Domains and Lambda-Calculi

Roberto M. Amadio
LIM, Marseille

Pierre-Louis Curien
LIENS, Paris

November 21, 1996

Contents

Preface

Denotational semantics is concerned with the mathematical meaning of programming languages. Programs (procedures, phrases) are to be interpreted in categories with structure (by which we mean sets and functions to start with, and suitable topological spaces and continuous functions to continue with). The main goals of this branch of computer science are, in our belief:

- To provide rigorous definitions that abstract away from implementation details, and that can serve as an implementation independent reference.

- To provide mathematical tools for proving properties of programs: as in logic, semantic models are guides in designing sound proof rules, that can then be used in automated proof-checkers like LCF.

Historically the first goal came first. In the sixties Strachey was writing semantic equations involving recursively defined data types without knowing if they had mathematical solutions. Scott provided the mathematical framework, and advocated its use in a formal proof system called LCF. Thus denotational semantics has from the beginning been applied to the two goals.

In this book we aim to present in an elementary and unified way the theory of certain topological spaces, best presented as order-theoretic structures, that have proved to be useful in the modelling of various families of typed λ-calculi considered as core programming languages and as meta-languages for denotational semantics. This theory is now known as Domain Theory, and has been founded as a subject by Scott and Plotkin.

The notion of continuity used in domain theory finds its origin in recursion theory. Here the works of Platek and Ershov come to mind: in Stanford and Novo-Sibirsk, independently, they both exploited the continuity properties of recursive functionals to build a theory of higher-order recursive functions. Recursion theory is implicit in the basics of domain theory, but becomes again explicit when *effective* domains are considered.

The topic is indebted to lattice theory and topology for ideas and techniques, however the aims are different. We look for theories that can be usefully applied to programming languages (and logic). Therefore a certain number of complications arise that are not usually considered. Just to name a few:

- The topologies we are interested in satisfy only a weak separation axiom (T_0). This stands in sharp contrast with classical topology based on T_2, or Hausdorff spaces, but it relates to the subject of pointless topology [Joh82].
- In applications it is difficult to justify the existence of a greatest element, hence the theory is developed without assuming the existence of arbitrary lub's, that is we will not work with complete lattices.
- There are several models of computation, certainly an important distinction is the possibility of computing in parallel or in series, hence the development of various notions of *continuous*, *stable*, and *sequential* morphisms.
- There is a distinction between an explicitly typed program and its untyped run time representation, hence the connection with *realizability interpretations.*

One of our main concerns will be to establish links between mathematical structures and more syntactic approaches to semantics, often referred to as operational semantics. The distinction operational vs. denotational is reminiscent of the distinction between "function as extension, or as a graph" (say, of a partial recursive function) and "function as a rule, or as an algorithm" (say, the specification of a Turing machine). The qualities of a denotational semantics can be measured in the way it matches an independently defined operational semantics. Conversely, an operational semantics, like any formal system, can be "blessed" by soundness or even completeness results with respect to some denotational semantics.

We shall therefore describe operational semantics as well as denotational semantics. In our experience it is essential to insist on these two complementary views in order to motivate computer scientists to do some mathematics and in order to interest mathematicians in structures that are somehow unfamiliar and far away from the traditional core of mathematics.

A description of the contents of each chapter follows. Unless stated otherwise we do not claim any novelty for the material presented here. We highlight this by mentioning some of the papers which were most influential in the writing of each chapter.

Chapter 1 introduces the first concepts in domain theory and denotational semantics: directed complete partial orders, algebraicity, Scott topology. A basic link between Scott continuity and computability (Myhill-Shepherdson theorem) is established. As an application, the denotational semantics of two simple imperative languages are presented, and are shown to be equivalent to their formal operational semantics [Sco72, Plo83].

Chapter 2 introduces the untyped λ-calculus. We establish several of the fundamental theorems of λ-calculus using a labelling technique due to Lévy. In this way we prove the Church-Rosser property, the standardization theorem, and the finite developments theorem. The same technique also yields the strong normalization property for the simply-typed λ-calculus. Finally, we show the Syntactic Continuity theorem (a further evidence of the role of continuity in the semantics of programming languages) and the Sequentiality theorem, which motivates the semantic study of sequentiality [Lev78, Ber79].

Chapter 3 is a case study of the fundamental domain equation $D = D \to D$, which provides models of the untyped λ-calculus. We detail the construction of the D_∞ models, obtained as suitable limits. The chapter is also a case study of Stone duality: the D_∞ models can also be constructed out of certain theories of "types", or functional characters [Bar84, CDHL82].

Chapter 4 is an introduction to the interpretation of simply-typed and untyped λ-calculi in categories. In particular we develop the categorical models of simply typed and type free λ-calculus and illustrate the techniques needed to prove the soundness and completeness of the related interpretations [LS86, Sco80]

Chapter 5 gives a complete presentation of the problem of classifying the largest cartesian closed categories of algebraic directed complete partial orders and continuous morphisms, which was solved by Jung, following earlier work by Smyth. Two important classes of algebraic cpo's come out of this study: bifinite domains, and L-domains [Jun88, Smy83a].

Chapter 6 presents the language PCF of Scott-Plotkin-Milner. This is a simply typed λ-calculus extended with fixpoints and arithmetic operators. For this calculus we discuss the *full abstraction* problem, or the problem of the correspondence between denotational and operational semantics [Sco93, Plo77].

Chapter 7 presents the basic apparatus for the solution of *domain equations*. It also includes material on the construction of *universal domains*, and on the representation of domains by *projections* [Sco72, SP82, DR93, Sco76, ABL86].

Chapter 8 studies λ-calculi endowed with a reduction strategy that stops at λ-abstractions. We analyse in particular a *call-by-value* λ-calculus and a λ-calculus with *control operators*. We introduce the semantic aspects of these calculi via a unifying framework proposed by Moggi and based on the idea of computation-as-monad [Plo75, Plo85, Mog89, Bou94].

Chapter 9 concentrates on *powerdomains* constructions (loosely speaking a powerset construction in domain theory) and their applications to the semantics of non-deterministic and concurrent computations. On the denotational side we develop the theory of Plotkin's convex powerdomain. On the syntactic side we introduce a process calculus (Milner's CCS) and its operational semantics based on the notion of *bisimulation*. We interpret CCS using a domain equation which involves the convex powerdomain and relate the denotational semantics to the operational one [Plo76, Mil89, Abr91a].

Chapter 10 presents Stone duality (originally the correspondence between Boolean algebras and certain topological spaces), applied to domains. Algebraic domains can be reconstructed from their compact elements, or from the opens of their Scott topology, which can be viewed as observable properties. Elements are then certain kinds of filters of properties. This idea can be exploited to the point of presenting domain theory in logical form, as advocated by Martin-Löf (a program which was carried out systematically by Abramsky) [Joh82, ML83, Abr91b].

Chapter 11 introduces the problem of the categorical interpretation of a typed λ-calculus with *dependent* and *second order* types along the lines established in chapter 4. We first develop some guidelines in a categorical framework, and then we apply them to the specific cases of categories of complete partial orders and Scott domains. Two popular fragments of this typed λ-calculus are considered in greater detail: the system LF of dependent types, and the system F of polymorphic types [Gir86, CGW88, AL87, Gir72, Rey74, HHP93].

Chapter 12 presents another theory of domains based on the notion of stable morphism. Stability was introduced by Berry, as an approximation of the sequential behaviour of λ-calculus. The definition of a stable function formalizes the property that there is always a *minimum* part of a given input needed to reach a given finite output. We develop the theory along the lines of chapters 1 and 5: we study stability on meet cpo's, dI-domains and event structures (and coherence spaces), stable bifinite domains (with an application to the construction of a retraction of all retractions), and continuous L-domains [Ber79, Ama91a, Ama95].

Chapter 13 is devoted to linear logic. The simplest framework for stability - coherence spaces - led Girard to the discovery of a new resource-sensitive logic. In linear logic, hypotheses, or data are consumed exactly once, and multiple uses (including no use) are re-introduced by explicit connectives. Linear logic has a rich model theory. We present only a few models: the stable model, Ehrhard's hypercoherence model (which is closer to capturing sequential behaviour), and Winskel's bistructures model (which combines the continuous and the stable models). Also continuity can be recast in the light of linear logic, as shown by Lamarche [Gir87, Ehr93, Win80].

Chapter 14 addresses the semantic notion of sequentiality, which is aimed at capturing sequential computation, as opposed to inherently parallel computation. We start with Kahn-Plotkin sequential functions, which do not lead to a cartesian closed category. But sequential algorithms, which are pairs (function, computation strategy) yield a model of PCF. They actually model, and are fully abstract for and extension of PCF with a control operator *catch*. Sequential algorithms lend themselves to a game interpretation. On the other hand, a term model of PCF, from which a fully abstract model of PCF is obtained by a quotient, can also be described in a syntax-independent way using games. Games semantics therefore appear as a powerful unifying framework, which is largely undeveloped at the time this book is written [Cur86, AJ92].

Chapter 15 is an elementary introduction to the ideas of *synthetic domain theory* via the category of partial equivalence relations (per). The category of per's is a useful tool in semantics; we exhibit an interpretation of system F, of a type assignment system, and of a subtyping system. Towards the interpretation of recursion we introduce various reflective subcategories of per's. In this context we prove a generalized Myhill-Shepherdson theorem [Hyl88, Ros86, FMRS92, Ama91b].

Chapter 16 discusses some connections between the functional and concurrent computational paradigms. As a main tool for this comparison we introduce the basics of π-calculus theory, a rather economical extension of Milner's CCS. We show that this calculus is sufficiently expressive to adequately encode a call-by-value λ-calculus enriched with parallel composition and synchronization operators [MPW92, ALT95].

Two appendices provide the basic material on recursion theory and category theory (see [Rog67, Soa87] for the former and [ML71, BW85, AL91] for the latter). We refer to [Bar84, GLT89] for more advanced results on the syntactic aspects of λ-calculus.

Most people never manage to read a scientific book from the beginning to the end. We guess this book will be no exception. In first approximation a precedence relation $\succ$ among the chapters can be defined as follows.

$$\begin{array}{l} 1,2 \succ 3 \succ 4 \succ 6 \succ 8 \succ 9 \succ 16 \\ 4 \succ 5 \succ 9,10 \\ 6 \succ 12 \succ 13 \succ 14 \\ 5 \succ 7 \succ 11 \succ 15 \end{array}$$

Clearly there are plenty of possible shortcuts. When using the book in an introductory graduate course or seminar it is perhaps a good idea to modulate the amount of domain-theoretical constructions which are presented.

This book arises out of a joint effort to develop and integrate lecture notes for graduate courses taught by the authors in the years 1991-1996 in a number of research institutions. A preliminary report of our work had appeared in [AC94]. Constructive criticisms and corrections are welcome and can be addressed to `amadio@gyptis.univ-mrs.fr` or `curien@dmi.ens.fr`.

Notation

Set theoretical.

$\emptyset$	empty set
ω	natural numbers
$\mathbf{B}$	two elements set
$\cup, \cap$	union, intersection of two sets
$\bigcup, \bigcap$	union, intersection of a family of sets
X^c	complement of X
$\mathcal{P}(X)$	parts of X
$\mathcal{P}_{fin}(X)$	finite subsets of X
$X \subseteq_{\mathit{fin}} Y$	X is a finite subset of Y
$X \subseteq^{\star}_{\mathit{fin}} Y$	X is a finite and non-empty subset of Y
$\sharp X$	cardinality of X
R^*	reflexive and transitive closure of R
$\{d_i\}_{i \in I}$	indexed set
$\{x_n\}_{n<\omega}$, $\{x_n\}_{n \in \omega}$, $\{x_n\}_{n \geq 0}$	equivalent notations for an enumerated set
$x \mapsto f(x), \lambda x.f(x)$	equivalent functional notations

Category theoretical.

$\mathbf{C}, \mathbf{D}$	categories
$\mathbf{C}[a, b]$	morphisms from a to b
$f \circ g$	composition of morphisms
$\langle f, g \rangle$	pairing of morphisms
$L \dashv R$	L left adjoint to R

Domain theoretical.

$(P, \leq)$	preorder (reflexive and transitive)
$f : (P, \leq) \rightarrow (P', \leq')$	f is monotonic if it preserves the preorder
$UB(X)$	upper bounds of X
$MUB(X)$	minimal upper bounds (mub's) of X
$\bigvee X$	least upper bound (lub)
$\bigwedge X$	greatest lower bound (glb)
$x \uparrow y$	elements with an upper bound (compatible elements)
$x \prec y$	immediate predecessor
$\uparrow X, \downarrow X$	smallest upper, lower set containing X
$\mathbf{O}$	poset $\{\perp, \top\}$ with $\perp < \top$
$d \rightarrow e$	step function
$\mathcal{K}(D)$	compact elements

Syntax.

BNF	Backus-Naur form for grammars
$V[U/x]$	substitution of U for x in V
$FV(M)$	free variables of M
$BT(M)$	Böhm tree of M
$\omega(M)$	syntactic approximant of M
$\vec{M}$	vector of terms

Semantics.

$f[e/d]$ environment update, $f[e/d](x) = \begin{cases} e & \text{if } x = d \\ f(x) & \text{otherwise} \end{cases}$

Recursion Theoretical.

$\{n\}, \phi_n$	function computed by the n-th Turing machine
$\downarrow, \uparrow$	convergence, divergence predicate
$\cong$	Kleene's equality on partially defined terms

Chapter 1

Continuity and Computability

As the computation of a computer program proceeds, some (partial) information is read from the input, and portions of the output are gradually produced. This is true of mathematical reasoning too. Consider the following abstraction of a typical highschool problem for simple equation solving. The student is presented with three numerical figures – the data of the problem (which might themselves be obtained as the results of previous problems). Call tem u, v, and w. The problem has two parts. In part 1, the student is required to compute a quantity x, and in the second part, using part 1 as a stepping stone, he is required to compute a quantity y. After some reasoning, the student will have found that, say, $x = 3u + 4$, and that $y = x - v$. Abstracting away from the actual values of u, v, w, x, and y, we can describe the problem in terms of information processing. We consider that the problem consists in computing x and y as a function of u, v, w, i.e., $(x, y) = f(u, v, w)$. A first remark is that w is not used (it was probably placed there on purpose to confuse the student...). In particular, if computing w was itself the result of a long, or even diverging, computation, the student would still be able to solve his problem. A second remark is that x depends on u only. Hence, again, if finding v is very painful, the student may still achieve at least part 1 of his problem. Finally, y depends on both u and v. If the actual value of y is needed to get the highest mark, then the student has no escape but to solve the other problem whose output is v.

We use the symbol $\perp$ to mark the absence of information. All what we have described with English words can be formalised as follows (we assume $u, v \neq \perp$):

$$\begin{array}{lcl} f(\perp,\perp,\perp) & = & (\perp,\perp) \\ f(u,\perp,\perp) & = & (3u+4,\perp) \\ f(u,v,\perp) & = & (3u+4,(3u+4)-v)\ . \end{array}$$

Input and output data may be ordered according to their information contents. Therefore we write:

$$\begin{array}{l} (\perp,\perp,\perp) \leq (u,\perp,\perp) \leq (u,v,\perp) \\ (\perp,\perp) \leq (3u+4,\perp) \leq (3u+4,(3u+4)-v)\ . \end{array}$$

The function f is monotonic with respect to this order, i.e., if $(x, y, z) \leq (x', y', z')$, then $f(x, y, z) \leq f(x', y', z')$. We are not concerned here with the order relation between numbers. It is not relevant in the analysis of the student's information processing activity. We are also confident that he or she is good at computing additions and multiplications (he might have a calculator...).

Another example involving an open-ended view of computation is offered by some popular programs running in the background at many academic institutions, which compute larger and larger prime numbers. In this case, larger and larger lists of prime numbers are obtained from scanning larger and larger portions of the (infinite) list of natural numbers, and making the appropriate primality checks. The currently produced list of prime numbers is an approximation of the infinite sorted list of all prime numbers, which is the ideal total output information. Continuity arises as the formalisation of the slogan: "any finite part of the output can be reached through a finite computation". The primality of an arbitrarily large number can be (in principle) checked in finite time and by scanning a finite portion of the sorted list of natural numbers.

Complete partial orders and continuous functions are introduced in section 1.1. The following two sections sketch links with topology and recursion theory. In section 1.2, we show that complete partial orders can be viewed as (quite special) topological spaces. In section 1.3, we indicate where all this came from: a theorem of recursion theory, due to Myhill and Shepherdson, shows that, in a suitable sense, computability implies continuity. In section 1.4, we come back to the order-theoretic treatment, and present basic domain constructions (product, function space, smash product, lifting, and different kinds of sums). In section 1.5, we apply the material of the previous sections to give a denotational semantics to a toy imperative language. In section 1.6, we consider a small extension of this language, and we introduce continuation semantics (continuations will be considered again in chapter 8).

1.1 Directed Completeness and Algebraicity

After giving the basic definitions concerning directed complete partial orders and continuous functions, we immediately arrive at a simple, but fundamental fixpoint theorem, which will be used to give meaning to loops (section 1.5) and to recursive definitions (section 6.1).

Definition 1.1.1 (dcpo) *Given a partial order $(D, \leq)$, a non-empty subset $\Delta \subseteq D$ is called* directed *if*

$$\forall\, x, y \in \Delta\ \ \exists\, z \in \Delta\ \ x \leq z \text{ and } y \leq z.$$

In the sequel, $\Delta \subseteq_{dir} D$ stands for: "Δ is a directed subset of D" (when clear from the context, the subscript is omitted). A partial order $(D, \leq)$ is called a directed complete partial order *(dcpo) if every $\Delta \subseteq D$ has a least upper bound (*lub*), denoted $\bigvee \Delta$. If moreover $(D, \leq)$ has a least element (written $\perp$), then it is called a* complete partial order *(cpo).*

Definition 1.1.2 (monotonic,continuous) *Let $(D, \leq)$ and $(D', \leq)$ be partial orders. A function $f : D \to D'$ is called monotonic if*

$$\forall x, y \in D \;\; x \leq y \Rightarrow f(x) \leq f(y).$$

If D and D' are dcpo's, a function $f : D \to D'$ is called continuous if it is monotonic and

$$\forall \Delta \subseteq_{dir} X \;\; f(\bigvee \Delta) = \bigvee f(\Delta).$$

(Notice that a monotonic function maps directed sets to directed sets.) A fixpoint *of $f : D \to D$ is an element x such that $f(x) = x$. A prefixpoint of $f : D \to D$ is an element x such that $f(x) \leq x$. If f has a least fixpoint, we denote it by $fix(f)$.*

The most noteworthy example of a directed set is an infinite ascending sequence $x_0 \leq x_1 \leq \cdots \leq x_n \cdots$. Actually they are the ones that matter. Most of domain theory can be formulated with partial orders that are complete only with respect to ascending chains.

Definition 1.1.3 (ω-dcpo) *A partial order $(D, \leq)$ is called an ω-dcpo if every ascending sequence $\{x_n\}_{n<\omega}$ has a lub.*

Clearly, dcpo's are ω-dcpo's. We stick to directed sets, which have a more abstract flavour.

Exercise 1.1.4 *Show that the identity functions are continuous, and that the composition of two continuous functions is continuous.*

Definition 1.1.5 *The category* **Dcpo** *is the category of directed complete partial orders and continuous functions. The category* **Cpo** *is the full subcategory of* **Dcpo** *whose objects are the cpo's.*

Example 1.1.6 *1. Given any set X, define $X_\perp = X \cup \{\perp\}$ (where $\perp \notin X$), and $x \leq y \Leftrightarrow (x = \perp$ or $x = y)$. Cpo's defined in this way are called* flat. *The two elements flat domain $\{\perp, \top\}$ is written* **O**. *The boolean flat domain $\{\perp, tt, ff\}$ is written* **T**.

2. All partial orders without infinite ascending chain are dcpo's (this includes all finite partial orders).

3. $X \rightharpoonup Y$ (the set of partial functions between two sets X, Y), endowed with the following order, is a cpo:

$$f \leq g \Leftrightarrow (f(x)\downarrow \Rightarrow (g(x)\downarrow \text{ and } f(x) = g(x)))$$

(where $f(x)\downarrow$ means "$f(x)$ is defined"), or equivalently $graph(f) \subseteq graph(g)$ (where $graph(f) = \{(x, y) \mid f(x)\downarrow$ and $f(x) = y\}$). The least element is the everywhere undefined function, and lub's are set-theoretic unions (of graphs).

The following proposition is the key to the interpretation of recursively defined programs or commands.

Proposition 1.1.7 (Kleene's fixpoint) *If D is a cpo and $f : D \rightarrow D$ is continuous, then $\bigvee_{n\in\omega} f^n(\perp)$ is a fixpoint of f, and is the least prefixpoint of f (hence it is the least fixpoint of f)* [1].

PROOF. From $\perp \leq f(\perp)$, we get by monotonicity that $\perp, f(\perp), \ldots, f^n(\perp), \ldots$ is an increasing chain, thus is directed. By continuity of f, we have

$$f(\bigvee_{n\in\omega} f^n(\perp)) = \bigvee_{n\in\omega} f^{n+1}(\perp) = \bigvee_{n\in\omega} f^n(\perp).$$

Suppose $f(x) \leq x$. We show $f^n(\perp) \leq x$ by induction on n. The base case is clear by minimality of $\perp$. Suppose $f^n(\perp) \leq x$: by monotonicity, $f^{n+1}(\perp) \leq f(x)$, and we conclude by transitivity. $\square$

More assumptions on D make it possible to prove the existence of least fixpoints for all monotonic functions.

