotkin, Axel Poigné, Mike Smyth, Steve Vickers and Glynn Winskel for very helpful discussions, and the U.K. Science and Engineering Research Council and the Alvey Programme for financial support. This paper is a revised version of Chapters 1–4 and 7 of [Abr87b]; an extended abstract appeared as [Abr87c].

Bibliography

[Abr] S. Abramsky. Total vs. partial objects in denotational semantics. To appear.

[Abr83] S. Abramsky. Semantic foundations for applicative multiprogramming. In J. Diaz, editor, *Automata, Languages and programming*, pages 1–14, Berlin, 1983. Springer-verlag. Lecture Notes in Computer Science Vol. 154.

[Abr87a] S. Abramsky. A domain equation for bisimulation. Chapter 5 of [Abr87b], 1987.

[Abr87b] S. Abramsky. *Domain Theory and the Logic of Observable Properties.* PhD thesis, University of London, October 1987.

[Abr87c] S. Abramsky. Domain theory in logical form. In *Symposium on Logic In Computer Science*, pages 47–53. Computer Society Press of the IEEE, 1987.

[Abr88] S. Abramsky. The lazy λ-calculus. In D. Turner, editor, *Declarative Programming.* Addison Wesley, 1988. To appear. Also Chapter 6 of [Abr87b].

[Acz77] Peter Aczel. An introduction to inductive definitions. In K. J. Barwise, editor, *The Handbook of Mathematical Logic*, Studies in Logic and Foundations of Mathematics. North Holland, 1977.

[AH87] S. Abramsky and C. L. Hankin, editors. *Abstract Interpretation for Declarative Languages.* Ellis Horwood, 1987.

[Bar77] K. J. Barwise. An introduction to first order logic. In K. J. Barwise, editor, *The Handbook of Mathematical Logic*, Studies in Logic and Foundations of Mathematics, pages 5–46. North Holland, 1977.

[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.

[BCDC83] H. Barendregt, M. Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type-assignment. *J. Symbolic Logic*, 48:931–940, 1983.

[BCL85] G. Berry, P.-L. Curien, 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.

[BW84] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer Verlag, Berlin, 1984.

[CDCHL84] 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] E. 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.

[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.

[Ers72] Yu. L. Ershov. Computable functionals of finite types. *Algebra and Logic*, 11(4):367–437, 1972.

[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] Jean-Yves 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.

[Gun85] C. Gunter. Profinite solutions for recursive domain equations. Technical Report CMU-CS-85-107, 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.

[Gun87] C. Gunter. Universal profinite domains. *Information and Computation*, 72(1):1–30, 1987.

[Har79] D. Harel. *First Order Dynamic Logic*, volume 68 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1979.

[HM85] M. C. B. Hennessy and Robin Milner. Algebraic laws for non-determinism 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.

[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*, Berlin, 1979. Springer-Verlag. 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*, Berlin, 1979. Springer-Verlag. Lecture Notes in Computer Science Vol. 74.

[Kan80] A. Kanda. *Fully Effective Solutions of Recursive Domain Equations*. PhD thesis, University of Warwick, 1980.

[Koz80] D. Kozen. A representation theorem for models of $\star$-free PDL. In J. W. de Bakker and J. van Leeuwen, editors, *7th International Colloquium on Automata, Languages and Programming*, pages 351–362, Berlin, 1980. Springer Verlag. Lecture Notes in Computer Science Vol. 85.

[Koz81] D. Kozen. On the duality of dynamic algebras and Kripke models. In *Workshop on Logics of Programs*, pages 1–11, Berlin, 1981. Springer Verlag. Lecture Notes in Computer Science Vol. 125.

[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.

[Law73] F. W. Lawvere. Metric spaces, generalised logic, and closed categories. In *Rend. del Sem. Mat. e Fis. di Milano*, 1973. Vol. XLIII.

[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, Berlin, 1984. Springer-Verlag. Lecture Notes in Computer Science Vol. 173.

[Mat85] S. Matthews. *Metric Domains for Completeness*. PhD thesis, University of Warwick, 1985.

[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.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[ML70] P. Martin-Löf. *Notes on Constructive Mathematics*. Almqvist and Wiksell, Stockholm, 1970.

[ML71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, Berlin, 1971.

[ML83] P. Martin-Löf. Lecture notes on the domain interpretation of type theory. In Programming Methodology Group, editor, *Workshop on the Semantics of Programming Languages*, Göteborg, Sweden, 1983. Chalmers University of Technology.

[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*, volume 109 of *Mathematical Centre Tracts*, 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.

[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"). Dept. of Computer Science, Univ. of Edinburgh, 1981.

[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, Berlin, 1982. EATCS, Springer-Verlag. Lecture Notes in Computer Science Vol. 140.

[Plo83] G. D. Plotkin. A Metalanguage for Predomains. In Programming Methodology Group, editor, *Workshop on the Semantics of Programming Languages*, Göteborg, Sweden, 1983. Chalmers University of Technology. Copies of slides for a lecture.

[Plo85] G. D. Plotkin. Lectures on predomains and partial functions. Notes for a course given at the Center for the Study of Language and Information, Stanford 1985, 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] A. Poigné. On specifications, theories and models with higher types. *Information and Control*, 68, 1986.

[Poi87] A. Poigné. Foundations are rich Institutions, but Institutions are poor Foundations. Draft paper, Imperial College, 1987.

[Pra81] V. R. Pratt. Dynamic logic. In J. W. de Bakker and J. van Leeuwen, editors, *Foundations of Computer Science III Part 2*, volume 109 of *Mathematical Centre Tracts*, pages 53–84. Centrum voor Wiskunde en Informatica, Amsterdam, 1981.

[PS86] G. Plotkin and C. Stirling. A framework for intuitionistic modal logics (extended abstract). In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 conference, March 19–22, Monterey, California*, pages 399–406. Morgan Kaufmann, 1986.

[Rob88] E. Robinson. Logical aspects of denotational semantics. In *Summer Conference on Category Theory and Computer Science, Edinburgh, 7th–9th September 1987*. Springer Verlag, 1988. Lecture Notes in Computer Science.

[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.

[Sco80] 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.

[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, Berlin, 1983. Springer-Verlag. 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.*, pages 37–111, 1936.

[Sto77] J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, 1977. The MIT Press Series in Computer Science.

[Vic87] S. J. Vickers. USCC. Draft paper, Imperial College, 1987.

[Vic88] S. J. Vickers. *Topology Via Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

[Win80] G. Winskel. *Events in Computation*. PhD thesis, University of Edinburgh, 1980.

[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, Berlin, 1985. Springer-Verlag. Lecture Notes in Computer Science Vol. 206.