Exercise 1.1.8 (Tarski's fixpoint) *Let D be a complete lattice (i.e., D is a partial order in which every subset has a lub). Show that any monotonic function $f : D \rightarrow D$ has a least fixpoint, which is $\bigwedge\{x \mid f(x) \leq x\}$, and that the set of fixpoints of f is a complete lattice.*

An alternative proof of the first part of Tarski's theorem appeals to a cardinality argument, as suggested in the following exercise.

Exercise 1.1.9 *Let D be a complete lattice, and $f : D \rightarrow D$ be a monotonic function. Set $f^0 = \perp$, $f^{\kappa+1} = f \circ f^\kappa$, and $f^\lambda(x) = \bigvee_{\kappa<\lambda} f^\kappa(x)$, for all x, where κ is an ordinal, and λ is a limit ordinal. Show that there is an ordinal μ such that $f^\mu = \mathit{fix}(f)$. Describe a dual construction for the greatest fixpoint.*

Next we introduce compact elements, which are used to model the notion of finite information.

Definition 1.1.10 (compact) *Let D be a dcpo. An element $d \in D$ is called compact (some authors say isolated) if the following implication holds, for each directed Δ:*

$$d \leq \bigvee \Delta \Rightarrow \exists x \in \Delta \;\; d \leq x.$$

We write $\mathcal{K}(D)$ for the collection of compact elements of D, and we let d, e range over compact elements.

Exercise 1.1.11 *Show that the lub of two compact elements, if any, is compact.*

[1]Why this fact is named after Kleene is explained in remark 1.3.3.

Definition 1.1.12 (algebraic) *A dcpo D is called algebraic if for all $x \in D$ the set $\{d \in \mathcal{K}(D) \mid d \leq x\}$, is directed and has lub x. It is called an ω-algebraic dcpo if it is algebraic and $\mathcal{K}(D)$ is denumerable. The elements of $\{d \in \mathcal{K}(D) \mid d \leq x\}$ are called the approximants of x, and $\mathcal{K}(D)$ is called the* basis *of D. We denote by* **Adcpo** *and* ω**Adcpo** *the full subcategories of* **Dcpo** *consisting of algebraic and ω-algebraic dcpo's, respectively. We also write*

$$\mathbf{Acpo} = \mathbf{Cpo} \cap \mathbf{Adcpo} \ , \ \omega\mathbf{Acpo} = \mathbf{Cpo} \cap \omega\mathbf{Adcpo}.$$

Thinking of a directed set Δ as describing the output of a possibly infinite computation, and of the elements of Δ as describing the larger and larger portions of the output produced as time passes, then the property of d being compact means that only a finite computing time is required to produce at least d. The algebraicity requirement says that we want to bother only about those abstract elements which can be described as the "limits" of their finite approximations.

Example 1.1.13 *1. We have seen that $X \rightharpoonup Y$ is a cpo. It is actually an algebraic cpo: the compact elements are the functions that have a finite domain of definition.*

2. The powerset of natural numbers, $\mathcal{P}(\omega)$, ordered by inclusion, is an ω-algebraic cpo.

3. Consider a signature Σ consisting of symbols f with an associated arity $arity(f)$. Define possibly infinite terms as partial functions S from $\omega^\star$ to Σ satisfying the following property:

$$S(un)\downarrow \Rightarrow \exists f \ \ S(u) = f \text{ with } n < arity(f).$$

The order is the restriction of the graph inclusion order on $\omega^\star \rightharpoonup \Sigma$.

4. The following is a minimal example of a non-algebraic cpo:

$$D = \omega \cup \{a, b\} \text{ with } x \leq y \ \text{ iff } \left\{ \begin{array}{l} y = b \text{ or} \\ x = a \text{ or} \\ x = m, y = n, \text{ and } m \leq n \ . \end{array} \right.$$

Exercise 1.1.14 *Let D be a dcpo, and $\mathcal{K} \subseteq \mathcal{K}(D)$ be such that for any $x \in D$ the set $\{d \in \mathcal{K} \mid d \leq x\}$ is directed and has lub x. Show that D is algebraic, and that $\mathcal{K} = \mathcal{K}(D)$.*

Exercise 1.1.15 *Define a notion of (ω)-algebraic ω-dcpo (cf. definition 1.1.3), and show that ω-algebraic ω-dcpo's and ω-algebraic dcpo's are the same.*

The following proposition formalises the idea that continuous means "finite input only is needed to produce finite output".

Proposition 1.1.16 ($\epsilon\delta$-continuity) *Let D and E be algebraic dcpo's.*

1. A function $f : D \to E$ is continuous iff it is monotonic, and for each $e \in \mathcal{K}(E)$ and $x \in D$ such that $e \leq f(x)$, there exists $d \leq x$ such that $d \in \mathcal{K}(D)$ and $e \leq f(d)$.

2. $\{(d, e) \in \mathcal{K}(D) \times \mathcal{K}(E) \mid e \leq f(d)\}$, denoted by graph$(f)$ and called graph *of f determines f entirely.*[2]

Proof. (1) We first prove ($\Leftarrow$). Let Δ be directed. We have $\bigvee f(\Delta) \leq f(\bigvee \Delta)$ by monotonicity. To show $f(\bigvee \Delta) \leq \bigvee f(\Delta)$, it is enough to prove that for any compact $e \leq f(\bigvee \Delta)$ there exists $\delta \in \Delta$ such that $e \leq f(\delta)$. By assumption there exists $d \leq \bigvee \Delta$ such that $d \in \mathcal{K}(D)$ and $e \leq f(d)$, and the conclusion follows by compactness of d. Conversely, if f is continuous and $e \leq f(x)$, take a directed $\Delta \subseteq \mathcal{K}(D)$ such that $x = \bigvee \Delta$. Then by continuity we can rephrase $e \leq f(x)$ as $e \leq \bigvee f(\Delta)$, and we conclude by compactness of e. For the second part of the statement, notice

$$f(x) = \bigvee \{e \mid e \leq f(x)\} = \bigvee \{e \mid \exists d \;\; d \leq x \text{ and } e \leq f(d)\}.$$

$\square$

Definition 1.1.17 (effective continuity) *If D and E are ω-algebraic dcpo's, and if two surjective enumerations $\{d_n\}_{n<\omega}$ and $\{e_n\}_{n<\omega}$ of the compact elements of D and E are given, then $f : D \to E$ is called effectively continuous iff it is continuous and the set $\{(m, n) \mid e_n \leq f(d_m)\}$ is recursively enumerable.*

Since a continuous function is determined in terms of compact elements, it is natural to ask for a characterisation of those sets of pairs that arise as graph of a continuous function.

Definition 1.1.18 (approximable relation) *If D and E are algebraic dcpo's, a relation $R \subseteq \mathcal{K}(D) \times \mathcal{K}(E)$ is called an approximable relation if it satisfies:*

$$\begin{array}{lll} (AR_1) & (d, e_1), (d, e_2) \in R & \Rightarrow \;\; e_1 \uparrow e_2 \\ (AR_2) & (d, e) \in R, d_1 \leq d, e \leq e_1 & \Rightarrow \;\; (d_1, e_1) \in R\ . \end{array}$$

Proposition 1.1.19 *The approximable relations are exactly the graphs of continuous functions.*

Proof. Clearly, the graph of a continuous functions satisfies (AR_1) and (AR_2). Conversely, let R be an approximable relation. We define f by

$$f(x) = \bigvee \{e \mid \exists\, d \leq x \;\; (d, e) \in R\}.$$

[2]This definition of *graph* is a variant of the set-theoretical definition used in example 1.1.6 (3), which is well suited for continuous functions over algebraic cpo's.

We show that f is well-defined. We have to check that $\{e \mid \exists\, d \leq x \;\; (d,e) \in R\}$ is directed. Let $(d_1, e_1), (d_2, e_2) \in R$, with $d_1, d_2 \leq x$. By algebraicity there exists a compact d such that $d_1, d_2 \leq d \leq x$. Then $(d, e_1), (d, e_2) \in R$ by AR_2, and $(d, e) \in R$ for some $e \geq e_1, e_2$ by AR_1, hence e fits. The proofs that f is monotonic and $graph(f) = R$ are easy. $\square$

Next we show how algebraic dcpo's correspond to a completion process, similar to that for obtaining real numbers from rationals.

Definition 1.1.20 (ideal) *Given a preorder $(P, \leq)$, an ideal I is a directed, lower subset of P. Write $Ide(P)$ for the collection of ideals over P, ordered by set-theoretic inclusion. An ideal I is called principal if $(\exists\, x \in P \;\; I = \downarrow x)$.*

Proposition 1.1.21 (ideal completion) *1. If P is a preorder, then $Ide(P)$ is an algebraic dcpo whose compact elements are exactly the principal ideals.*

2. If D is an algebraic dcpo, then D and $Ide(\mathcal{K}(D))$ are isomorphic in **Dcpo**.

PROOF. (1) (1) Let Δ be a directed set of ideals. Define $\bigvee \Delta$ as the set-theoretic union of the ideals in Δ. It is easily checked that this is an ideal, thus it is the lub of Δ in $Ide(P)$. The directedness of $\{\downarrow x \mid \downarrow x \subseteq I\}$ follows from the directedness of I. The rest of (1) follows from the following obvious facts: $I = \bigcup\{\downarrow x \mid \downarrow x \subseteq I\}$, and principal ideals are compact.

(2) The two inverse functions are $x \mapsto \{d \in \mathcal{K}(D) \mid d \leq x\}$ and $I \mapsto \bigvee I$. $\square$

Ideal completion is a universal construction, characterised by an adjunction. (The notion of adjunction is recalled in appendix B.)

Proposition 1.1.22 (ideal completion free) *Ideal completion is left adjoint to the forgetful functor U :* **Dcpo** $\rightarrow$ **P***, where* **P** *is the category of partial orders and monotonic functions, and where U takes a dcpo to the underlying partial order and a continuous function to the underlying monotonic function. Less abstractly, given any partial order X and any dcpo D, any monotonic function $f : X \rightarrow D$ extends uniquely to a continuous function $\hat{f} : Ide(X) \rightarrow D$.*

PROOF. We define the counity of the adjunction by $\eta(x) = \downarrow x$. Take a monotonic $f : X \rightarrow D$. The unique continuous extension $\hat{f}$ of f to $Ide(X)$ is defined by $\hat{f}(I) = \bigvee_{x \in I} f(x)$. $\square$

In a different perspective, ideal completion determines an equivalence of categories.

Proposition 1.1.23 *The ideal completion and the transformation $D \mapsto \mathcal{K}(D)$ determine an equivalence between* **Adcpo** *and the category of partial orders and approximable relations.*

PROOF. First we make sure that partial orders and approximable relations form a category. Composition is defined as graph composition:

$$R' \circ R = \{(d, d'') \mid \exists\, d' \ \ (d, d') \in R \text{ and } (d', d'') \in R'\}.$$

We only check that $R' \circ R$ satisfies AR_1. Let $(d, d_1') \in R$, $(d_1', d_1'') \in R'$, $(d, d_2') \in R$, and $(d_2', d_2'') \in R'$. Then

$$\begin{array}{ll} (d, d') \in R \text{ for some } d' \geq d_1', d_2' & \text{by } AR_1 \\ (d', d_1'') \in R', (d', d_2'') \in R' & \text{by } AR_2 \\ (d', d'') \in R' \text{ for some } d'' \geq d_1'', d_2'' & \text{by } AR_1\ . \end{array}$$

The rest of the proposition follows easily from propositions 1.1.21 and 1.1.19. □

Exercise 1.1.24 *Consider the finite partial terms over a signature $\Sigma \cup \{\Omega\}$ (disjoint union) including a special symbol Ω of arity 0, ordered as follows: $s \leq t$ iff $\vdash s \leq t$ can be established by the following rules:*

$$\frac{}{\vdash \Omega \leq t} \qquad \frac{\vdash s_1 \leq t_1 \ \cdots \ \vdash s_n \leq t_n}{f(s_1, \ldots, s_n) \leq f(t_1, \ldots, t_n)}$$

Show that the ideal completion of this partial order is isomorphic to the set of finite and infinite terms as defined in example 1.1.13.

1.2 Dcpo's as Topological Spaces

Any partial order $(X, \leq)$ may be endowed with a topology, called *Alexandrov topology*, whose open sets are the upper subsets of X. It has as basis the sets $\uparrow x$, where x ranges over X. Conversely, with every topological space $(X, \Omega X)$, one may associate a preorder, called *specialisation preorder*, defined by

$$x \leq y \ \text{ iff } \ \forall U \in \Omega X \ \ x \in U \Rightarrow y \in U.$$

A T_0 topology is by definition a topology whose associated preorder is a partial order, i.e., if $x \neq y$, then either there exists an open U such that $x \in U$ and $y \notin U$, or there exists an open U such that $y \in U$ and $x \notin U$. Classical topology assumes a much stronger separation axiom, known as T_2 or Hausdorff: if $x \neq y$, then there exist disjoint opens U and V such that $x \in U$ and $y \in V$. The topological spaces arising from dcpo's are not Hausdorff. They are not even T_1, where T_1 is the following intermediate property: if $x \neq y$, then there exists an open U such that $x \in U$ and $y \notin U$. (Clearly, if T_1 holds, then $x \leq y \Rightarrow x = y$.) We seek a T_0 topology associated with a dcpo in such a way that:

the specialisation order is the dcpo order, and
order-theoretic continuity coincides with topological continuity.

Recall that the opens of a topological space X are in one-to-one correspondence with the continuous functions $X \rightarrow \{\bot, \top\}$, where the only non-trivial open of $\{\bot, \top\}$ is $\{\top\}$. Precisely, the correspondence associates with an open its characteristic function, and maps any f to $f^{-1}(\top)$. The specialisation order for this topology on $\{\bot, \top\}$ yields the flat cpo $\mathbf{O}$ (cf. section 1). So the open sets of a dcpo D must be the sets of the form $f^{-1}(\top)$, for f continuous from D to $\mathbf{O}$, in the order-theoretical sense. This motivates the following definition.

Definition 1.2.1 (Scott topology) *A subset $A \subseteq D$ of a dcpo D is called Scott open if:*

1. *$x \in A$ and $x \leq y \Rightarrow y \in A$,*
2. *Δ directed and $\bigvee \Delta \in A \Rightarrow \exists x \in \Delta\ \ x \in A$.*

The collection $\Omega_S(D)$ of Scott opens (which is clearly a topology) is called Scott topology over D.

Exercise 1.2.2 *Show that $U_x = \{y \in D \mid y \not\leq x\}$ is Scott open.*

Lemma 1.2.3 *The specialisation order on (D, Ω_S) is $(D, \leq)$. In particular, Ω_S is T_0.*

PROOF. Call $\leq'$ the specialisation order. It is obvious from the definition of Scott topology that $\leq \subseteq \leq'$. Conversely, let $x \leq' y$ and suppose $x \not\leq y$, i.e., $x \in U_y$ (cf. exercise 1.2.2). Then $y \in U_y$ by definition of $\leq'$, contradicting reflexivity. □

Proposition 1.2.4 *Let D, E be dcpo's. The continuous functions (in the topological sense) from (D, Ω_S) to (E, Ω_E) are exactly the morphisms in* **Dcpo**.

PROOF. Let f be Ω_S-continuous. By lemma 1.2.3, f is monotonic (a continuous function is always monotonic with respect to the specialisation order). Suppose $f(\bigvee \Delta) \not\leq \bigvee f(\Delta)$, i.e., $\bigvee \Delta \in f^{-1}(U_{\bigvee f(\Delta)})$. Thus $f(\delta) \in U_{\bigvee f(\Delta)}$ for some $\delta \in \Delta$, since $f^{-1}(U_{\bigvee f(\Delta)})$ is Scott-open. But this contradicts $f(\delta) \leq \bigvee f(\Delta)$. The converse is easy and left to the reader. □

Proposition 1.2.5 (Scott basis) *If D is algebraic, then the sets $\uparrow d$, for d compact, form a basis of Ω_S.*

PROOF. The sets $\uparrow d$ are Scott-open, by definition of compactness. We have to show that if $\uparrow d \cap \uparrow d' \neq \emptyset$, then $\uparrow d'' \subseteq \uparrow d \cap \uparrow d'$, for some d'', that is, $d, d' \leq d''$. Let $x \in \uparrow d \cap \uparrow d'$, that is, $d, d' \in \{e \in \mathcal{K}(D) \mid e \leq x\}$. We find d'' by directedness. We also have to show that if U is open and $x \in U$, then $x \in \uparrow d \subseteq U$ for some d: this trivially follows from the definition of opens and by algebraicity. □

Exercise 1.2.6 gives a topological justification of ideal completion. Recall that opens of a topological space can be viewed as the morphisms from that space into $\mathbf{O}$. Suppose that we are interested in the dual exercise. We have an abstract topology, consisting a partial order of "opens" with arbitrary lub's and finite greatest lower bounds (glb's) distributing over them. Such a structure is called a frame. (The set of opens of a topological space, ordered by inclusion, is a frame.) Dually to the way of obtaining opens out of points, a way to recover points from (abstract) opens is to take the frame morphisms from A to $\mathbf{O}$ (i.e., those that preserve the frame structure), where $\mathbf{O}$ is considered as a frame. The construction that takes a topological space to its frame of opens, then to the set of points of this frame, is called soberification. All these notions of abstract topology will be developped in section 10.1.

Exercise 1.2.6 (ideals/points) *Let $(X, \leq)$ be a partial order. Show that ideals of X are in one-to-one correspondence with the points of the Alexandrov topology over X, i.e., the frame homomorphisms from ΩX to $\mathbf{O}$. In other words, ideal completion is an instance of soberification.*

1.3 Computability and Continuity

We give a recursion-theoretic characterisation of the set $(\omega \rightharpoonup \omega) \to_{\mathit{eff}} (\omega \rightharpoonup \omega)$ of effectively continuous functions from $\omega \rightharpoonup \omega$ to $\omega \rightharpoonup \omega$. Let $\{\phi_n\}_{n<\omega}$ be an enumeration of the set PR of partial recursive functions. We have

$$\mathcal{K}(\omega \rightharpoonup \omega) \subseteq PR \subseteq \omega \rightharpoonup \omega.$$

We recall theorem A.3.1: if A is a subset of PR such that $\{x \mid \phi_x \in A\}$ is recursively enumerable (r.e.), then for any partial recursive f:

$$f \in A \text{ iff there exists a finite function } \theta \leq f \text{ such that } \theta \in A.$$

In particular, A is an upper subset.

Theorem 1.3.1 (Myhill-Shepherdson) *1. Let f be a total recursive function that is extensional, i.e., $\phi_{f(m)} = \phi_{f(n)}$ whenever $\phi_m = \phi_n$. Then there is a unique continuous function $F : (\omega \rightharpoonup \omega) \to (\omega \rightharpoonup \omega)$ "extending" f, i.e., such that $F(\phi_n) = \phi_{f(n)}$ for all n. Moreover, F is effectively continuous.*

2. Conversely, any effectively continuous function $F : (\omega \rightharpoonup \omega) \to (\omega \rightharpoonup \omega)$ maps partial recursive functions to partial recursive functions, and there is a total (extensional) recursive function f such that $F(\phi_n) = \phi_{f(n)}$ for all n.

Proof. (1) Define $F_0 : PR \to PR$ by $F_0(\phi_n) = \phi_{f(n)}$. The key property of F_0 is:

$$(\star) \quad F_0(g)(m) \downarrow n \;\text{ iff }\; F_0(\theta)(m) \downarrow n \text{ for some finite } \theta \leq g \quad (g \in PR).$$

We get this by theorem A.3.1, taking $A = \{g \in PR \mid F_0(g)(m) \downarrow n\}$ (m, n fixed): a procedure in p that terminates when $\phi_p \in A$ is given by:

computing $f(p)$, and then,
computing $\phi_{f(p)}(m)$ and checking $\phi_{f(p)}(m) = n$.

Since F has to extend F_0, it extends a fortiori the restriction of F_0 to finite functions, thus there is no choice for the definition of F:

$$F(g)(m) \downarrow n \ \text{ iff } \ F_0(\theta)(m) \downarrow n \text{ for some finite } \theta \leq g \quad (g \in \omega \rightharpoonup \omega).$$

(Hereafter θ always ranges over finite functions.) We show that F is well defined. Suppose that $F_0(\theta)(m) \downarrow n$ and $F_0(\theta')(m) \downarrow n'$ for some finite $\theta, \theta' \leq g$. By $(\star)$, we have $F_0(g)(m) \downarrow n$ and $F_0(g)(m) \downarrow n'$, which forces $n = n'$. F extends F_0 by definition. It is also continuous by definition. We show finally that F is effectively continuous. A procedure in (encodings of) θ, θ', which terminates when $\theta' \leq F(\theta) = F_0(\theta)$, is obtained as a sequence of procedures in θ, which terminate when $F_0(\theta)(m) \downarrow n$, for all m, n such that $\theta'(m) = n$. Such procedures can be obtained by prefixing the procedure considered above with a (total) procedure taking θ to an index p such that $\theta = \phi_p$.

(2) Conversely, let F be effectively continuous. We build f as in the statement by a simple application of the s-m-n theorem A.1.5: it is enough to show that $(p, m) \mapsto F(\phi_p)(m)$ is partial recursive. This in turn is equivalent to proving that $F(\phi_p)(m) \downarrow n$ is r.e. in p, m, n. We know from the effectivity of the continuity of F that the predicate $F(\theta)(m) \downarrow n$ is r.e. in θ, m, n. Whence the following procedure for p, m, n: try in parallel the successive θ's, checking whether $\theta \leq \phi_p$ and $F(\theta)(m) \downarrow n$, and stop when one such θ has been found. Continuity guarantees that the procedure will succeed if $F(\phi_p)(m) \downarrow n$. $\square$

Exercise 1.3.2 *Let F be as in the statement of Myhill-Shepherdson's theorem 1.3.1. Show that the least fixpoint of F is in PR.*

Remark 1.3.3 *Forgetting about minimality, exercise 1.3.2 can be reformulated as follows: for any total and extensional recursive function $f : \omega \to \omega$, there exists n_0 such that $\phi_{f(n_0)} = \phi_{n_0}$. This is known as Kleene's recursion theorem. The proof followed here uses the (computable $\Rightarrow$ continuous) direction of theorem 1.3.1 and the proposition 1.1.7.*

1.4 Constructions on Dcpo's

In this section we show how to construct new dcpo's out of dcpo's. First we consider the product and function space constructions. Then we consider other basic domain constructions: lifting, smash product and sum.

Let D, E be two dcpo's. The product $D \times E$ of D and E in the category of sets becomes a product in the category of dcpo's (for the categorical notion of product, see appendix B), when endowed with the following componentwise order:

$$(x, y) \leq (x', y') \ \text{ iff } \ x \leq x' \text{ and } y \leq y'.$$

Proposition 1.4.1 (dcpo of pairs) *If D, E are dcpo's, then $D \times E$ ordered as above is a dcpo. The statement also holds, replacing "dcpo" by "cpo".*

PROOF. If Δ is directed in $D \times E$, define $\Delta_D = \{x \mid \exists y \ \ (x,y) \in \Delta\}$, and symmetrically Δ_E. Then $(\bigvee \Delta_D, \bigvee \Delta_E)$ is the lub of Δ. If D, E are cpo's, then $(\bot, \bot)$ is the minimum of $D \times E$. $\square$

If the dcpo's are algebraic, the product in **Dcpo** coincides with the product in **Top**, the category of topological spaces.

Exercise 1.4.2 *Let D, E be dcpo's, let Ω_S be the Scott topology on $D \times E$, and let τ be the product of the Scott topologies on D and E (a basis of τ is $\{U \times V \mid U, V$ Scott open$\}$). Show $\tau \subseteq \Omega_S$. Show that if D, E are algebraic, then $\tau = \Omega_S$. (See exercise 1.3.12 in [Bar84] for a situation where $\tau \neq \Omega_S$.)*

In general topology, it is not true that a continuous function of several arguments is continuous as soon as it is continuous in each argument, but this is true for dcpo's.

Proposition 1.4.3 (argumentwise continuity) *Let D, D', and E be dcpo's. A function $f : D \times D' \to E$ is continuous iff for all $x \in D$ the functions $f_x : D' \to E$, and for all $y \in D'$ the functions $f_y : D \to E$, defined by $f_x(y) = f(x,y)$ and $f_y(x) = f(x,y)$, respectively, are continuous.*

PROOF. Let $f : D \times D' \to E$ be continuous, and Δ be a directed subset of D'. Then $(x, \Delta) = \{(x, \delta) \mid \delta \in \Delta\}$ is a directed subset of $D \times D'$. Thus

$$f_x(\bigvee \Delta) = f(\bigvee(x, \Delta)) = \bigvee f(x, \Delta) = \bigvee f_x(\Delta).$$

Suppose conversely that f is continuous in each argument separately. Let Δ be directed in $D \times D'$. Let Δ_D and $\Delta_{D'}$ be as in the proof of proposition 1.4.1. Then

$$\begin{aligned} f(\bigvee \Delta) &= f(\bigvee \Delta_D, \bigvee \Delta_{D'}) = \bigvee f(\Delta_D, \bigvee \Delta_{D'}) \\ &= \bigvee \{\bigvee f(\delta, \Delta_{D'}) \mid \delta \in \Delta_D\} = \bigvee f(\Delta_D, \Delta_{D'}) \,. \end{aligned}$$

It remains to show $\bigvee f(\Delta_D, \Delta_{D'}) = \bigvee f(\Delta)$. One side is obvious since $\Delta \subseteq \Delta_D \times \Delta_{D'}$. Conversely, one uses directedness of Δ to check that each element of $\Delta_D \times \Delta_{D'}$ has an upper bound in Δ. $\square$

Next we consider the construction of function spaces.

Proposition 1.4.4 (dcpo of functions) *Let D, E be dcpo's. The set $D \to_{cont} E$ of continuous functions from D to E, endowed with the* pointwise ordering *defined by*

$$f \leq_{ext} f' \ \ \text{iff} \ \ \forall x \ \ f(x) \leq f'(x)$$

is a dcpo. (We shall omit the subscripts $_{cont}$ and $_{ext}$ until chapter 12.) Moreover, if E is a cpo, then $D \to E$ is a cpo.

Proof. Let Δ be a directed set of functions. Define $f(x) = \bigvee \Delta(x)$. Let Δ' be a directed subset of D. Then

$$\begin{array}{rcl} f(\bigvee \Delta') & = & \bigvee \Delta(\bigvee \Delta') = \bigvee\{\bigvee g(\Delta') \mid g \in \Delta\} = \bigvee \Delta(\Delta') \\ & = & \bigvee\{\bigvee \Delta(\delta') \mid \delta' \in \Delta'\} = \bigvee f(\Delta')\,. \end{array}$$

For the last part of the statement, notice that the constant function $\lambda x.\bot$ is the minimum of $D \to E$. □

Exercise 1.4.5 (*fix* **continuous)** *Show that the fixpoint functional fix* $: (D \to D) \to D$ *of proposition 1.1.7 is continuous.*

The material needed to show that $D \times E$ and $D \to E$ are categorical product and function spaces are collected in exercises 1.4.6 and 1.4.7. We refer to section 4.2, and in particular to exercises 4.2.11 and 4.2.12, for the full categorical treatment.

Exercise 1.4.6 *Show the following properties: (1) The projections π_1 and π_2, defined by $\pi_1(x,y) = x$ and $\pi_2(x,y) = y$, are continuous. (2) Given continuous functions $f : D \to E$ and $g : D \to E'$, the pairing $\langle f,g\rangle$ defined by $\langle f,g\rangle(x) = (f(x),g(x))$ is continuous.*

Exercise 1.4.7 *Show the following properties: (1) The evaluation defined by $ev(x,y) = x(y)$ is continuous. (2) Given $f : D \times D' \to E$, show that $\Lambda(f) : D \to (D' \to E)$ defined by $\Lambda(f)(x)(y) = f(x,y)$ is well-defined and continuous.*

What is the situation for algebraic dcpo's? Unfortunately, if D, E are algebraic, $D \to E$ may fail to be algebraic. The story seems to begin well, though. The following lemma shows how compact functions can be naturally constructed out of compact input and output elements.

Lemma 1.4.8 (step functions) *(1) Let D, E be cpo's, $d \in D$ and $e \in \mathcal{K}(E)$. Then the step function $d \to e$, defined as follows, is compact:*

$$(d \to e)(x) = \begin{cases} e & \textit{if } x \geq d \\ \bot & \textit{otherwise}\,. \end{cases}$$

(2) If D and E are algebraic, then $f = \bigvee\{d \to e \mid (d \to e) \leq f\}$, for any f.

Proof. (1) If $d \to e \leq \bigvee \Delta$, then $e = (d \to e)(d) \leq \bigvee\{f(d) \mid f \in \Delta\}$. Since e is compact, we get $e \leq f(d)$ for some f, i.e., $d \to e \leq f$.

(2) Notice that $\{d \to e \mid (d \to e) \leq f\} \leq g$ iff $(e \leq f(d) \Rightarrow e \leq g(d))$ for all d, e iff $f \leq g$. □

The trouble is that the sets $\{g \mid g \leq f$ and g compact$\}$ are not directed in general (see exercise 1.4.15). They become directed under a further assumption on the domains. The following observation motivates the next definition: if $d \to e \leq f$, $d' \to e' \leq f$, and $d \uparrow d'$, then also $e \uparrow e'$.

Definition 1.4.9 (Scott domain) *A dcpo satisfying the following axiom is called* bounded complete *(some authors say consistently complete):*

$$x \uparrow y \Rightarrow x \vee y \textit{ exists, for any } x \textit{ and } y.$$

Bounded complete and algebraic cpo's are often called Scott domains.

Exercise 1.4.10 *Show that a dcpo D is bounded complete iff any non-empty upper bounded subset of D has a lub iff any non-empty subset of D has a glb.*

Exercise 1.4.11 *Show that an algebraic dcpo is bounded complete iff $d \uparrow d' \Rightarrow d \vee d'$ exists, for any compacts d and d'.*

Suppose that E is bounded complete; then define, for compatible $d \to e$ and $d' \to e'$:

$$h(x) = \begin{cases} e \vee e' & x \geq d \text{ and } x \geq d' \\ e & x \geq d \text{ and } x \not\geq d' \\ e' & x \not\geq d \text{ and } x \geq d' \\ \bot & \text{otherwise .} \end{cases}$$

It is easily checked that h is the lub of $d \to e$ and $d' \to e'$.

Theorem 1.4.12 (Scott CCC) *If D is algebraic and E is a Scott domain, then $D \to E$ is a Scott domain. The compact elements of $D \to E$ are exactly the functions of the form $(d_1 \to e_1) \vee \cdots \vee (d_n \to e_n)$.*

Proof. Let Δ be the set of lub's of finite non-empty bounded sets of step functions (which always exist by a straightforward extension of the above construction of h). Then $f = \bigvee\{d \to e \mid (d \to e) \leq f\}$ implies $f = \bigvee\{g \in \Delta \mid g \leq f\}$, which shows that $D \to E$ is algebraic, since $\{g \in \Delta \mid g \leq f\}$ is directed by definition, and since Δ is a set of compact elements (cf. exercise 1.1.14). Bounded completeness for compact elements obviously follows from the definition of Δ. □

The terminology "CCC" is a shorthand for "cartesian closed category" (see appendix B and section 4.2).

Remark 1.4.13 *The lub's of finite sets of step functions, when they exist, are described by the following formula:*

$$((d_1 \to e_1) \vee \cdots \vee (d_n \to e_n))(x) = \bigvee\{e_i \mid d_i \leq x\}.$$

Exercise 1.4.14 *Show that an algebraic cpo is a lattice (i.e., it has all finite lub's and glb's) iff it has all finite lub's (cf. exercise 1.4.10). Show that if D, E are algebraic lattices, then so is $D \to E$.*

There are larger full subcategories of algebraic dcpo's and algebraic cpo's that are closed under the function space construction. This will be the subject matter of chapter 5.

Exercise 1.4.15 (non-algebraic $\rightarrow$) *Consider example (A) in figure 5.1 (ahead). Show that D is ω-algebraic and that $D \rightarrow D$ is not algebraic. Hints: (1) Show:*

$$\begin{array}{rcl} a \rightarrow a, b \rightarrow b \leq f \leq id & \Rightarrow & f(\overline{\omega}) \subseteq \overline{\omega}, f(a) = a \text{ and } f(b) = b \\ & \Rightarrow & \bigvee_{n \in \omega} f_n = f \end{array}$$

where

$$f_n(d) = \begin{cases} f(d) & \text{if } d \notin \omega \text{ or } (d = \overline{m} \text{ and } m \leq n) \\ f(\overline{m+1}) & \text{if } d = \overline{m} \text{ and } m > n\ . \end{cases}$$

(2) Notice that $f = f_m$ entails that f becomes constant, contradicting $f \leq id$. (3) Conclude that the set of approximants of the identity is not directed.

The following constructions play an essential role in the semantics of call-by-value, which is addressed in chapter 8, we introduce call-by-value semantics. Most of the proofs are easy and omitted.

Definition 1.4.16 (lifting) *Let D be a partial order. Its lifting $D_\perp$ is the partial order obtained by adjoining a new element $\perp$ (implicitly renaming the $\perp$ element of D, if any) below all the elements of D:*

$$x \leq y \text{ in } D_\perp \quad \Leftrightarrow \quad x = \perp \text{ or } (x, y \in D \text{ and } x \leq y \text{ in } D).$$

In particular, the flat domains, introduced in example 1.1.6, are liftings of discrete orders.

Definition 1.4.17 (partial continuous, strict) *Let D, E be dcpo's.*

1. A partial function $f : D \rightharpoonup E$ is called continuous if the domain of definition $dom(f)$ of f is Scott open, and if f restricted to $dom(f)$ is continuous (in the sense of either definition 1.1.1 [3] *or proposition 1.2.4).*

2. If D and E are cpo's, a continuous function $f : D \rightarrow E$ is called strict if $f(\perp) = \perp$.

3. If D, D', and E are cpo's, a continuous function $f : D \times D' \rightarrow E$ is called

$$\begin{array}{ll} \textit{left-strict} & \text{if } \forall x' \in D' \ \ f(\perp, x') = \perp, \\ \textit{right-strict} & \text{if } \forall x \in D \ \ f(x, \perp) = \perp. \end{array}$$

Given two dcpo's D, E, the following sets are in bijective correspondence (in fact, they are order-isomorphic):

1. the set of partial continuous functions from D to E,
2. the set of continuous functions from D to $E_\perp$,
3. the set of strict continuous functions from $D_\perp$ to $E_\perp$.

The transformations (all denoted as $f \mapsto \hat{f}$) are:

[3] More precisely: if $f(\bigvee \Delta) \Downarrow$, then $f(\bigvee \Delta) = \bigvee \{f(\delta) \mid \delta \in \Delta \text{ and } f(\delta) \Downarrow\}$ (notice that since $dom(f)$ is open, the right hand set is non-empty).

- (1) to (2): $\hat{f}(x) = \begin{cases} f(x) & \text{if } f(x) \Downarrow \\ \bot & \text{otherwise .} \end{cases}$
- (2) to (1): $\hat{f}$ is the restriction of f to $\{x \mid f(x) \neq \bot\}$ (notice that this set is open, cf. exercise 1.2.2).
- (2) to (3): $\hat{f}(x) = \begin{cases} f(x) & \text{if } x \neq \bot \\ \bot & \text{if } x = \bot \text{ .} \end{cases}$
- (3) to (2): $\hat{f}$ is the restriction of f to D.

The following proposition characterises this relationship in a more abstract manner. We define the image of a functor $F : \mathbf{C} \to \mathbf{C}'$ as the subcategory of $\mathbf{C}'$ whose objects are (the objects isomorphic to) Fa for some $a \in Ob_{\mathbf{C}}$, and whose arrows are the morphisms Ff for some morphism f of $\mathbf{C}$.

Proposition 1.4.18 (lifting as adjunction) *1. The lifting of a dcpo is a cpo. Lifting is right adjoint to the inclusion functor from* **Dcpo** *to the category* **Pdcpo** *of dcpo's and partial continuous functions.*

2. The lifting functor is faithful, and its image is the category **Scpo** *of cpo's and strict continuous functions.*

3. Lifting is left adjoint to the inclusion functor from **Scpo** *to* **Cpo**.

PROOF. We only show how (3) follows from (1) and (2) by categorical "abstract nonsense". Suppose that we have an adjunction $F \dashv G$, with $F : \mathbf{C} \to \mathbf{C}'$ and $G : \mathbf{C}' \to \mathbf{C}$, Then call $\mathbf{C_1}$ the image of G, and $\mathbf{C_2}$ the full subcategory of $\mathbf{C}$ whose objects are those of $\mathbf{C_1}$. There are inclusion functors $Inc_1 : \mathbf{C_1} \to \mathbf{C_2}$ and $Inc_2 : \mathbf{C_2} \to \mathbf{C}$. It is easy to see that $F \circ Inc_2 \dashv Inc_1 \circ G$. If moreover G is faithful, and faithful on objects (i.e., if $Ga' = Gb'$ implies $a' = b'$), then $G : \mathbf{C}' \to \mathbf{C_1}$ is actually an isomorphism of categories, so that, composing with G/G^{-1}, the adjunction becomes

$$G \circ F \circ Inc_2 \dashv Inc_1 \circ G \circ G^{-1} = Inc_1 .$$

If we take $\mathbf{C} = \mathbf{Dcpo}$, $\mathbf{C}' = \mathbf{Pdcpo}$, and the inclusion and lifting functors as F and G, respectively, we obtain (3). □

Remark 1.4.19 *The two adjunctions have actually nothing to do specifically with continuity, and can be reformulated in categories of partial orders and (partial) monotonic functions (see section 8.2).*

Definition 1.4.20 (smash product) *Let D and E be two cpo's. Their smash product is the subset $D \otimes E$ of $D \times E$ defined by*

$$D \otimes E = \{(x, y) \mid (x \neq \bot \textit{ and } y \neq \bot) \textit{ or } (x = \bot \textit{ and } y = \bot)\}$$

and ordered by the induced pointwise ordering.

Smash products enjoy a universal property.

Proposition 1.4.21 *1. The smash product of two cpo's D, D' is a cpo, and the function $\otimes : D \times D' \to D \otimes D'$ defined as follows is continuous:*

$$\otimes(x, x') = \begin{cases} (x, x') & \textit{if } (x, x') \in D \otimes E \\ (\bot, \bot) & \textit{otherwise}\,. \end{cases}$$

2. The function $\otimes$ is universal in the following sense: for any E and any continuous function $f : D \times D' \to E$ that is both left-strict and right-strict, there exists a unique strict continuous function $\hat{f} : D \otimes D' \to E$ such that $\hat{f} \circ \otimes = f$.

Several notions of sums have been used to give meaning to sum types.

Definition 1.4.22 (coalesced, separated sum) *Let D, E be two cpo's. Their coalesced sum $D + E$ is defined as follows:*

$$D + E = \{(1, x) \mid x \in D\backslash\{\bot\}\} \cup \{(2, y) \mid y \in E\backslash\{\bot\}\} \cup \{\bot\}.$$

The separated sum of D and E is defined as $D_\bot + E_\bot$.

Thus, in a colesced sum, the two $\bot$'s are identified, while in the separated sum, a new $\bot$ element is created and acts as a switch, because any two elements above $\bot$ are either incompatible or come from the same component D or E. None of these two sum constructors yields a categorical coproduct in **Cpo**. The situation is different in **Dcpo**.

Exercise 1.4.23 *Let D and E be two dcpo's. Show that their disjoint union, ordered in the obvious way, is a categorical coproduct in* **Dcpo**.

1.5 Toy Denotational Semantics

Let us illustrate the use of domains with a denotational semantics for a simple imperative language IMP, whose set of commands is given by the following syntax:

$$\textbf{Commands} \quad c ::= a \mid \textit{skip} \mid c; c \mid \textit{if } b \textit{ then } c \textit{ else } c \parallel \textit{while } b \textit{ do } c$$

where b and a range over two unspecified sets *Bexp* and *Act* of boolean expressions and of actions, respectively. The set of commands is written *Com*. We define the meaning of the commands of this language, first by means of rules, second by means of mathematical objects: sets and functions with structure. Thus we specify their operational and denotational semantics, respectively, as discussed in the preface. In IMP, these two semantics agree. We shall see later that it is difficult to achieve this goal in general (see section 6.4).

With the unspecified syntactic domains *Bexp* and *Act* we associate unspecified denotation functions $[\![\_]\!] : \textit{Bexp} \to (\Sigma \to \mathbf{B})$ and $[\![\_]\!] : \textit{Act} \to (\Sigma \to \Sigma)$, where Σ

$$\frac{[\![a]\!]\sigma = \sigma'}{\langle a, \sigma\rangle \to \sigma'} \qquad \frac{}{\langle \mathit{skip}, \sigma\rangle \to \sigma} \qquad \frac{\langle c_0, \sigma\rangle \to \sigma' \quad \langle c_1, \sigma'\rangle \to \sigma''}{\langle c_0; c_1, \sigma\rangle \to \sigma''}$$

$$\frac{[\![b]\!]\sigma = \mathit{tt} \quad \langle c_0, \sigma\rangle \to \sigma'}{\langle \mathit{if}\ b\ \mathit{then}\ c_0\ \mathit{else}\ c_1, \sigma\rangle \to \sigma'} \qquad \frac{[\![b]\!]\sigma = \mathit{ff} \quad \langle c_1, \sigma\rangle \to \sigma'}{\langle \mathit{if}\ b\ \mathit{then}\ c_0\ \mathit{else}\ c_1, \sigma\rangle \to \sigma'}$$

$$\frac{[\![b]\!]\sigma = \mathit{ff}}{\langle \mathit{while}\ b\ \mathit{do}\ c, \sigma\rangle \to \sigma} \qquad \frac{[\![b]\!]\sigma = \mathit{tt} \quad \langle c, \sigma\rangle \to \sigma' \quad \langle \mathit{while}\ b\ \mathit{do}\ c, \sigma'\rangle \to \sigma''}{\langle \mathit{while}\ b\ \mathit{do}\ c, \sigma\rangle \to \sigma''}$$

Figure 1.1: The operational semantics of Imp

is an unspecified set of states (for example an environment assigning values to identifiers), and $\mathbf{B} = \{\mathit{tt}, \mathit{ff}\}$ is the set of truth values.

The operational semantics of Imp is given by the formal system described in figure 1.1. In this figure, there are so-called judgments of the form $\langle c, \sigma\rangle \to \sigma'$, which should be read as: "starting with state σ, the command c terminates and its effect is to transform the state σ into the state σ'". A proof, or derivation, of such a judgment is a tree, all of whose nodes are instances of the inference rules. The rules show that Imp has no side effects. The evaluation of expressions does not change the state.

Lemma 1.5.1 (while rec) *Set $w = \mathit{while}\ b\ \mathit{do}\ c$. Then*

$$w \approx \mathit{if}\ b\ \mathit{then}\ (c; w)\ \mathit{else}\ \mathit{skip}$$

where $\approx$ is defined by: $c_0 \approx c_1$ iff $\forall \sigma, \sigma'\ \langle c_0, \sigma\rangle \to \sigma' \Leftrightarrow \langle c_1, \sigma\rangle \to \sigma'$.

Proof. By a simple case analysis on the last rule employed to show $\langle c, \sigma\rangle \to \sigma'$, where c stands for w and for *if* b *then* $c; w$ *else* *skip*, respectively. □

Exercise 1.5.2 *The following is a specified version of Bexp and Act (the actions are assignment commands, therefore we introduce a syntactic category Aexp of arithmetical expressions):*

Bexp	$b ::= \mathit{tt} \mid \mathit{ff} \mid e = e \mid e \leq e \mid \neg b \mid b \wedge b \mid b \vee b$
Aexp	$e ::= i \mid X \mid e + e \mid e - e \mid e \times e$
Act	$a ::= (X := e)$

where i ranges over the set ω of natural numbers, and X ranges over a set Loc of locations. The set Σ is defined by $\Sigma = \mathit{Loc} \to \omega$. (1) Complete the description of the operational semantics, by rules like:

$$\frac{\langle b, \sigma\rangle \to \mathit{tt}}{\langle \neg b, \sigma\rangle \to \mathit{ff}} \qquad \frac{}{\langle X, \sigma\rangle \to \sigma(X)}$$

(2) Prove that the evaluation of expressions is deterministic:

$$\langle e, \sigma\rangle \to m \text{ and } \langle e, \sigma\rangle \to n \Rightarrow m = n \text{ (similarly for Bexp)}$$

(hint: use structural induction, that is, induction on the size of expressions). (3) Prove that the evaluation of commands is deterministic:

$$\langle c, \sigma\rangle \to \sigma' \text{ and } \langle c, \sigma\rangle \to \sigma'' \Rightarrow \sigma' = \sigma''$$

(hint: use induction on the size of derivations). (3) Prove $\not\exists \sigma' \ \langle \textit{while } tt \textit{ do } c, \sigma\rangle \to \sigma'$*. (hint: reason by contradiction, with a minimal derivation).*

The denotational semantics of Imp is given by a function

$$[\![\_]\!] : \mathit{Com} \to (\Sigma \rightharpoonup \Sigma)$$

i.e., a function that associates with every command a partially defined function from states to states. This function extends the predefined $[\![\_]\!] : \mathit{Act} \to (\Sigma \to \Sigma)$. The semantics employs the partial functions type $\Sigma \rightharpoonup \Sigma$, because loops may cause non-termination, like in *while tt do skip*. The meaning of *skip* and command sequencing are given by the identity and by function composition, respectively. The meaning of conditionals is defined by cases. In other words, the meanings of these three constructs is an obvious rephrasing of the operational semantics:

$$\begin{array}{rcl}
[\![\mathit{skip}]\!]\sigma & = & \sigma \\
[\![c_0; c_1]\!]\sigma & = & [\![c_1]\!]([\![c_0]\!]\sigma) \\
[\![\mathit{if}\ b\ \mathit{then}\ c_0\ \mathit{else}\ c_1]\!]\sigma & = & \left\{ \begin{array}{ll} [\![c_0]\!]\sigma & \text{if } [\![b]\!]\sigma = tt \\ [\![c_1]\!]\sigma & \text{if } [\![b]\!]\sigma = \mathit{ff}\ . \end{array} \right.
\end{array}$$

The denotational meaning of *while* is a fixpoint construction suggested by lemma 1.5.1. The full definition of $[\![\_]\!]$ by structural induction is given in figure 1.2.

Theorem 1.5.3 (op/den equivalence) *The following equivalence holds, for any* c, σ, σ': $\langle c, \sigma\rangle \to \sigma' \Leftrightarrow [\![c]\!]\sigma = \sigma'$.

Proof hint. ($\Rightarrow$): This is easily proved by induction on derivations.

($\Leftarrow$): This is proved by structural induction, and in the *while* case, by mathematical induction. Let $[\![\mathit{while}\ b\ \mathit{do}\ c]\!]\sigma = \sigma'$, i.e., $\mathit{fix}(\Phi)(\sigma) = \sigma'$, where

$$\Phi = \lambda\phi.\mathit{cond} \circ \langle [\![b]\!], \langle \phi \circ [\![c]\!], \mathit{skip}\rangle\rangle.$$

Then since $\mathit{graph}(\mathit{fix}(\Phi)) = \bigcup_{n \geq 0} \mathit{graph}(\Phi^n(\bot))$, we have $(\sigma, \sigma') \in \mathit{graph}(\Phi^n(\bot))$ for some n. Hence it is enough to prove

$$\forall n \ (\Phi^n(\bot)(\sigma)\downarrow \Rightarrow \ \langle \mathit{while}\ b\ \mathit{do}\ c, \sigma\rangle \to \Phi^n(\bot)(\sigma))$$

$$\begin{array}{rcl}
[\![skip]\!] & = & id \\
[\![c_0; c_1]\!] & = & [\![c_1]\!] \circ [\![c_0]\!] \\
[\![if\ b\ then\ c_0\ else\ c_1]\!] & = & cond \circ \langle [\![b]\!], \langle [\![c_0]\!], [\![c_1]\!] \rangle \rangle \\
[\![while\ b\ do\ c]\!] & = & fix(\lambda \phi . cond \circ \langle [\![b]\!], \langle \phi \circ [\![c]\!], id \rangle \rangle)
\end{array}$$

- $\langle \_, \_ \rangle$ is the set-theoretical pairing of f and g (cf. exercise 1.4.6).
- $cond : \mathbf{B} \times (\mathbf{\Sigma} \times \mathbf{\Sigma}) \to \mathbf{\Sigma}$ is the conditional function: $cond(tt, (\sigma, \sigma')) = \sigma$, and $cond(f\!f, (\sigma, \sigma')) = \sigma'$.
- $fix : ((\Sigma \rightharpoonup \Sigma) \to (\Sigma \rightharpoonup \Sigma)) \to (\Sigma \rightharpoonup \Sigma)$ is the least fixpoint function (cf. proposition 1.1.7).

Figure 1.2: The denotational semantics of Imp

by induction on n. The base case is obvious, because $\Phi^0(\bot) = \bot$ has an empty graph. For the induction step, there are two cases:

1. $[\![b]\!]\sigma = tt$: then $\Phi^{n+1}(\bot)(\sigma) = \Phi^n(\bot)(\sigma')$, where $[\![c]\!]\sigma = \sigma'$; by induction $\langle b, \sigma \rangle \to tt$, $\langle c, \sigma \rangle \to \sigma'$, and $\langle while\ b\ do\ c, \sigma' \rangle \to \Phi^n(\bot)(\sigma')$. Hence, by the definition of the operational semantics:

$$\langle while\ b\ do\ c, \sigma \rangle \to \Phi^n(\bot)(\sigma') = \Phi^{n+1}(\bot)(\sigma).$$

2. $[\![b]\!]\sigma = f\!f$: then $\Phi^{n+1}(\bot)(\sigma) = \sigma$, and by induction $\langle b, \sigma \rangle \to f\!f$. Hence, by the definition of the operational semantics, $\langle while\ b\ do\ c, \sigma \rangle \to \sigma = \Phi^{n+1}(\bot)(\sigma)$. □

1.6 Continuation Semantics *

The language Imp lacks an essential feature of imperative programming: control operators, which allow to break the normal flow of a program. In chapter 8, we shall discuss control operators in the setting of functional programming. Here, we briefly present a simple imperative language Imp′, which is Imp extended with a *go to* statement. The commands of Imp′ are written using the following syntax:

$$\textbf{Commands} \quad c ::= a \mid skip \mid c; c \mid if\ b\ then\ c\ else\ c \mid goto\ l \mid l : c$$

where *Bexp* is as in Imp, and where l ranges over a set *Lab* of labels. We impose a further condition: in a command, any occurrence of *goto* must always be in the scope of a previously declared label: e.g. $(l : c); goto\ l$ is ruled out. Formally, we define the following simple deductive system, whose judgments have the form $L \vdash c$, where L is a (finite) subset of labels:

$$
\begin{array}{cc}
\dfrac{}{L \vdash a} & \dfrac{}{L \vdash \mathit{skip}} \\[2ex]
\dfrac{L \vdash c_0 \quad L \vdash c_1}{L \vdash c_0; c_1} & \dfrac{L \vdash c_0 \quad L \vdash c_1}{L \vdash \mathit{if}\ b\ \mathit{then}\ c_0\ \mathit{else}\ c_1} \\[2ex]
\dfrac{l \in L}{L \vdash \mathit{goto}\ l} & \dfrac{L \cup \{l\} \vdash c}{L \vdash l : c}
\end{array}
$$

The set Com' of the commands of language IMP$'$ is defined as the set of the commands c such that $L \vdash c$ for some L.

The semantics for IMP$'$ is more difficult than that of IMP. The inclusion of *goto* complicates the task of determining "what to do next". In our first language, a command acted as a state transformer, and handed its resulting state to the next command. The presence of *goto* creates a new situation. In a command $c_0; c_1$, c_0 may jump to a completely different area of the program, so that c_1 is possibly not the "next command". Consequently, we can no longer consider an interpretation where c_0 produces a state that c_1 can start with. An appropriate way of approaching the semantics of *goto* is by means of *continuations*. The main idea is that, since possible jumps make future – or continuation – of the program unpredictable, then future must become a parameter of the semantics. This guides us to the definition of the following sets:

$$
\begin{array}{rcl}
\mathit{Cont} & = & \Sigma \rightharpoonup \Sigma \\
\mathit{Env} & = & \mathit{Lab} \rightharpoonup \mathit{Cont}\ .
\end{array}
$$

Cont is called the set of command continuations, and *Env* is called the set of environments; ; θ, ρ range over *Cont*, *Env*, respectively. The semantic function $\llbracket \_ \rrbracket'$ for IMP$'$ has the following type:

$$\llbracket c \rrbracket' : \mathit{Env} \to (\mathit{Cont} \to \mathit{Cont}).$$

First, we define $\llbracket \_ \rrbracket'$ on the subset *Act* of Com', for which the function $\llbracket \_ \rrbracket : \mathit{Act} \to (\Sigma \to \Sigma)$ is available. Then the definition of $\llbracket \_ \rrbracket'$ is extended to Com'. The full definition is given in figure 1.3. The tests are interpreted with the predefined function $\llbracket \_ \rrbracket : \mathit{Bexp} \to (\Sigma \to \mathbf{B})$ of section 1.5.

Exercise 1.6.1 *Show that if $L \vdash c$, then $\llbracket c \rrbracket' \rho_1 = \llbracket c \rrbracket' \rho_2$ if $\rho_1(l) = \rho_2(l)$ for all $l \in L$.*

Exercise 1.6.2 (while/goto) *A while command can be encoded in* IMP$'$. *Specifically, the effect of (while b do c) can be achieved by (l : if b then (c; goto l) else skip). Use this encoding to define a translation $(\_)^\star$ from* IMP *to* IMP$'$, *and show $\llbracket c \rrbracket = \llbracket c^\star \rrbracket' \bot \mathit{id}$, for every $c \in \mathit{Com}$.*

We turn to the operational semantics of IMP$'$. The key idea is to implement continuations by using a stack to store them. (In chapter 8, similar techniques will be used to implement an extension of λ-calculus with control operators.) The judgments have the form $\langle c, \rho, S, \sigma \rangle \to \sigma'$, where c is a command, ρ is a partial function from labels to

$$\begin{array}{rcl}
[\![a]\!]'\rho\theta & = & \theta \circ [\![a]\!] \\
[\![skip]\!]' & = & id \\
[\![c_0; c_1]\!]'\rho & = & [\![c_0]\!]'\rho \circ [\![c_1]\!]'\rho \\
[\![if\ b\ then\ c_0\ else\ c_1]\!]'\rho\theta\sigma & = & cond([\![b]\!]\sigma, [\![c_0]\!]'\rho\theta\sigma, [\![c_1]\!]'\rho\theta\sigma) \\
[\![goto\ l]\!]'\rho\theta & = & \rho(l) \\
[\![l : c]\!]'\rho\theta & = & fix(\lambda\theta'.[\![c]\!]'\rho[\theta'/l]\theta)
\end{array}$$

Figure 1.3: The denotational semantics of IMP$'$

commands such that $dom(\rho) \vdash c$, S is a stack – or list – of pairs (c, ρ), and σ, σ' are states. It is convenient to use a slightly different syntax for commands c:

$$\begin{array}{l}
d ::= a \mid skip \mid if\ b\ then\ c\ else\ c \mid goto\ l \mid l : c \\
c ::= empty \mid d \cdot c\ .
\end{array}$$

In other words, sequencing is also treated stackwise, with *empty* acting as an end marker. The operational semantics is specified as follows:

$$\frac{[\![a]\!]\sigma = \sigma' \qquad \langle c, \rho, S, \sigma'\rangle \to \sigma''}{\langle (a; c), \rho, S, \sigma\rangle \to \sigma''} \qquad \frac{\langle c, \rho, S, \sigma\rangle \to \sigma''}{\langle (skip; c), \rho, S, \sigma\rangle \to \sigma'}$$

$$\frac{[\![b]\!]\sigma = tt \qquad \langle (c_0; c), \rho, S, \sigma\rangle \to \sigma'}{\langle (if\ b\ then\ c_0\ else\ c_1); c, \rho, S, \sigma\rangle \to \sigma'} \qquad \frac{[\![b]\!]\sigma = f\!f \qquad \langle (c_1; c), \rho, S, \sigma\rangle \to \sigma'}{\langle (if\ b\ then\ c_0\ else\ c_1); c, \rho, S, \sigma\rangle \to \sigma'}$$

$$\frac{\langle c_0, \rho[c_0/l], (c_1, \rho) \cdot S, \sigma\rangle \to \sigma'}{\langle (l : c_0); c_1, \rho, S, \sigma\rangle \to \sigma'} \qquad \frac{\langle \rho(l), \rho, S, \sigma\rangle \to \sigma'}{\langle (goto\ l); c, \rho, S, \sigma\rangle \to \sigma'}$$

$$\frac{\langle c, \rho, S, \sigma\rangle \to \sigma'}{\langle empty, \rho', (c, \rho) \cdot S, \sigma\rangle \to \sigma'}$$

Exercise 1.6.3 (op/den-IMP$'$) * *Show that the operational and denotational semantics of* IMP$'$ *agree in the following sense:* $\langle c, \rho, S, \sigma\rangle \to \sigma' \Leftrightarrow [\![c]\!][\![\rho]\!]_{[\![S]\!]}[\![S]\!]\sigma = \sigma'$, *where the meanings of syntactic environments* ρ *and of syntactic continuations* S *are defined as follows:*

$$\begin{array}{rcll}
[\![\rho]\!]_\theta(l) & = & [\![\rho(l)]\!][\![\rho]\!]_\theta\theta & \textit{(recursive definition, where } \theta \textit{ is a fixed parameter)} \\
[\![empty]\!] & = & id & \\
[\![(c, \rho) \cdot S]\!] & = & [\![c]\!][\![\rho]\!]_{[\![S]\!]}[\![S]\!]\ . &
\end{array}$$

In particular, we have $\langle c, \emptyset, empty, \sigma\rangle \to \sigma' \Leftrightarrow [\![c]\!] empty\, id\, \sigma = \sigma'$.

Chapter 2

Syntactic Theory of the λ-calculus

This chapter introduces the untyped λ-calculus. We establish some of its fundamental theorems, among which we count the syntactic continuity theorem, which offers another indication of the relevance of Scott continuity (cf. section 1.1 and theorem 1.3.1).

The λ-calculus was introduced around 1930 by Church as part of an investigation in the formal foundations of mathematics and logic. The related formalism of combinatory logic had been introduced some years ealier by Schönfinkel, and Curry. While the foundational program was later relativised by such results as Gödel's incompleteness theorem, λ-calculus nevertheless provided one of the concurrent formalisations of partial recursive functions. Logical interest in λ-calculus was resumed by Girard's discovery of the second order λ-calculus in the early seventies (see chapter 11).

In computer science, the interest in λ-calculus goes back to Landin [Lan66] and Reynolds [Rey70]. The λ-notation is also instrumental in MacCarthy's LISP, designed around 1960. These pioneering works have eventually lead to the development of functional programming languages like Scheme or Standard ML. In parallel, Scott and Strachey used λ-calculus as a metalanguage for the description of the denotational semantics of programming languages. The most comprehensive reference on λ-calculus is Barendregt's reference book [Bar84]. A more introductory textbook has been written by Hindley [HS86]. We refer to these books for more historical pointers.

In section 2.1, we present the untyped λ-calculus. The motivation to prove a strong normalisation theorem leads us to the simply typed λ-calculus, Typed λ-calculi, and extensions of them, will be considered later in the book, particularly in chapters 4, 11, 16. In section 2.2 we present a labelled λ-calculus which turns out to be a powerful tool for proving many fundamental theorems of the λ-calculus. One of them is the syntactic continuity theorem. The proof of this theorem is a bit technical, and is the subject of section 2.3. Finally, section 2.4

motivates the study of sequentiality, which will be undertaken in section 6.5 and chapter 14. Another fundamental theorem of the λ–calculus is Böhm's theorem. It is stated (but not proved) as theorem 3.2.10.

2.1 Untyped λ-Calculus

We present the λ-calculus, and its basic computation rule – the β-reduction. A proof of the confluence property is sketched, and the notion of standardisation is defined.

Definition 2.1.1 (λ-calculus) *The syntax of the untyped λ-calculus (λ-calculus for short) is given by*

$$M ::= x \mid MM \mid \lambda x.M$$

where x is called a variable, M_1M_2 is called an application, and $\lambda x.M$ is called an abstraction. The set of all λ-terms is denoted by Λ.

The following are frequently used abbreviations and terms:

$$\begin{array}{rcl} \lambda x_1 \cdots x_n.M & = & \lambda x_1.(\cdots \lambda x_n.M \cdots) \\ MN_1 \cdots N_n & = & (\cdots (MN_1) \cdots N_n) \end{array}$$

$$\begin{array}{ll} I = \lambda x.x & K = \lambda xy.x \\ \Delta = \lambda x.xx & S = \lambda xyz.(xz)(yz)\,. \end{array}$$

Definition 2.1.2 (head normal form) *A term $\lambda x_1 \cdots x_n.xM_1 \cdots M_p$, where x may or may not be equal to one of the x_i's, is called a head normal form (hnf for short).*

Remark 2.1.3 *Any λ-term has exactly one of the following two forms: either it is a hnf, or it is of the form $\lambda x_1 \cdots x_n.(\lambda x.M)M_1 \cdots M_p$ ($n \geq 0$, $p \geq 1$).*

We next introduce occurrences, which provide a notation allowing to manipulate subterms. Another tool for that purpose is the notion of context.

Definition 2.1.4 (occurrence) *Let M be a term, and u be a word over the alphabet $\{0,1,2\}$. The subterm of M at occurrence u, written M/u, is defined as follows:*

$$\frac{}{M/\epsilon = M} \qquad \frac{M/u = N}{\lambda x.M/0u = N}$$

$$\frac{M_1/u = N}{M_1M_2/1u = N} \qquad \frac{M_2/u = N}{M_1M_2/2u = N}$$

$$\begin{array}{rcl}
[\,]_i[N_1 \cdots N_n] & = & N_i \\
x[\vec{N}] & = & x \\
(C_1C_2)[\vec{N}] & = & C_1[\vec{N}]C_2[\vec{N}] \\
(\lambda x.C)[\vec{N}] & = & \lambda x.(C[\vec{N}])
\end{array}$$

Figure 2.1: Filling the holes of a context

where ϵ is the empty word. The term M/u may well not be defined. If it is defined, we say that u is an occurrence of M. The result of replacing the subterm M/u by another term N is denoted $M[N/u]$. We often write $M[N/u]$ just to say that $M/u = N$. We write:

- *$u \leq v$ (u is a prefix of v) if $\exists\, w\ \ (v = uw)$, and*
- *$u \not\uparrow v$ (u and v are disjoint) if neither $u \leq v$ nor $v \leq u$, or equivalently if $\not\exists\, w_1, w_2\ \ (uw_1 = vw_2)$.*

Example 2.1.5

$$(\lambda x.xy)/02 = y \quad (\lambda x.xy)[x/02] = \lambda x.xx$$

Definition 2.1.6 (context) *The contexts with numbered holes are defined by the following syntax (where $i \in \omega$):*

$$C ::= [\,]_i \mid x \mid CC \mid \lambda x.C\ .$$

If only one hole $[\,]_i$ occurs in a term, we denote it $[\,]$ for short.

In figure 2.1, we define the operation of filling the holes of a context by a (sufficiently long) vector of terms. Occurrences and contexts are related as follows.

Proposition 2.1.7 (occurrences / contexts) *For every term M and every occurrence u of M, there exists a unique context C with a unique hole occurring exactly once, such that $M = C[M/u]$. Such contexts are called* occurrence contexts.

Free occurrences of variables are defined in figure 2.2 through a predicate *Free*(u, M). We define *Bound*(u, v, M) (u is bound by v in M) by

$$\frac{M/v = \lambda x.P \quad u = v0w \quad M/u = x \quad \mathit{Free}(w, P)}{\mathit{Bound}(u, v, M)}$$

$$\frac{}{\mathit{Free}(\epsilon, x)}$$

$$\frac{\mathit{Free}(u, M)}{\mathit{Free}(1u, MN)} \qquad \frac{\mathit{Free}(u, N)}{\mathit{Free}(2u, MN)} \qquad \frac{\mathit{Free}(u, M) \qquad M/u \neq x}{\mathit{Free}(0u, \lambda x.M)}$$

Figure 2.2: Free occurrences

If we are not interested in the actual occurrences at which variables appear bound or free, we can define the sets $FV(M)$ and $BV(M)$ of free and bound variables of M by

$$\begin{array}{rcl} FV(M) & = & \{x \mid \exists\, u \;\; M/u = x \text{ and } \mathit{Free}(u, M)\} \\ BV(M) & = & \{x \mid \exists\, u, v \;\; M/u = x \text{ and } \mathit{Bound}(u, v, M)\} \,. \end{array}$$

If M is a term and $x \notin FV(M)$, one often says that x is fresh (relatively to M).

The definition of substitution of a term for a (free) variable raises a difficulty (there is a similar difficulty for the quantifiers in predicate calculus). We expect $\lambda y.x$ and $\lambda z.x$ to be two different notations for the same thing: the constant function with value x. But careless substitution leads to

$$(\lambda y.x)[y/x] = \lambda y.y \qquad (\lambda z.x)[y/x] = \lambda z.y \,.$$

We want $\lambda z.y$, not $\lambda y.y$, as the result. We thus have to avoid the capturing of free variables of the substituted term. This leads to the definition of substitution given in figure 2.3. The choice of z satisfying the side condition in the last clause of figure 2.3 is irrelevant: we manipulate terms up to the following equivalence $\equiv$, called α-conversion:

$$(\alpha) \quad C[\lambda x.M] \equiv C[\lambda y.(M[y/x])] \quad (y \notin FV(M))$$

for any context C and any term M.

The basic computation rule of λ-calculus is β-reduction.

Definition 2.1.8 (β-rule) *The β-rule is the following relation between λ-terms:*

$$(\beta) \quad C[(\lambda x.M)N] \to C[M[N/x]]$$

where C is an occurrence context and M, N are arbitrary terms. A term of the form $(\lambda x.M)N$ is called a redex. The arrow $\to$ may be given optional subscripts u (to witness the occurrence of the redex being reduced) or β (to clarify that the reduction is a β-reduction).

$$\begin{array}{lcll} x[N/x] & = & N & \\ y[N/x] & = & y & (y \neq x) \\ (M_1M_2)[N/x] & = & (M_1[N/x])(M_2[N/x]) & \\ (\lambda y.M)[N/x] & = & \lambda z.(M[z/y][N/x]) & (z \notin FV(M) \cup FV(N)) \end{array}$$

Figure 2.3: Substitution in the λ-calculus

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

$$(\nu)\ \frac{M \to M'}{MN \to M'N} \quad (\mu)\ \frac{N \to N'}{MN \to MN'} \quad (\xi)\ \frac{M \to M'}{\lambda x.M \to \lambda x.M'}$$

Figure 2.4: β-reduction

In figure 2.4, we give an alternative presentation of β-reduction, by means of an axiom and inference rules.

Definition 2.1.9 (derivation) *We denote by $\to_\beta^*$ (or simply $\to^*$) the reflexive and transitive closure of $\to_\beta$, and use $\to^+$ to express that at least one step is performed. The reflexive, symmetric, and transitive closure of $\to_\beta$ is denoted simply with $=_\beta$. A derivation is a sequence of reduction steps $M \to_{u_1} M_1 \cdots \to_{u_n} M_n$, written $D : M \to^* M_n$, with $D = u_1 \cdots u_n$.*

Example 2.1.10

$$\begin{array}{ll} II \to I & SKK \to^* I \\ \Delta\Delta \to \Delta\Delta & (\lambda x.f(xx))(\lambda x.f(xx)) \to f((\lambda x.f(xx))(\lambda x.f(xx))) \,. \end{array}$$

The last two examples show that there are infinite reduction sequences. Moreover, the last example indicates how fixpoints can be encoded in the λ-calculus. If we set

$$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

then we have $Yf =_\beta f(Yf)$.

Another rule, in addition to β, is often considered:

$$(\eta) \quad C[\lambda x.Mx] \to C[M] \quad (x \notin FV(M)).$$

This is an extensionality rule, asserting that every term is a function (if it is read backwards). The η-rule is not studied further in this chapter, but will be considered in chapter 4.

Remark 2.1.11 *Any reduction sequence starting from a head normal form*

$$\lambda x_1 \cdots x_n.xM_1 \cdots M_p$$

consists of an interleaving of independent reductions of $M_1, \ldots, M_p$. By this we mean:

$$(\lambda x_1 \cdots x_n.xM_1 \cdots M_p \rightarrow^* P) \Rightarrow \exists\, N_1, \ldots N_p \begin{cases} P = \lambda x_1 \cdots x_n.xN_1 \cdots N_p \text{ and} \\ \forall i \leq p \;\; M_i \rightarrow^* N_i\,. \end{cases}$$

We omit most details of the proofs of the next results (see e.g. [HS86]).

Lemma 2.1.12 *1. If $M \rightarrow M'$, then $M[N/x] \rightarrow M'[N/x]$.*

2. If $N \rightarrow N'$, then $M[N/x] \rightarrow^* M[N'/x]$.

Lemma 2.1.12 is the key to the proof of the following property, called local confluence.

Proposition 2.1.13 (local confluence) *The β-reduction is locally confluent: if $M \rightarrow N$ and $M \rightarrow P$, then $N \rightarrow^* Q$ and $P \rightarrow^* Q$ for some Q.*

The following is one of the fundamental theorems of the λ-calculus.

Theorem 2.1.14 (Church-Rosser) *The β-reduction is confluent: If $M \rightarrow^* N$ and $M \rightarrow^* P$, then $N \rightarrow^* Q$ and $P \rightarrow^* Q$ for some Q.*

PROOF HINT. In section 2.2, the theorem will be proved completely as a consequence of a powerful labelling method. Here we sketch an elegant direct proof due to Tait and Martin-Löf. A strongly confluent relation is a relation $\Rightarrow$ that satisfies

$$M \Rightarrow N, M \Rightarrow P \text{ implies } \exists\, Q \;\; N \Rightarrow Q \text{ and } P \Rightarrow Q.$$

By a straightforward paving argument, the strong confluence of a relation $\Rightarrow$ implies the confluence of $\Rightarrow^*$. Unfortunately, β-reduction is not strongly confluent:

$$\begin{array}{llll} (\lambda x. \cdots x \cdots x \cdots)N & \rightarrow & (\lambda x. \cdots x \cdots x \cdots)N' & \rightarrow & \cdots N' \cdots N' \cdots \\ (\lambda x. \cdots x \cdots x \cdots)N & \rightarrow & \cdots N \cdots N \cdots & \rightarrow^{\geq 2} & \cdots N' \cdots N' \cdots \end{array}$$

(by $\rightarrow^{\geq 2}$, we mean that the reduction from $\cdots N \cdots N \cdots$ to $\cdots N' \cdots N' \cdots$ takes at least two steps). But parallel reduction, defined in figure 2.5, is strongly confluent. In a parallel reduction, several redexes can be simultaneously reduced in one step. For example, we have $\cdots N \cdots N \cdots \Rightarrow \cdots N' \cdots N' \cdots$. Finally, the confluence of $\rightarrow$ easily follows from the following inclusions, which hold by definition of parallel reduction: $\rightarrow \subseteq \Rightarrow \subseteq \rightarrow^*$. □

The following exercise states a negative result due to Klop [Klo85].

$$\frac{}{M \Rightarrow M} \qquad \frac{M \Rightarrow M' \quad N \Rightarrow N'}{(\lambda x.M)N \Rightarrow M'[N'/x]}$$

$$\frac{M \Rightarrow M' \quad N \Rightarrow N'}{MN \Rightarrow M'N'} \qquad \frac{M \Rightarrow M'}{\lambda x.M \Rightarrow \lambda x.M'}$$

Figure 2.5: Parallel β-reduction

Exercise 2.1.15 ** Suppose that three constants D, F, S are added to the λ-calculus, together with the following new rewriting axiom:*

$$(SP) \quad D(Fx)(Sx) \to x.$$

Show that confluence fails for β+(SP). Hints [CH94]: (1) Consider the following so-called Turing fixpoint combinator:

$$Y_T = (\lambda xy.y((xx)y)(\lambda xy.y((xx)y).$$

The advantage of this term over Y (cf. example 2.1.10) is that $Y_T f$ is not only equal to, but reduces to, $f(Y_T f)$. Set $C = Y(\lambda xy.D(F(Ey))(S(E(xy))))$ and $B = YC$, where E is a free variable. (2) Notice that $B \to^* A$ and $B \to^* CA$, where $A = E(CB)$. Show that A and CA have no common reduct, by contradiction, taking a common reduct with a minimum number of E's in head position.

Another fundamental theorem of the λ-calculus is the standardisation theorem. It will fall out from the general technique of section 2.2, but we shall need part of it to develop this technique. As a first approximation, a reduction from M to N is standard when it does not reduce a redex if there is no need to reduce it. For example

$$(\lambda x.y)(\Delta\Delta) \to_2 (\lambda x.y)(\Delta\Delta) \to_\epsilon y$$

is not standard, because the final term y of the sequence could have been reached without reducing the redex at occurrence 2 in $(\lambda x.y)(\Delta\Delta)$, since we have, directly:

$$(\lambda x.y)(\Delta\Delta) \to_\epsilon y.$$

The standardisation theorem asserts that any derivation $M \to^* N$ can be transformed to a standard derivation from M to N. To formalise the notion of standard reduction, we need to define the notion of residual, which formalises what a redex in a term M becomes after the reduction of a different redex of M.

Definition 2.1.16 (residual) *If u, v are redex occurrences in a term M, and if $M \to_u N$, then v/u, the set of residuals of v after the reduction of u, is defined* *by*

$$v/u = \begin{cases} \{v\} & (u \not\uparrow v \text{ or } v < u) \\ \emptyset & (v = u) \\ \{uw'w \mid Bound(u10w', u1, M)\} & (v = u2w) \\ \{uw\} & (v = u10w)\ . \end{cases}$$

The notation is easily extended to V/D, where V stands for a set of redex occurrences, and D for a derivation.

$$\begin{array}{lcl} V/u & = & \bigcup\{v/u \mid v \in V\} \\ V/(uD) & = & (V/u)/D\ . \end{array}$$

Here is an informal description of v/u. Let $M/u = (\lambda x.P)Q$ and $M/v = (\lambda y.R)S$:

- The second case is obvious: a redex is entirely "consumed" when it is reduced.
- The first and the last cases of the definition correspond to the situation where the redex at v "remains in place".
  - If $u \not\uparrow v$, then $N/v = M/v$.
  - If $v < u$, then M/u is a subterm of R or S, say of R, and N/v has the form $(\lambda y.R')S$ for some R'.
  - If $v = u10w$, the occurrence of the redex at v has to be readjusted, and moreover the redex gets instantiated:

    $$\begin{array}{l} P/w = ((\lambda x.P)Q)/10w = M/v = (\lambda y.R)S \\ N/uw = (P)[Q/x]/w = (P/w)[Q/x] = (\lambda y.R[Q/x])S[Q/x]\ . \end{array}$$

- In the third case, the subterm at occurrence v is a subterm of Q, and gets copied by the substitution which replaces x by Q. In particular the redex $(\lambda y.R)S$ may be duplicated if there is more than one free occurrence of x in P. If on the contrary $x \notin FV(P)$, then v has no residual.

Example 2.1.17 *For $M = I((\lambda x.(Ix)x)(\lambda x.Ix))$, we have:*

$$220/2101 = \{220\} \quad \epsilon/2 = \{\epsilon\} \quad 2/2 = \emptyset \quad 2101/2 = \{21\} \quad 220/2 = \{212, 22\}.$$

Definition 2.1.18 (left) *If u, v are redex occurrences of M, we say that u is to the left of v if*

$$u < v \quad \text{or} \quad \exists\, w, u', v'\ (u = w1u' \text{ and } v = w2v')$$

or equivalently, if the first symbol of the redex at u is to the left of the first symbol of the redex at v.

Definition 2.1.19 (standard) *A derivation $D : M = M_0 \to_{u_1} M_1 \cdots \to_{u_n} M_n$ is called standard if*

$$\forall i, j \ \ 1 \leq i < j \leq n \Rightarrow \not\exists u \text{ to the left of } u_i \text{ such that } u_j \in u/D_{ij}$$

where $D_{ij} : M_{i-1} \to_{u_i} M_i \cdots \to_{u_{j-1}} M_{j-1}$. We then write $M \xrightarrow{stnd} {}^{*}M_n$. A special case of standard derivation is the normal *derivation, which always derives the leftmost redex. We write $M \xrightarrow{norm} {}^{*}N$ if N is reached from M by the normal derivation. We denote with $Val(M)$ the abstraction, if any, characterised by*

$$M_0 = M \xrightarrow{norm} {}^{*}\, Val(M) = M_n \text{ and } \forall\, i < n \ \ M_i \text{ is not an abstraction.}$$

Example 2.1.20 *The derivation $(\lambda x.y)(\Delta\Delta) \to_2 (\lambda x.y)(\Delta\Delta) \to_\epsilon y$ is indeed standard. Set $u_1 = 2$ and $u_2 = \epsilon$. Then $\epsilon = \epsilon/2 = \epsilon/D_{12}$, and ϵ is to the left of 2 in $(\lambda x.y)(\Delta\Delta)$.*

Lemma 2.1.21 *If $D : M \xrightarrow{stnd} {}^{*}\lambda x.N$, then D decomposes into*

$$M \xrightarrow{norm} {}^{*}\, Val(M) \xrightarrow{stnd} {}^{*}\lambda x.N.$$

PROOF. By induction on the length of D. If M is already an abstraction, then the statement holds vacuously. If $M = xM_1 \cdots M_p$, then all its reducts have the form $xN_1 \cdots N_p$, hence the statement again holds vacuously. If $M = (\lambda x.M)M_1 \cdots M_p$, and the first step in D does not reduce the leftmost redex, then the definition of standard implies that the terms in D all have the form $M = (\lambda x.P)P_1 \cdots P_p$. Hence the first step of D must be the first step of the normal derivation. The conclusion then follows by induction. □

2.2 The Labelled λ-Calculus

In this section, we introduce simple types, and show that simply typed terms are strongly normalisable. Next we introduce Lévy's labelled λ-calculus, and prove a more general strong normalisation theorem. The following fundamental theorems of the λ-calculus appear as simple consequences of this general theorem:

the confluence of β-reduction,
the standardisation theorem,
the finite developments theorem of the λ-calculus,
the syntactic continuity theorem.

In this section and in the following one, we follow [Lev78] and [Ber79].

Definition 2.2.1 (strongly normalisable) *A λ-term M is called strongly normalisable if there is no infinite β-derivation starting from M. We denote by SN the set of strongly normalisable expressions. A term which cannot be further reduced is called a* normal form.

Definition 2.2.2 (size,reduction depth) *The size of a term M is defined as follows:*

$$size(x) = 1 \ , \ size(MN) = size(M) + size(N) + 1 \ , \ size(\lambda x.M) = size(M) + 1.$$

If $M \in SN$, the maximal length of a derivation starting from M is called the reduction depth of M, and is denoted $depth(M)$.

Confluence and normalisation are the cornerstones of (typed) λ-calculus and rewriting theory. They ensure that any term has a unique normal form, which is a good candidate for being considered as the final result of the computation. The two properties[1] imply the decidability of the equality, defined as the reflexive, transitive and symmetric closure of $\rightarrow$. To decide whether two terms M, N are β-equal, reduce M, N to their normal form, and check whether these normal forms coincide (up to α-conversion). As a stepping stone for our next results, we show the standardisation theorem for strongly normalisable terms.

Lemma 2.2.3 *If $M \in SN$ and $M \rightarrow^* N$, then $M \xrightarrow{stnd}{}^* N$.*

Proof. By induction on $(depth(M), size(M))$. The only non-trivial case is $M = M_1M_2$.

• If $N = N_1N_2$ and $M_1 \rightarrow^* N_1$, $M_2 \rightarrow^* N_2$, then $M_1 \xrightarrow{stnd}{}^* N_1$, $M_2 \xrightarrow{stnd}{}^* N_2$ by induction, and we have $M_1M_2 \xrightarrow{stnd}{}^* N_1M_2 \xrightarrow{stnd}{}^* N_1N_2$.

• Otherwise, $M_1M_2 \rightarrow^* (\lambda x.N_1)N_2 \rightarrow N_1[N_2/x] \rightarrow^* N$, with $M_1 \rightarrow^* \lambda x.N_1$ and $M_2 \rightarrow^* N_2$. By induction and lemma 2.1.21:

$$M_1 \xrightarrow{norm}{}^* \lambda x.P \xrightarrow{stnd}{}^* \lambda x.N_1.$$

Hence $M_1M_2 \xrightarrow{norm}{}^+ P[M_2/x]$. Also, by lemma 2.1.12, $P[M_2/x] \rightarrow^* N$ follows from $P \rightarrow^* N_1$, $M_2 \rightarrow^* N_2$ and $N_1[N_2/x] \rightarrow^* N$. Hence, by induction, $P[M_2/x] \xrightarrow{stnd}{}^* N$. We conclude by observing that prefixing a standard derivation with a normal derivation yields a standard derivation. □

Lemma 2.2.4 *The following implication holds:*

$$M[N/x] \xrightarrow{stnd}{}^* \lambda y.P \Rightarrow \begin{cases} (M \rightarrow^* \lambda y.Q \text{ and } Q[N/x] \rightarrow^* P) \text{ or} \\ M \rightarrow^* M' = xM_1' \cdots M_n' \text{ and } M'[N/x] \rightarrow^* \lambda y.P \ . \end{cases}$$

Proof hint. The statement follows quite easily from lemma 2.1.21. □

We now engage in an attempt to show that any term M is strongly normalisable. We know by the example $\Delta\Delta \rightarrow \Delta\Delta$ that this property does *not* hold

[1]Actually, only the existence of an effective way to reduce a term to a normal form is sufficient for this purpose.

for arbitrary terms. But it holds for *typed* terms (and more generally for labelled terms whose labels are bounded, as we shall see in section 2.2). Types will be introduced right after we discover a failure in our proof attempt.

We proceed by induction on $size(M)$, as in the proof of lemma 2.2.3. We examine all the reduction paths originating from M. The only non-trivial case is $M = M_1M_2$. If M_1 and M_2 never interact, then we conclude by induction on the size. Otherwise, we have $M_1 \to^* \lambda x.N_1$, $M_2 \to^* N_2$, and $M \to^* N_1[N_2/x]$. By induction, N_1, N_2 are strongly normalisable. Hence, strong normalisation can be proved from the following property:

$$(\sigma SN) \quad M, N \in SN \Rightarrow M[N/x] \in SN.$$

Let us see how an attempt to prove (σSN) by induction on $(depth(M), size(M))$ fails. The only interesting case is

$$M = M_1M_2,\ M_1[N/x] \to^* \lambda y.P \text{ and } M_2[N/x] \to^* N_2$$

(as above). We want to prove:

$$(1) \quad P[N_2/y] \in SN.$$

By induction and lemma 2.2.3, we can apply lemma 2.2.4. We thus consider two cases:

(A) $M_1 \to^* \lambda y.Q$ and $Q[N/x] \to^* P$: Consider $M' = Q[M_2/y]$. The conclusion $P[N_2/y] \in SN$ follows from:

- $M \to^+ M'$, hence $depth(M') < depth(M)$, and $M'[N/x] \in SN$ by induction.
- $\left.\begin{array}{c} M'[N/x] = Q[N/x][M_2[N/x]/y] \\ Q[N/x] \to^* P \text{ and } M_2[N/x] \to^* N_2 \end{array}\right\} \Rightarrow M'[N/x] \to^* P[N_2/y].$

(B) $M_1 \to^* M' = xP_1 \cdots P_n$ and $M'[N/x] \to^* \lambda y.P$: This is where we get stuck. Think of $\Delta\Delta$.

To get around the difficulty, it would be enough to have a new measure ϕ for which we could show, in case (B):

$$(2) \quad \phi(N_2) < \phi(N).$$

Then we could carry the whole argument, by induction on

$$(\phi(N), depth(M), size(M)).$$

Let us briefly revisit the proof atttempt. Case (A) is unchanged, since the induction is applied to $M'[N/x]$, for which the first component of the ordinal is the same as for M and N. Case (B) is settled by the decreasing of the first component of the ordinal. The simply typed λ-calculus offers such a measure ϕ.

Definition 2.2.5 (simple types) *The simple types are defined by the following syntax:*

$$\sigma ::= \kappa \mid \sigma \to \sigma$$

where κ ranges over a collection K of basic types. The size of a type σ is defined by

$$size(\kappa) = 1 \quad size(\sigma \to \tau) = size(\sigma) + size(\tau) + 1.$$

In other words, types are built from a collection of basic types (like natural numbers, or booleans) by a unique constructor, the function space constructor. Next we introduce a syntax of raw typed terms.

Definition 2.2.6 (raw typed) *The raw simply-typed terms are λ-terms, all of whose occurrences are labelled by a type. Formally, they are the terms P declared by the following mutually recursive clauses:*

$$\begin{array}{l} M ::= x \mid PP \mid \lambda x.P \\ P ::= M^\sigma \ . \end{array}$$

To a raw typed term P we associate an untyped term by stripping all type superscripts. We denote the resulting term by $erase(P)$.

Definition 2.2.7 (typed) *The typed terms, or $\lambda^{\to}$-terms, are the raw typed terms P satisfying the following constraints:*

1. *All the free occurrences of x in P have the same superscript.*
2. *If $P = (M^{\sigma_1} M^{\sigma_2})^{\sigma_3}$, then $\sigma_1 = \sigma_2 \to \sigma_3$.*
3. *If $P = (\lambda x.M^{\sigma_1})^{\sigma_2}$, then $\sigma_2 = \sigma_3 \to \sigma_1$ for some σ_3, and all free occurrences of x in M have superscript σ_3.*

The typed β-reduction is defined by

$$(\beta^{\to}) \quad P[(\lambda x.M^\tau)^{\sigma\to\tau} N^\sigma/u] \to_u P[M^\tau[N^\sigma/x^\sigma]/u].$$

In this chapter, we consider typed λ-calculus only in passing, on our way to Lévy's labelled λ-calculus. Our presentation of the simply typed λ-calculus in definition 2.2.7 is rather ad hoc. A more standard presentation is by means of sequents. Natural deduction and sequent presentations of the simply typed λ-calculus are discussed in section 4.1.

Lemma 2.2.8 (subject reduction) *If $erase(M^\sigma) \to N$, then $M^\sigma \to N'^\sigma$ for some N'^σ such that $erase(N'^\sigma) = N$.*

Theorem 2.2.9 (strong normalisation – simple types) *In the simply typed λ-calculus all terms are $\beta^{\to}$-strongly normalisable.*

PROOF. The argument attempted above now goes through. We prove

$$(\sigma SN^{\rightarrow}) \quad M^\tau, N^\sigma \in SN \Rightarrow M^\tau[N^\sigma/x^\sigma] \in SN$$

by induction on $(size(\sigma), depth(M), size(M))$. The typed version of the crucial case (B) is:

$$\begin{array}{l} M^\tau = M_1^{\sigma'\to\tau} M_2^{\sigma'} \\ M_1^{\sigma'\to\tau} \to^* M'^{\sigma'\to\tau} = x^{\sigma_1\to\cdots\to\sigma_n\to\sigma'\to\tau} P_1^{\sigma_1}\cdots P_n^{\sigma_n} \\ M'^{\sigma'\to\tau}[N^\sigma/x^\sigma] \to^* \lambda y.P^\tau \\ M_2^{\sigma'}[N^\sigma/x^\sigma] \to^* N_2^{\sigma'} \; . \end{array}$$

with $\sigma = \sigma_1 \to \cdots \to \sigma_n \to \sigma' \to \tau$. Then $size(\sigma') < size(\sigma)$. Hence, defining $\phi(N^\sigma)$ as the size of the type of N, condition (2) holds. □

We now turn to a more general system of labels.

Definition 2.2.10 (labels) *We define a calculus of labels by the following syntax:*

$$\alpha ::= e \mid \underline{\alpha} \mid \overline{\alpha} \mid \alpha\alpha.$$

where e ranges over an alphabet E of atomic labels (E stands for "étiquette"). We let α, β range over labels. The height *of a label l is defined as follows:*

$$\begin{array}{l} height(e) = 0 \quad (e \in E) \\ height(\underline{\alpha}) = height(\overline{\alpha}) = height(\alpha) + 1 \\ height(\alpha\beta) = \max\{height(\alpha), height(\beta)\} \; . \end{array}$$

Labelled terms are defined in the same way as (raw) typed terms.

Definition 2.2.11 (labelled terms) *Labelled terms P are defined by the following mutually recursive syntax:*

$$\begin{array}{l} M ::= x \mid PP \mid \lambda x.P \\ P ::= M^\alpha \; . \end{array}$$

We write $\alpha \cdot M^\beta = M^{\alpha\beta}$, and $height(M^\alpha) = height(\alpha)$.

Substitution for labelled terms is defined in figure 2.6. We define $M[P/x]$, where M ranges over labelled terms and P ranges over labelled terms or unlabelled variables (the latter arise from α-conversion). As for the typed terms, the erasure of a labelled term is obtained by stripping off the labels. The labelled version $\beta^l_{\mathcal{P}}$ of β-reduction is defined relatively to a predicate $\mathcal{P}$ on labels:

$$(\beta^l_{\mathcal{P}}) \quad P[((\lambda x.P)^\alpha Q)^\beta/u] \to_u P[\beta \cdot \overline{\alpha} \cdot P[\underline{\alpha} \cdot Q/x]/u] \quad \text{if } \mathcal{P}(\alpha) \text{ holds.}$$

The label α is called the degree of the redex. Unrestricted labelled restriction is defined as the labelled reduction with respect to the full predicate consisiting of all labels.

$$\begin{array}{lll}
x^\alpha[P/x] & = \alpha \cdot P & (P \text{ is a labelled term}) \\
x^\alpha[z/x] & = z^\alpha & \\
y^\alpha[P/x] & = y^\alpha & (y \neq x) \\
(P_1P_2)^\alpha[P/x] & = (P_1[P/x]P_2[P/x])^\alpha & \\
(\lambda y.P)^\alpha[P/x] & = (\lambda z.M[z/y][P/x])^\alpha & (z \notin FV(M) \cup FV(N))
\end{array}$$

Figure 2.6: Substitution in the labelled λ-calculus

Definition 2.2.12 (q-bounded) *Let $q \in \omega$. A q-bounded predicate is a predicate $\mathcal{P}$ such that $(\forall\, \alpha \;\; \mathcal{P}(\alpha) \Rightarrow height(\alpha) \leq q)$.*

Theorem 2.2.13 (strong normalisation – labels) *If $\mathcal{P}$ is q-bounded for some q, then all labelled terms are strongly $\beta^l_\mathcal{P}$-normalisable.*

Proof hint. Similar to the proof of theorem 2.2.9. We prove

$$(\sigma SN^l_\mathcal{P}) \quad P, N^\alpha \in SN \Rightarrow P[N^\alpha/x] \in SN$$

by induction on $(q - height(\alpha), depth(P), size(erase(P)))$. The labelled version of the crucial case (B) is:

$$\begin{array}{l}
P = (P_1P_2)^\delta \\
P_1 \rightarrow^* P' = (\cdots(x^\gamma P_1)^{\alpha_1} \cdots P_n)^{\alpha_n} \\
P'[N^\alpha/x] \rightarrow^* (\lambda y.Q_1)^\beta \\
P_2[N^\alpha/x] \rightarrow^* Q_2\ .
\end{array}$$

Since $P[N^\alpha/x] \rightarrow^* ((\lambda y.Q_1)^\beta Q_2)^\delta \rightarrow \delta \cdot \overline{\beta} \cdot Q_1[\underline{\beta} \cdot Q_2/y]$, property (2) is rephrased here as $height(\alpha) < height(\underline{\beta} \cdot Q_2)$. A fortiori it is enough to prove $height(\alpha) \leq height(\beta)$ (notice the use of underlining). This will follow from the following claim:

Claim. $(\cdots(M^\zeta P_1)^{\beta_1} \cdots P_n)^{\beta_n} \rightarrow^* (\lambda y.Q)^\beta \Rightarrow height(\zeta) \leq height(\beta)$

We prove the claim by induction on the length of the derivation. The only interesting case is $M = \lambda x.P$. If $n = 0$, then $\zeta = \beta$. If $n > 0$, then

$$(\cdots(M^\zeta P_1)^{\beta_1} \cdots P_n)^{\beta_n} \rightarrow (\cdots(\beta_1 \cdot \overline{\zeta} \cdot P[\underline{\zeta} \cdot P_1/x]) \cdots P_n)^{\beta_n}$$

and we conclude by induction, since $height(\zeta) \leq height(\beta_1 \cdot \overline{\zeta} \cdot P[\underline{\zeta} \cdot P_1/x])$.

Applying the claim to $\zeta = \gamma\alpha$ and $M^\zeta = \gamma \cdot N^\alpha$, we get $height(\alpha) \leq height(\zeta) \leq height(\beta)$. □

Now we show how the confluence property and the standardisation theorem (general case) follow from theorem 2.2.13.

Lemma 2.2.14 *If $D : P \rightarrow^* Q$ and $v \in u/D$, then P/u and Q/v have the same degree.*

Proposition 2.2.15 *If $\mathcal{P}$ is q-bounded, then $\beta^l_{\mathcal{P}}$- reduction is confluent.*

PROOF. We get local confluence by proving a labelled version of lemma 2.1.12, using lemma 2.2.14. Then we use Newman's lemma (see exercise 2.2.16): $\beta^l_{\mathcal{P}}$ is confluent, since it is locally confluent and strongly normalising. □

Exercise 2.2.16 (Newman) *Prove that any locally confluent and strongly normalising system is confluent. Hint: use induction on the depth of the term from which the two derivations originate.*

Next we transfer labelled confluence to the unlabelled calculus. We start with a term M, which we label arbitrarily. That is, we construct P such that $erase(P) = M$. If $M \rightarrow^* M_1$ and $M \rightarrow^* M_2$, then, with unrestricted labelled reduction, we get

$$P \rightarrow^* P_1,\ P \rightarrow^* P_2 \text{ with } erase(P_1) = M_1,\ erase(P_2) = M_2.$$

Next we construct a predicate $\mathcal{P}$ that fits the situation: $\mathcal{P}(\alpha)$ iff α is the degree of a redex reduced in $P \rightarrow^* P_1$ or $P \rightarrow^* P_2$. This predicate is finite, hence bounded. Thus we can complete $P_1 \rightarrow^* P_3$, $P_2 \rightarrow^* P_3$ by $\beta^l_{\mathcal{P}}$-reduction, by proposition 2.2.15, and we get $M_1 \rightarrow^* erase(P_3)$ and $M_2 \rightarrow^* erase(P_3)$. This gives an alternative proof of theorem 2.1.14.

Theorem 2.2.17 (standardisation) *If $M \rightarrow^* N$, then $M \xrightarrow{stnd} {}^*N$.*

PROOF. By theorem 2.2.13 and (a labelled version of) lemma 2.2.3, using $\mathcal{P}$ defined as follows: $\mathcal{P}(\alpha)$ iff α is the degree of a redex reduced in $M \rightarrow^* N$. □

Corollary 2.2.18 (normal) *If M has a normal form N, then $M \xrightarrow{norm} {}^*N$.*

PROOF. By theorem 2.2.17, we have $M \xrightarrow{stnd} {}^*N$. If at some stage the leftmost redex was not reduced, it would have a residual in N: contradiction, since N is a normal form. □

Next we define the notion of development, and we prove the finite developments theorem.

Definition 2.2.19 (development) *A derivation $M \rightarrow_{u_1} M_1 \cdots \rightarrow_{u_n} M_n$ is relative to a set F of redex occurrences in M if $u_1 \in F$, and u_i is a residual of an occurrence in F, for all $i > 1$. If moreover M_n does not contain any residual of F, then $M \rightarrow_{u_1} M_1 \cdots \rightarrow_{u_n} M_n$ is called a complete development (or development for short) of M relative to F.*

Theorem 2.2.20 (finite developments-1) *Let M, F be as above. All reductions relative to F terminate, and they terminate on the same term.*

PROOF. Take P such that $erase(P) = M$, and define $\mathcal{P}$ by: $\mathcal{P}(\alpha)$ iff α is the degree of a redex of F. The conclusion then follows by lemma 2.2.14. □

In the following exercises, we propose a stronger version of the finite developments theorem, and we indicate how simple types can be related to labels.

Exercise 2.2.21 (finite developments-2) ** If $M \rightarrow_u N$ and v is a redex occurrence of N that is not a residual of a redex occurrence in M, we say that v is created by u. (1) Let α, β be the degrees of the redexes u in M and v in N. Show $height(\beta) > height(\alpha)$, (2) Show that if $D : M \rightarrow^* N$ and $D' : M \rightarrow^* N$ are two developments of M relative to F, then $G/D = G/D'$ for any derivation G originating from M. Hints: There are three cases of redex creation:*

$$\begin{array}{ll} (\lambda xy.M)N_1N_2 & (u = 1, v = \epsilon) \\ I(\lambda x.M)N & (u = 1, v = \epsilon) \\ (\lambda x.C[xN])(\lambda x.M) & (u = \epsilon, v = \textit{any occurrence of } [\,]) \;. \end{array}$$

Overlining is crucial in the proof of (1). Choose the initial labelling of M such that the degrees of F are distinct letters of E.

Exercise 2.2.22 ** Derive theorem 2.2.9 as a corollary of theorem 2.2.13. Hints: Take as E the finite collection of the types of the subterms of M (the term we start from). Define Ξ from labels to types by*

$$\Xi(\sigma) = \sigma \quad \frac{\Xi(\alpha) = (\sigma \rightarrow \tau)}{\Xi(\overline{\alpha}) = \tau} \quad \frac{\Xi(\alpha) = (\sigma \rightarrow \tau)}{\Xi(\underline{\alpha}) = \sigma} \quad \frac{\Xi(\alpha) = \Xi(\beta)}{\Xi(\alpha\beta) = \Xi(\alpha)}$$

Define $\mathcal{P}(\alpha)$ as $\Xi(\alpha)\downarrow$. For P whose labels all satisfy $\mathcal{P}$, define $\Xi(P)$ as the term obtained by applying Ξ to all labels. Say that P is well-behaved if all its labels satisfy $\mathcal{P}$ and if $\Xi(P)$ is well typed. Show that any $\beta^l_{\mathcal{P}}$-reduct Q of a well-behaved term P is well- behaved, and that $\Xi(P)$ $\beta^{\rightarrow}$-reduces to $\Xi(Q)$.

There exist many other proofs of finite developments, of confluence, and of standardisation. In exercise 2.2.23, we propose a particularly simple recent proof of finite developments due to Van Raamsdonk [vR96]. In exercises 2.2.24 and 2.2.25, we propose proofs of confluence and of standardisation based on finite developments.

Exercise 2.2.23 *Consider the set of underlined λ-terms, defined by the following syntax:*

$$M ::= x \mid MM \mid \lambda x.M \mid (\underline{\lambda} x.M)M.$$

Consider the least set $\mathcal{FD}$ of underlined terms containing the variables, closed under abstraction and application, and such that, for all underlined M, N:

$$(M[N/x] \in \mathcal{FD} \textit{ and } N \in \mathcal{FD}) \Rightarrow (\underline{\lambda} x.M)N \in \mathcal{FD}.$$

Show that $\mathcal{FD}$ is the set of all underlined terms, and exploit this to show the finite developments theorem. Hint: finite developments amount to the strong normalisation of underlined β-reduction $(\underline{\lambda} x.M)N \to M[N/x]$.

Exercise 2.2.24 *Show confluence as a consequence of finite developments. Hint: consider any development as a new notion of one step reduction.*

Exercise 2.2.25 *Show the standardisation theorem as a consequence of the finite developments theorem, by the following technique, which goes back to Curry. Let $D : M_0 \to_{u_0} M_1 \to_{u_1} M_1 \to^* M_n$ be the reduction sequence to standardise. Take a leftmost (cf. definition 2.1.18) occurrence u in the set of redex occurrences of M that have a residual reduced in D. Let $M \to_u M_{01}$, and build the reduction sequence $M_{01} \to^* M_{j1} = M_{j+1}$ where each step is a finite development of u/u_i, where (because u is leftmost) $M_i \to_u M_{1i}$, and where $u = u_j$. Continue the construction, applying it to the sequence $D_1 : M_{01} \to^* M_{j1} \to M_{j+2} \to^* M_n$, which is shorter than D.*

2.3 Syntactic Continuity *

Recall that a head normal form is a term of the form $\lambda x_1 \cdots x_n.xM_1 \cdots M_p$. We define an algebraic (or symbolic) semantics for the λ-calculus. We interpret the (finite) λ-terms as (potentially) infinite terms. For this purpose, we need to introduce partial terms (cf. exercise 1.1.24) to allow for a description of finite approximations.

Definition 2.3.1 (Böhm trees) *The set $\mathcal{N}$ is defined by*

$$\frac{}{\Omega \in \mathcal{N}} \qquad \frac{A_1 \in \mathcal{N} \ \cdots \ A_p \in \mathcal{N}}{\lambda x_1 \cdots x_n.xA_1 \cdots A_p \in \mathcal{N}}$$

$\mathcal{N}$ is a subset of the set of partial λ-terms, also called Ω-terms, defined by

$$M ::= \Omega \mid x \mid MM \mid \lambda x.M$$

and inherits its order, defined by:

$$\frac{}{\Omega \leq M} \qquad \frac{M_1 \leq M_1' \quad M_2 \leq M_2'}{M_1M_2 \leq M_1'M_2'} \qquad \frac{M \leq M'}{\lambda x.M \leq \lambda x.M'}$$

The elements of the ideal completion (cf. proposition 1.1.21) $\mathcal{N}^\infty$ of $\mathcal{N}$ are called Böhm trees. For any λ-term M, we define $\omega(M) \in \mathcal{N}$, called immediate aproximation *of M, as follows:*

$$\omega(M) = \begin{cases} \Omega & \text{if } M = \lambda x_1 \cdots x_n.(\lambda x.M)M_1 \cdots M_p \\ \lambda x_1 \cdots x_n.x\omega(M_1) \cdots \omega(M_p) & \text{if } M = \lambda x_1 \cdots x_n.xM_1 \cdots M_p \end{cases}$$

where $p \geq 1$ is assumed in the first case. The function ω is extended to Ω-terms by setting $\omega(\lambda x_1 \cdots x_n.\Omega M_1 \cdots M_p) = \Omega$.

Lemma 2.3.2 *If $M \to N$, then $\omega(M) \leq \omega(N)$.*

PROOF. By induction on the size of M. If $M = \lambda x_1 \cdots x_n.xM_1 \cdots M_p$, then the reduction occurs in one of the M_i's, and induction can be applied. $\square$

Definition 2.3.3 *For any λ-term M we define $BT(M) = \bigvee\{\omega(N) \mid M \rightarrow^* N\}$. $BT(M)$ is called the Böhm tree of M.*

By lemma 2.3.2 and by confluence, $\{\omega(N) \mid M \rightarrow^* N\}$ is directed, for fixed M, hence Böhm trees are well defined. The immediate approximation $\omega(M)$ can be understood as the current approximation of $BT(M)$, obtained (roughly) by replacing the redexes with Ω's. It is sometimes called a partial normal form. If computation proceeds, with $M \rightarrow^* N$, then $\omega(N)$ may be a better partial normal form of M.

Example 2.3.4

If $M \in SN$, then $BT(M)$ is the normal form of M.
$BT(\Delta\Delta) = \Omega$.
$BT((\lambda x.f(xx))((\lambda x.f(xx))) = \bigvee_{n \geq 0} f^n(\Omega)$.

The last example shows that Böhm trees can be infinite.

Proposition 2.3.5 *If $M \rightarrow N$, then $BT(M) = BT(N)$.*

PROOF HINT. Use the confluence property. $\square$

Lemma 2.3.6 *If M and N differ only by the replacement of some (disjoint) occurrences of subterms of the form $\lambda x_1 \cdots x_n.\Omega M_1 \cdots M_p$ by Ω or vice-versa, then $BT(M) = BT(N)$.*

PROOF HINT. If M, N are as in the statement, then $\omega(M) = \omega(N)$; moreover, if $M \rightarrow M'$, then either $\omega(M') = \omega(N)$ or $\omega(M') = \omega(N')$ for some N' such that $N \rightarrow N'$. $\square$

Let M be a λ-term, and F be a set of redex occurrences in M. Then F determines a context $C_{M,F}$ such that $M = C_{M,F}[\vec{R}]$, where $\vec{R}$ enumerates

$$\{M/u \mid u \in F \text{ and } (v < u \Rightarrow v \notin F)\}.$$

Lemma 2.3.7 *Let M, F be as above. Then $\omega(C_{M,F}[\vec{\Omega}]) = \omega(M)$.*

PROOF. By the definition of ω and by simple induction on the size of M. $\square$

We say that a derivation $M \rightarrow_{u_1} M_1 \cdots \rightarrow_{u_n} M_n$ does not touch a set F of redex occurrences in M if none of the u_i's is a residual of an occurrence in F. We write $M \xrightarrow{\neg F} {}^* M_n$.

Lemma 2.3.8 *If $D : M \xrightarrow{\neg F} {}^* N$, then $C_{M,F}[\vec{\Omega}] \rightarrow^* C_{N,F/D}[\vec{\Omega}]$.*

PROOF. The one step case implies the multistep case straightforwardly. Let thus $D = u$: There are two cases:

$$\begin{array}{ll} \exists\, u_1 \in F \;\; u_1 < u: & \text{Then } u_1/u = \{u_1\}, \text{ hence } C_{N,F/u}[\vec{\Omega}] = C_{M,F}[\vec{\Omega}]. \\ \forall\, u_1 \in F \;\; (u < u_1 \text{ or } u \not\uparrow u_1): & \text{Then } C_{M,F}[\vec{\Omega}] \to_u C_{N,F/u}[\vec{\Omega}]. \end{array}$$

□

When F is the set of all redexes in $\vec{M}$, we write $C[\vec{M}] \xrightarrow{\neg\vec{M}} {}^*N$ for $C[\vec{M}] \xrightarrow{\neg F} {}^*N$.

Lemma 2.3.9 (inside-out) *If $C[\vec{M}] \to^* P$, then there exist $\vec{N}$, Q such that*

$$\vec{M} \to^* \vec{N},\; C[\vec{N}] \xrightarrow{\neg\vec{N}} {}^*Q, \text{ and } P \to^* Q$$

where $\vec{M} \to^* \vec{N}$ has the obvious componentwise meaning.

PROOF. Once more, we use labelled reduction. Assume that $C[\vec{M}]$ is labelled. Let $\mathcal{P}$ consist of the degrees of the redexes reduced in $C[\vec{M}] \to^* P$. Let $\vec{N}$ be the $\beta^l_{\mathcal{P}}$-normal forms of $\vec{M}$. By $\beta^l_{\mathcal{P}}$-confluence, we have $P \to^* Q$ and $C[\vec{N}] \to^* Q$, for some Q. Let u be an occurrence of a β-redex in $\vec{N}$. Since the components of $\vec{N}$ are normal, the degree of u, which by lemma 2.2.14 is the degree of all its residuals, does not satisfy $\mathcal{P}$, hence u is not reduced in the derivation $C[\vec{N}] \to^* Q$. □

Informally, the lemma says that reductions can be first carried out "inside" (in the terms $\vec{M}$), and then "outside" only. In this outside phase, the actual nature of the redexes in $\vec{N}$ is irrelevant, as formalised by the next lemma.

Lemma 2.3.10 *If $D : C[\vec{N}] \xrightarrow{\neg\vec{N}} {}^*Q$, then $\omega(Q) \leq BT(C[\omega(\vec{N})])$.*

PROOF. Let F be the family of all the redex occurrences in $\vec{N}$. By lemma 2.3.8 we have $C_{C[\vec{N}],F}[\vec{\Omega}] \to^* C_{Q,F/D}[\vec{\Omega}]$. Hence $\omega(Q) = \omega(C_{Q,F/D}[\vec{\Omega}]) \leq BT(C_{C[\vec{N}],F}[\vec{\Omega}])$, by lemma 2.3.7 and by definition of Böhm trees. We are left to prove

$$BT(C_{C[\vec{N}],F}[\vec{\Omega}]) = BT(C[\omega(\vec{N})])$$

which follows from lemma 2.3.6. □

Finally, we can prove the main result of the section: the context operation is continuous. This result is due to Hyland and Wadsworth, independently [Wad76, Hyl76]. We follow the proof of Lévy [Lev78].

Theorem 2.3.11 (syntactic continuity) *For all contexts C, for any $\vec{M}$ and any $B \in \mathcal{N}$, the following implication holds:*

$$B \leq BT(C[\vec{M}]) \Rightarrow (\exists\, \vec{A} \in \mathcal{N}\;\; (\vec{A} \leq BT(\vec{M}) \text{ and } B \leq BT(C[\vec{A}])).$$

PROOF. If $B \leq BT(C[\vec{M}])$, then $C[\vec{M}] \rightarrow^* P$ for some P such that $B \leq \omega(P)$. By lemma 2.3.9, there exist $\vec{N}$, Q such that $\vec{M} \rightarrow^* \vec{N}$, $C[\vec{N}] \xrightarrow{\neg\vec{N}} {}^*Q$, and $P \rightarrow^* Q$. We have:

$$\begin{array}{ll} \omega(P) \leq \omega(Q) & \text{by lemma 2.3.2, and} \\ \omega(Q) \leq BT(C[\omega(\vec{N})]) & \text{by lemma 2.3.10 .} \end{array}$$

Take $\vec{A} = \omega(\vec{N})$. Then

$$\begin{array}{l} B \leq \omega(P) \leq \omega(Q) \leq BT(C[\vec{A}]), \\ \vec{A} \leq BT(\vec{M}), \text{ by definition of Böhm trees.} \end{array}$$ □

Thus, informally, the proof proceeds by organizing the reduction in an inside-out order, and by noticing that the partial information gathered about the Böhm trees of $\vec{M}$ during the inside phase is sufficient.

Exercise 2.3.12 ($\underline{C}$-cont) *Let M, N be Ω-terms such that $M \leq N$. Show that $BT(M) \leq BT(N)$. This allows us to define $\underline{C} : \mathcal{N}^\infty \rightarrow \mathcal{N}^\infty$, for any context C, by*

$$\underline{C}(A) = \bigvee\{BT(C[B]) \mid B \leq A \text{ and } B \text{ is finite}\}.$$

Show that $\underline{C}(BT(M)) = BT(C[M])$, for any M.

2.4 The Syntactic Sequentiality Theorem *

The context operation is not only continuous, but also sequential. The syntactic sequentiality theorem, due to Berry [Ber79], which we present in this section, motivates a semantic investigation of sequentiality, which is covered in section 6.5 and chapter 14. Two technical lemmas are needed before we can state the theorem.

Lemma 2.4.1 *If M is an Ω-term and $M \rightarrow^* M'$, then there exists a mapping † from the set $\{v_1, \ldots v_n\}$ of occurrences of Ω in M' to the set of occurrences of Ω in M such that $N \rightarrow^* M'[(N/v_1^\dagger)/v_1, \ldots, (N/v_n^\dagger)/v_n]$, for any $N > M$.*

In particular, if $M \rightarrow^* M'$ and $N > M$, then there exists $N' > M'$ such that $N \rightarrow^* N'$.

Lemma 2.4.2 *If the normal derivation sequence from M contains only terms of the form $M' = \lambda x_1 \cdots x_n.(\lambda x.P')M_1' \cdots M_k'$, then $BT(M) = \Omega$.*

PROOF. As in corollary 2.2.18. Suppose $BT(M) \neq \Omega$. Then there exists a derivation

$$M \rightarrow^* M'' = \lambda x_1 \cdots x_n.yM_1'' \cdots M_k''$$

and we can suppose that this derivation is standard, by theorem 2.2.17. But the shape of M'' forces this derivation to be actually normal. □

The following theorem asserts that the function BT is sequential. A general definition of sequential functions will be given in section 14.1

Theorem 2.4.3 (syntactic sequentiality) *The Böhm tree function satisfies the following property: for any Ω-term M and any u such that $BT(M)/u = \Omega$ and $BT(P)/u \neq \Omega$ for some $P > M$, there exists v, depending on M, u only, such that whenever $N > M$ and $BT(N)/u \neq \Omega$, then $N/v \neq \Omega$*

Proof. By induction on the size of u. Suppose $BT(M)/u = \Omega$, $M < N$, and $BT(N)/u \neq \Omega$. We distinguish two cases:

1. $BT(M) \neq \Omega$. Then

   $$\begin{array}{l} M \rightarrow^* M' = \lambda x_1 \cdots x_n.yM'_1 \cdots M'_k \text{ and} \\ BT(M) = BT(M') = \lambda x_1 \cdots x_n.yBT(M'_1) \cdots BT(M'_k) \ . \end{array}$$

   We observe that any $N' > M'$ has the form $N' = \lambda x_1 \cdots x_n.yN'_1 \cdots N'_k$, and also that $BT(N') = \lambda x_1 \cdots x_n.yBT(N'_1) \cdots BT(N'_k)$. Then u occurs in some $BT(M'_i)$, so that we can write $BT(M')/u = BT(M'_i)/u'$ $(= \Omega)$, for an appropriate proper suffix u' of u. On the other hand, let $N > M$ with $BT(N)/u \neq \Omega$, and let $N' > M'$ be such that $N \rightarrow^* N'$. Then $N'_i > M'_i$ and $BT(N'_i)/u' = BT(N')/u = BT(N)/u \neq \Omega$. We can thus apply induction to M'_i, u', and obtain an index v at M', u. It remains to bring this index back to M. This is done thanks to lemma 2.4.1: the index at M, u is $v^\dagger$, in the terminology of this lemma. Indeed, if $N > M$ and $N/v^\dagger = \Omega$, then by the lemma:

   $$\begin{array}{ll} N \rightarrow^* N' & \text{hence } BT(N) = BT(N') \\ \text{with } N' > N \text{ and } N'/v = \Omega & \text{hence } BT(N')/u = \Omega \ . \end{array}$$

   Putting this together, we have $BT(N)/u = \Omega$, which shows that $v^\dagger$ is a sequentiality index.

2. $BT(M) = \Omega$. Suppose that the leftmost reduction sequence from M contains only terms of the form $M' = \lambda x_1 \cdots x_n.(\lambda x.P')M'_1 \cdots M'_k$. Then the normal sequence from N is the sequence described by lemma 2.4.1, whose terms are of the same shape, which entails $BT(N) = \Omega$ by lemma 2.4.2. Hence the leftmost reduction sequence from M contains a term $M' = \lambda x_1 \cdots x_n.\Omega M'_1 \cdots M'_k$. The only chance of getting $BT(N') \neq \Omega$ for an $N' > M'$ is to increase M' in his head Ω, which is therefore a sequentiality index at M', u. As above, we use lemma 2.4.1 to bring this index back to M. □

Exercise 2.4.4 ($\underline{C}$-seq) *Show, as a corollary of theorem 2.4.3, that the function $\underline{C}$ defined in exercise 2.3.12 is sequential. (The reader can refer to definition 14.1.8, or guess a definition.)*

Chapter 3

D_∞ Models and Intersection Types

In this chapter we address the fundamental domain equation $D = D \to D$ which serves to define models of the λ-calculus. By "equation", we actually mean that we seek a D together with an order-isomorphism $D \cong D \to D$. Taking $D = \{\perp\}$ certainly yields a solution, since there is exactly one function $f : \{\perp\} \to \{\perp\}$. But we are interested in a non-trivial solution, that is a D of cardinality at least 2, so that not all λ-terms will be identified! Domain equations will be treated in generality in chapter 7.

In section 3.1 we construct Scott's D_∞ models as order-theoretic limit constructions. In section 3.2 we define λ-models, and we discuss some properties of the D_∞ models. In section 3.3, we present a class of λ-models based on the idea that the meaning of a term should be the collection of properties it satisfies in a suitable "logic". In section 3.4 we relate the constructions of sections 3.1 and 3.3, following [CDHL82]. Finally in section 3.5 we use intersection types as a tool for the syntactic theory of the λ-calculus [Kri91, RdR93].

3.1 D_∞ Models

In chapter 1, we have considered products and function spaces as constructions on cpo's. They actually extend to functors (and are categorical products and exponents, as will be shown in chapter 4). Here is the action of $\to$ on pairs of morphisms of **Dcpo**.

Definition 3.1.1 ($\to$ functor) *Let D, D', E, E' be dcpo's and $f : D' \to D$ and $g : E \to E'$ be continuous. Then $f \to g : (D \to E) \to (D' \to E')$ is defined by*

$$(f \to g)(h) = g \circ h \circ f.$$

Notice the "reversal" of the direction: f goes from D' to D, not from D to D'. This is called contravariance (cf. appendix B).

The association $D \mapsto D \to D$ is not functorial in **Cpo**, because it is both contravariant and covariant in D. But it becomes functorial in the category of cpo's and injection-projection pairs.

Definition 3.1.2 (injection-projection pair) *An injection-projection pair between two cpo's D and D' is a pair $(i : D \to D', j : D' \to D)$, written $(i, j) : D \to_{ip} D'$, such that*

$$j \circ i = id \quad \text{and} \quad i \circ j \le id$$

where $\le$ is the pointwise ordering, cf. proposition 1.4.4. If only $j \circ i = id$ holds, we say that (i, j) is a retraction pair and that D' is a retract of D. Injection-projection pairs are composed componentwise:

$$(i_1, j_1) \circ (i_2, j_2) = (i_1 \circ i_2, j_2 \circ j_1)$$

and the identity injection-projection pair is (id, id).

Proposition 3.1.3 *If $(i, j) : D \to_{ip} D'$ is an injection-projection pair, then i determines j. Moreover, if D is algebraic, then j is defined as follows:*

$$j(x') = \bigvee\{y \mid i(y) \le x'\}.$$

Proof. Suppose that (i, j') is another pair. Then observe:

$$j' = id \circ j' = j \circ i \circ j' \le j \circ id = j.$$

The second part of the statement follows from the fact that an injection-projection pair is a fortiori an adjunction, i.e., $i(d) \le x'$ iff $d \le j(x')$. Then $\bigvee\{d \mid i(d) \le x'\} = \bigvee\{d \mid d \le j(x')\} = j(x')$. □

Proposition 3.1.4 *1. For any injection-projection pair $(i, j) : D \to_{ip} D'$, i maps compact elements of D to compact elements of D'.*

2. If D, D' are algebraic dcpo's, a function $i : D \to D'$ is the injection part of an injection-projection pair (i, j) iff i restricted to $\mathcal{K}(D)$ is a monotonic injection into $\mathcal{K}(D')$ such that for any finite $M \subseteq \mathcal{K}(D)$, if $i(M)$ is bounded by d' in $\mathcal{K}(D')$, then M is bounded in $\mathcal{K}(D)$ by some d such that $i(d) \le d'$.

Proof. (1) If $i(d) \le \bigvee \Delta'$, then $d = j(i(d)) \le j(\bigvee \Delta') = \bigvee j(\Delta')$ implies $d \le j(\delta')$ for some $\delta' \in \Delta'$. Then $i(d) \le i(j(\delta')) \le \delta'$.

(2) Let (i, j) be an injection-projection pair. By (1), i restricted to $\mathcal{K}(D)$ is a monotonic injection into $\mathcal{K}(D')$. Suppose $i(M) \le d'$. Then $M = j(i(M)) \le j(d')$. Hence M is bounded in D, from which we deduce $M \le d$ for some $d \le j(d')$, by algebraicity and by finiteness of M. Then d fits since $i(d) \le i(j(d')) \le d'$. Conversely, let $i : \mathcal{K}(D) \to \mathcal{K}(D')$ be as in the statement. It extends to a

continuous function $i : D \to D'$: $i(x) = \bigvee\{i(d) \mid d \leq x\}$. Let $j : D' \to D$ be defined by

$$j(x') = \bigvee\{d \mid i(d) \leq x'\}.$$

We prove that j is well-defined. We have to check that $\{d \mid i(d) \leq x'\}$ is directed. If $i(d_1), i(d_2) \leq x'$, then by algebraicity $i(d_1), i(d_2) \leq d'$ for some $d' \leq x'$, and by the assumption $\{d_1, d_2\} \leq d$ with $i(d) \leq d'$. This d fits since a fortiori $i(d) \leq x'$. It is easy to check $j \circ i = id$ and $i \circ j \leq id$. □

Remark 3.1.5 *If moreover the lub's of bounded subsets exist (cf. exercise 1.4.11), then the statement (2) of proposition 3.1.4 simplifies: "if $i(M)$ is bounded by d' in $\mathcal{K}(D')$, then M is bounded in $\mathcal{K}(D)$ by some d in $\mathcal{K}(D)$ such that $i(d) \leq d'$" can be replaced by: "if $i(M)$ is bounded in $\mathcal{K}(D')$, then M is bounded in $\mathcal{K}(D)$ and $i(\bigvee M) = \bigvee i(M)$". Indeed, from $i(M) \leq d'$, we deduce as above $M = j(i(M)) \leq j(d')$, i.e., $\bigvee M \leq j(d')$, from which $i(\bigvee M) \leq i(j(d')) \leq d'$ follows.*

The characterisation of injection-projection pairs at the level of compact elements will be rediscussed in chapters 7 and 10.

Definition 3.1.6 (D_∞) *Let $(i, j) : D \to_{ip} D'$ be an injection-projection pair. We define $(i', j') = (i, j) \to (i, j) : (D \to D) \to_{ip} (D \to_{cont} D)$ by*

$$i'(f) = i \circ f \circ j \quad j'(f') = j \circ f' \circ i.$$

Given a cpo D, we define the standard injection-projection pair $(i_0, j_0) : D \to_{ip} (D \to D)$ by

$$i_0(x)(y) = x \quad j_0(f) = f(\bot).$$

The cpo D_∞ is defined as follows:

$$D_\infty = \{(x_0, \ldots, x_n, \ldots) \mid \forall n \;\; x_n \in D_n \text{ and } x_n = j_n(x_{n+1})\}$$

where $(x_0, \ldots, x_n, \ldots)$ is an infinite tuple, standing for a map from ω to $\bigcup_{n\in\omega} D_n$, and

$$D_0 = D \quad D_{n+1} = D_n \to D_n \quad (i_{n+1}, j_{n+1}) = (i_n, j_n) \to (i_n, j_n)$$

so that $(i_n, j_n) : D_n \to_{ip} D_{n+1}$ for all n. We write x_n for the nth component of $x \in D_\infty$. In the description of an element of D_∞ we can omit the first components, for example $(x_1, \ldots, x_n, \ldots)$ determines $(j_0(x_1), x_1, \ldots, x_n, \ldots)$.

Remark 3.1.7 *There may be other choices for the initial pair (i_0, j_0). For example the chosen (i_0, j_0) is just the instance $(i_\bot, j_\bot)$ of the family (i_d, j_d) $(d \in \mathcal{K}(D))$ defined by*

$$\begin{array}{rcl} i_d(e) & = & d \to e \quad \textit{(step function)} \\ j_d(f) & = & f(d)\,. \end{array}$$

Hence the construction of D_∞ is parameterised by D and (i_0, j_0).

The lub's in D_∞ are defined pointwise: $(\bigvee \Delta)_n = \bigvee\{x_n \mid x \in \Delta\}$ (the continuity of the j_n's guarantees that $\bigvee \Delta$ indeed belongs to D_∞).

Lemma 3.1.8 *The following define injection-projection pairs from D_n to D_∞:*

$$\begin{array}{rcl} i_{n\infty}(x) & = & (k_{n0}(x), \ldots, k_{nn}(x), \ldots, k_{nm}(x), \ldots) \\ j_{n\infty}(x) & = & x_n \end{array}$$

where $k_{nm} : D_n \to D_m$ is defined by

$$k_{nm} = \begin{cases} j_m \circ k_{n(m+1)} & (n > m) \\ id & (n = m) \\ i_{m-1} \circ k_{n(m-1)} & (n < m)\ . \end{cases}$$

We shall freely write x for $i_{n\infty}(x)$. Under this abuse of notation, we have

$$\begin{array}{lcllcl} x \in D_n & \Rightarrow & x_n = x & m \leq n & \Rightarrow & x_m \leq x_n \\ x \in D_n & \Rightarrow & i_n(x) = x & (x_n)_m & = & x_{\min(n,m)} \\ x \in D_{n+1} & \Rightarrow & j_n(x) \leq x & x & = & \bigvee_{n \geq 0} x_n\ . \end{array}$$

PROOF. We check only the second implication and the last equality.

- $x \in D_n \Rightarrow i_n(x) = x$: $i_n(x)$ stands for $i_{(n+1)\infty}(i_n(x))$, that is,

$$(k_{(n+1)0}(i_n(x)), \ldots, k_{(n+1)n}(i_n(x)), i_n(x), \ldots, k_{(n+1)m}(i_n(x)), \ldots)$$

  which is $(k_{n0}(x), \ldots, x, i_n(x), \ldots, k_{nm}(x), \ldots)$, that is, x.

- $x = \bigvee_{n \geq 0} x_n$: By the continuity of $j_{n\infty}$, we have

$$(\bigvee_{n \geq 0} x_n)_p = (\bigvee_{n \geq p} x_n)_p = \bigvee_{n \geq p} (x_n)_p = \bigvee_{n \geq p} x_p = x_p.$$

□

Remark 3.1.9 *As a consequence of $x = \bigvee_{n \geq 0} x_n$, a compact element of D_∞ must belong to D_n, for some n.*

Lemma 3.1.10 *The following properties hold:*

1. $\forall n \leq p, x \in D_{n+1}, y \in D_p \;\; x(y_n) = x_{p+1}(y)$,
2. $\forall n \leq p, x \in D_{p+1}, y \in D_n \;\; x_{n+1}(y) = x(y_p)_n$.

PROOF. We check only the case $p = n + 1$.
(1) By definition of i_{n+1}, we have $i_{n+1}(x) = i_n \circ x \circ j_n$. Hence, as claimed, we have $i_{n+1}(x)(y) = i_n(x(j_n(y))) = i_n(x(y_n)) = x(y_n)$. (2) $(x(i_n(y))_n = j_n(x(i_n(y))) = j_{n+1}(x)(y) = x_{n+1}(y)$. □

Definition 3.1.11 *We define $\bullet : D_\infty \times D_\infty \to D_\infty$ and $G : (D_\infty \to D_\infty) \to D_\infty$ by*

$$\begin{array}{rcl} x\bullet y & = & \bigvee_{n\geq 0} x_{n+1}(y_n) \\ G(f) & = & \bigvee_{n\geq 0} G_n(f) \end{array}$$

where $G_n(f) \in D_{n+1}$ is defined by $G_n(f)(y) = f(y)_n$.

It is straightforward to check that these mappings are continuous.

Lemma 3.1.12 *The following properties hold:*

1. *If $x \in D_{n+1}$, then $x\bullet y = x(y_n)$.*
2. *If $y \in D_n$, then $(x\bullet y)_n = x_{n+1}(y)$.*

Proof. (1) Using lemma 3.1.10, we have

$$x\bullet y = \bigvee_{i\geq n} x_{i+1}(y_i) = \bigvee_{i\geq n} x((y_i)_n) = x(y_n).$$

(2) By continuity of $j_{n\infty}$ and by lemma 3.1.10, we have

$$(x\bullet y)_n = \bigvee_{p\geq n} (x_{p+1}(y_p))_n = \bigvee_{p\geq n} x_{n+1}(y) = x_{n+1}(y).$$

□

Theorem 3.1.13 *Let $F(x)(y) = x\bullet y$. The maps F and G are inverse isomorphisms between D_∞ and $D_\infty \to D_\infty$.*

Proof. • $G \circ F = id$: Thanks to lemma 3.1.12, we have $G_n(F(x)) = x_{n+1}$. Hence $G(F(x)) = \bigvee_{n\geq 0} x_{n+1} = x$.

• $F \circ G = id$: We have to prove $G(f)\bullet x = f(x)$ for any $f : D_\infty \to D_\infty$ and $x \in D_\infty$. By continuity, we have $G(f)\bullet x = \bigvee_{n\geq 0} G_n(f)\bullet x$. Since $G_n(f)\bullet x = G_n(f)(x_n)$ by lemma 3.1.12, we have $G(f)\bullet x = \bigvee_{n\geq 0} f(x_n)_n$. On the other hand we have $f(x) = \bigvee_{n\geq 0} f(x_n)$ by continuity, hence $f(x) = \bigvee_{n\geq 0, p\geq n} f(x_n)_p$. Finally, observing that $f(x_n)_p \leq f(x_p)_p$, we have

$$G(f)\bullet x = \bigvee_{n\geq 0} f(x_n)_n = \bigvee_{n\geq 0, p\geq n} f(x_n)_p = f(x).$$

□

We have thus obtained a solution to the equation $D = D \to D$. The heuristics has been to imitate Kleene's fixpoint construction, and to build an infinite sequence $D_0 = D, \ldots, H^n(D_0), \ldots$, with $H(D) = D \to D$. In fact it can be formally shown that:

- D_∞ is in a suitable sense the least upper bound of the D_n's, and, as a consequence,
- D_∞ is in a suitable sense the least D' "above" D for which "$H(D') \leq D'$" holds, and that moreover "$H(D') \cong D'$" holds.

This is fairly general, and will be addressed in chapter 7 (see also exercises 3.1.14, 3.1.15 for an anticipation).

Exercise 3.1.14 *Show that, for any D' with a collection of $(\phi_{n\infty}, \psi_{n\infty}) : D_n \to_{ip} D'$ such that $(\forall n\ (\phi_{n\infty}, \psi_{n\infty}) = (\phi_{(n+1)\infty}, \psi_{(n+1)\infty}) \circ (i_n, j_n))$, there exists a unique pair $(\phi, \psi) : D_\infty \to_{ip} D'$ such that*

$$\forall n\ (\phi, \psi) \circ (i_{n\infty}, j_{n\infty}) = (\phi_{n\infty}, \psi_{n\infty}).$$

(Hint: define $\phi(x) = \bigvee \phi_n(x_n)$, $\psi(y)_n = \psi_{n\infty}(y)$.) Recover the definition of $(F, G) : D_\infty \to_{ip} HD_\infty$ by taking

$$\begin{array}{rcl} D' & = & HD_\infty \\ (\phi_{(n+1)\infty}, \psi_{(n+1)\infty}) & = & (\phi_{n\infty}, \psi_{n\infty}) \to (\phi_{n\infty}, \psi_{n\infty}) . \end{array}$$

Exercise 3.1.15 *Define an (i_0, j_0)-H-algebra as a pair of two injection-projection pairs $(\alpha, \beta) : D_0 \to_{ip} D'$ and $(\gamma, \delta) : HD' \to_{ip} D'$ such that*

$$(\gamma, \delta) \circ H(\alpha, \beta) \circ (i_0, j_0) = (\alpha, \beta) \quad \textit{where } H(\alpha, \beta) = (\alpha, \beta) \to (\alpha, \beta)$$

and define an (i_0, j_0)-H-algebra morphism from $((\alpha, \beta), (\gamma, \delta))$ to $((\alpha_1, \beta_1) : D_0 \to_{ip} D'_1, (\gamma_1, \delta_1) : HD'_1 \to_{ip} D'_1)$
as a morphism $(\mu, \nu) : D' \to_{ip} D'_1$ such that

$$(\mu, \nu) \circ (\alpha, \beta) = (\alpha_1, \beta_1) \textit{ and } (\mu, \nu) \circ (\gamma, \delta) = (\gamma_1, \delta_1) \circ H(\mu, \nu).$$

Show that $((i_{0\infty}, j_{0\infty}), (G, F))$ is an initial (i_0, j_0)-H-algebra, that is, it has a unique morphism into each (i_0, j_0)-H-algebra.

We end the section with a lemma which will be needed in the proof of lemma 3.4.7(3).

Lemma 3.1.16 *For any $f \in D_{n+1}$ we have $G(i_{n\infty} \circ f \circ j_{n\infty}) = f$.*

Proof. We have $G(i_{n\infty} \circ f \circ j_{n\infty}) = \bigvee_{p \geq n} G_p(i_{n\infty} \circ f \circ j_{n\infty})$. Let $y \in D_p$. From

$$G_p(i_{n\infty} \circ f \circ j_{n\infty})(y) = i_{n\infty}(f(j_{n\infty}(y)))_p$$

we get $G_p(i_{n\infty} \circ f \circ j_{n\infty})(y) = f(y_n)$ (with our abuse of notation), hence $G_p(i_{n\infty} \circ f \circ j_{n\infty}) = k_{(n+1)(p+1)}(f)$ by lemma 3.1.10, and the conclusion follows. □

3.2 Properties of D_∞ Models

We have not yet shown with precision why a solution to $D = D \to D$ gives a model of the λ-calculus. We shall first give a few definitions: applicative structure, prereflexive domain, functional λ-model, and reflexive domain. The D_∞ models are reflexive, as are many other models of the λ-calculus. A fully general (and more abstract) definition of model will be given in chapter 4. Then we discuss some specific properties of D_∞ models.

Definition 3.2.1 (pre-reflexive) *An* applicative structure $(X, \bullet)$ *is a set X equipped with a binary operation $\bullet$. The set of representable functions $X \to_{rep} X$ is the set of functions from X to X defined by*

$$X \to_{rep} X = \{f \in X \to X \mid \exists y \in X \;\; \forall x \in X \;\; f(x) = y \bullet x\}.$$

A pre-reflexive domain (D, F, G) is given by a set D, a set $[D \to D]$ of functions from D to D, and two functions $F : D \to [D \to D]$ and $G : [D \to D] \to D$ such that $F \circ G = id$. The uncurried form of F is written $\bullet$. Hence a pre-reflexive domain D is an applicative structure, and $D \to_{rep} D = F(D) = [D \to D]$. If moreover D is a partial order and (G, F) forms an injection-projection pair, then (D, F, G) is called coadditive.

Notice that the conjunction of $F \circ G \geq id$ and $G \circ F \leq id$ is an adjunction situation. In coadditive pre-reflexive domains, we thus have

$$G(f) \leq x \Leftrightarrow f \leq F(x)$$

which entails that:

- G preserves existing lub's ($f \leq F(G(f))$, $g \leq F(G(g))$ entail $f \vee g \leq F(G(f) \vee G(g))$).
- G maps compact elements to compact elements (cf. proposition 3.1.4).

The functions F and G can serve to interpret untyped λ-terms. As in universal algebra and in logic, the meaning of a term is given relative to a so-called environment ρ mapping variables to elements of the model. Thus the meaning of a term is to be written as an element $[\![M]\!]\rho$ of D, read as: "the meaning of M at ρ". This motivates the following definition.

Definition 3.2.2 (λ-model) *A functional λ-model (λ-model for short) is a pre-reflexive domain (D, F, G) such that the interpretation of λ-terms given by the equations of figure 3.1 is correctly defined. In these equations, the point is to make sure that $\lambda d.[\![M]\!]\rho[d/x] \in [D \to D]$.*

$$\begin{array}{rcl} [\![x]\!]\rho & = & \rho(x) \\ [\![MN]\!]\rho & = & F([\![M]\!]\rho)([\![N]\!]\rho) \\ [\![\lambda x.M]\!]\rho & = & G(\lambda d.[\![M]\!]\rho[d/x]) \end{array}$$

Figure 3.1: The semantic equations in a functional λ-model

Proposition 3.2.3 (soundness) *In a λ-model, the following holds:*

$$\textit{if } M \to_\beta N, \textit{ then } [\![M]\!]\rho = [\![N]\!]\rho \textit{ for any } \rho.$$

Proof hint. Since $F \circ G = id$, we have

$$\begin{array}{rcl} [\![(\lambda x.M)N]\!]\rho & = & F(G(\lambda d.[\![M]\!]\rho[d/x]))([\![N]\!]\rho) \\ & = & [\![M]\!]\rho[[\![N]\!]/x]\,. \end{array}$$

Then the conclusion follows from the following substitution property, which can be proved by induction on the size of M: $[\![M[N/x]]\!]\rho = [\![M]\!]\rho[[\![N]\!]\rho/x]$. □

We refer to chapter 4 for a more detailed treatment of the validation of β (and η).

Definition 3.2.4 (reflexive) *A pre-reflexive domain (D, F, G) is called reflexive if D is a cpo and $[D \to D] = D \to D$.*

Proposition 3.2.5 *A reflexive domain is a functional λ-model.*

Proof. One checks easily by induction on M that $\lambda \vec{d}.[\![M]\!]\rho[\vec{d}/\vec{x}]$ is continuous. In particular, $\lambda d.[\![M]\!]\rho[d/x] \in [D \to D]$. □

The D_∞ models are reflexive. The additional property $G \circ F = id$ which they satisfy amounts to the validation of the η-rule (see chapter 4).

Remark 3.2.6 *It should come as no surprise that the D_∞ models satisfy η as well as β, since for β we expect a retraction from $D_\infty \to D_\infty$ to D_∞, while the construction exploits retractions from D_n to $D_n \to D_n$ which are the other way around.*

We now come back to D_∞ and prove a result originally due to Park: the fixpoint combinator is interpreted by the least fixpoint operator in D_∞. The proof given here is inspired by recent work of Freyd, and Pitts, about minimal invariants, a notion which will be discussed in section 7.2.

Proposition 3.2.7 *Let $\delta_J : (D_\infty \to D_\infty) \to (D_\infty \to D_\infty)$ be the function defined by*

$$\delta_J(e) = G \circ (e \to id) \circ F$$

where $\to$ is used in the sense of definition 3.1.1. The function δ_J has the identity as unique fixpoint.[1]

PROOF. Let ϵ be a fixpoint of δ_J. Considering ϵ, id as elements of D_∞, we shall prove that, for any n, $\epsilon_n = id_n$, i.e., (for $n \geq 1$), $\epsilon_n = id : D_{n-1} \to D_{n-1}$. The general case ($n \geq 2$) is handled as follows, using lemma 3.1.12:

$$\begin{array}{rclcl} \epsilon_{n+2}(x)(y) & = & (\epsilon \to id)_{n+2}(x)(y) & = & ((\epsilon \to id)(x))_{n+1}(y) \\ & = & (x \circ \epsilon)_{n+1}(y) & = & (x \circ \epsilon)(y)_n \ . \end{array}$$

On the other hand:

$$(x \circ \epsilon_{n+1})(y) = x(\epsilon_{n+1}(y)) = x(\epsilon(y)_n) = (x \circ \epsilon)(y)_n.$$

Then we can use induction:

$$\epsilon_{n+2}(x)(y) = (x \circ \epsilon)(y)_n = (x \circ \epsilon_{n+1})(y) = (x \circ id)(y) = x(y).$$

Hence $\epsilon_{n+2}(x) = x$, and $\epsilon_{n+2} = id$. We leave the base cases $n = 0, 1$ to the reader (hint: establish first that, in D_∞, $\bot \bullet y = \bot$, and $x_0(y) = x_0 = x(\bot_0)$). □

Remark 3.2.8 *1. Proposition 3.2.7 does not depend on the initial choice of D_0, but the proof uses the fact that the initial pair (i_0, j_0) is the standard one (this is hidden in the hint).*

2. The functional δ_J may seem a bit ad hoc. A more natural functional would be δ defined by: $\delta(e) = G \circ (e \to e) \circ F$. More generally, for a domain equation of the form $D = H(D)$, with a solution given by inverse isomorphisms $F : D \to H(D)$ and $G : H(D) \to D$, we can set $\delta(e) = G \circ H(e) \circ F$. But replacing δ_J by δ in the proof of proposition 3.2.7 fails to work in the base cases $n = 0, 1$. On the other hand, we shall see (proposition 7.1.23) that the functional δ works well with the initial *solution of $D = H(D)$. (Remember that we are not interested in the trivial initial solution $\{\bot\}$ of $D = D \to D$.)*

A fortiori, the identity is the least fixpoint of δ_J. This fact can be used as a tool to prove properties of D_∞ by induction. We illustrate this with a simple proof (adapted from [Pit95]) of Park's result.

Proposition 3.2.9 (Park) *Let $Y = \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$. then $[\![Y]\!]$ in D_∞ is the least fixpoint operator (cf. proposition 1.1.7 and exercise 1.4.5).*

[1]The subscript J in δ_J comes from the term $J = Y(\lambda fxy.x(fy))$ (see the discussion on the relations between D_∞ and Böhm trees, later in this section).

PROOF. Let $f : D \to D$. Let $\Delta_f = [\![\lambda x.y(xx)]\!]\rho[f/y] = G(x \mapsto f(x{\bullet}x))$. We have to prove $\Delta_f{\bullet}\Delta_f \leq \mathit{fix}(f)$ (the other direction follows from soundness, since $YM =_\beta M(YM)$). Consider

$$E = \{e : D_\infty \to D_\infty \mid e \leq id \text{ and } e(\Delta_f){\bullet}\Delta_f \leq \mathit{fix}(f)\}.$$

By continuity, E is closed under directed lub's; also, obviously, $\perp \in E$. We have to show that $id \in E$. By proposition 3.2.7, it is enough to show that if $e \in E$, then $\delta_J(e) \in E$. We have:

$$\begin{array}{rcll}
\delta_J(e)(\Delta_f){\bullet}\Delta_f & = & G((e \to id)(x \mapsto f(x{\bullet}x))){\bullet}\Delta_f & \\
& = & (e \to id)(\lambda x.f(x{\bullet}x))(\Delta_f) & \\
& = & f(e(\Delta_f){\bullet}e(\Delta_f)) & \\
& \leq & f(e(\Delta_f){\bullet}\Delta_f) & \text{since } e \leq id \\
& \leq & f(\mathit{fix}(f)) & \text{since } e \in E \\
& = & \mathit{fix}(f)\ . &
\end{array}$$

□

D_∞ models and Böhm trees. The rest of the section is an overview of early independent work of Wadsworth and Hyland, relating D_∞ models and Böhm trees (cf. definition 2.3.1) [Wad76, Hyl76]. They proved that the following are equivalent for any two λ-terms M and N:

1. $M \leq_{op} N$, which means (cf. definition 2.1.6)

   $$\forall C \ \ (C[M] \text{ has a head normal form } \Rightarrow \ C[N] \text{ has a head normal form}).$$

2. $BT(M) \ {}^\eta{\leq}^\eta \ BT(N)$ (the meaning of ${}^\eta{\leq}^\eta$ is sketched below)

3. $[\![M]\!] \leq [\![N]\!]$ in D_∞ (for any choice of D_0, but with the standard initial (i_0, j_0))

The equivalence (1)⇔(3) is called a full-abstraction property (cf. section 6.4). We briefly indicate the techniques used to prove these equivalences.

(1) ⇒ (2) : This is the hard core of the theorem. It is proved by contradiction, by the so-called Böhm-out technique. Roughly, the technique consists in associating with a subterm P of a term M a context C with the property that $C[M] =_{\beta\eta} P$. In the proof by contradiction of (1)⇒(2) we use a context associated with an occurrence u where the two Böhm trees differ. If $BT(M)/u \neq \Omega$ and $BT(N)/u = \Omega$, the associated context witnesses rightaway $M \not\leq_{op} N$ (remember that for a partial λ-term P, by definition $\omega(P) = \Omega$ exactly when P is not a hnf). (As for the case where $BT(M)/u \neq \Omega$ and $BT(N) \neq \Omega$, see theorem 3.2.10.) The following example should suggest the contents of ${}^\eta{\leq}^\eta$. Consider (cf. proposition 3.2.7)

$$I = \lambda x.x \text{ and } J = Y(\lambda fxy.x(fy)).$$

It can be proved (as in proposition 3.2.9) that $[\![I]\!] = [\![J]\!]$ in D_∞. But the Böhm tree of I is just I, while the Böhm tree of J is infinite:

$$\lambda xz_0.x(\lambda z_1.z_0(\lambda z_2.z_1 \ldots \ .$$

These two Böhm trees get equalised through infinite η-expansion of I:

$$I = \lambda xz_0.xz_0 = \lambda xz_0x(\lambda z_1.z_0z_1) = \ldots \ .$$

$(2) \Rightarrow (3)$: This follows from the following approximation theorem:

$$[\![M]\!] = \bigvee\{[\![A]\!] \mid A \leq BT(M)\}.$$

$(3) \Rightarrow (1)$: A corollary of the approximation theorem is the adequacy theorem:

$$[\![M]\!] = \bot \Leftrightarrow M \text{ has no head normal form} \quad \Leftrightarrow BT(M) = \Omega).$$

Therefore $M \leq_{op} N$ can be rephrased as:

$$\forall C \ \ ([\![C[N]]\!] = \bot) \Rightarrow ([\![C[M]]\!] = \bot).$$

This holds, because we have $[\![C[M]]\!] \leq [\![C[N]]\!]$, by the compositionality of the interpretation in D_∞. The Böhm-out technique was first used in the proof of Böhm's theorem, which we now state.

Theorem 3.2.10 (Böhm) *Let M, N be λ-terms which have both a $\beta\eta$-normal form, and whose $\beta\eta$-normal forms are distinct. Then any equation $P = Q$ is derivable from the system obtained by adding the axiom $M = N$ to $\beta\eta$.*

PROOF HINT. Given fixed M, N as in the statement, the proof consists in associating with any pair of terms P, Q a context $C_{P,Q}$ such that $C[M] =_{\beta\eta} P$ and $C[N] =_{\beta\eta} Q$. The induction is based on the size of M, N. We only describe a few typical situations. First we can assume without loss of generality that M and N have no head λ's, since they can be brought to that form by contexts of the form $[\ ]x_1 \ldots x_n$. Notice that here η-interconvertibility is crucial, since in this process, say, $\lambda x.Nx$ and N are identified. We now briefly discuss examples of the differents cases that may arise:

1. Base case: $M = x\vec{M}$ and $N = y\vec{N}$ $(y \neq x)$. Then we take

$$C = (\lambda xy.[\ ])(\lambda\vec{u}.P)(\lambda\vec{v}.Q).$$

2. $M = xM_1$ and $N = xN_1N_2$. We turn this difference in the number of arguments into a base case difference, in two steps. First, a context $[\ ]y_1y_2$, with y_1, y_2 distinct, yields $xM_1y_1y_2$ and $xN_1N_2y_1y_2$. Second, we substitute the term $\alpha_2 = \lambda z_1z_2z.zz_1z_2$, called *applicator*, for x. Altogether, we set $D = (\lambda x.[\ ]y_1y_2)\alpha_2$, and we have

$$D[M] =_{\beta\eta} y_2M_1y_2 \quad \text{and} \quad D[N] =_{\beta\eta} y_1N_1N_2y_2$$

which is a base case situation.

3. $M = xM_1M_2$, $N = xN_1N_2$: Then $M \neq_{\beta\eta} N$ implies, say, $N_1 \neq_{\beta\eta} N_2$. It is enough to find D such that $D[M] =_{\beta\eta} M_2$ and $D[N] =_{\beta\eta} N_2$, because then one may conclude by an induction argument. In first approximation we are inclined to substitute the projection term $\pi_2 = \lambda z_1 z_2.z_2$ for x, yielding $M_2[\pi_2/x]$ and $N_2[\pi_2/x]$. But we do not want the substitution of π_2 in M_2 and N_2. To avoid this, we proceed in two steps: First we apply the two terms to a fresh variable z and substitute α_2 for x, then we substitute π_2 for z. Formally, we take $D = D_2[D_1]$, where $D_1 = (\lambda x.[\;]z)\alpha_2$, and $D_2 = (\lambda z.[\;])\pi_2$. Then

$$D[M] =_{\beta\eta} M_2[\alpha_2/x] \quad \text{and} \quad D[N] =_{\beta\eta} N_2[\alpha_2/x].$$

The substitution of α_2 turns out to be harmless. We make such substitutions by applicators into a parameter of the induction, together with P, Q, so that above we can have by induction a context $C_{P,Q,\alpha_2/x}$ with the property that

$$C_{P,Q,\alpha_2/x}[M_2[\alpha_2/x]] =_{\beta\eta} P \quad \text{and} \quad C_{P,Q,\alpha_2/x}[N_2[\alpha_2/x]] =_{\beta\eta} Q.$$

For full details on the proof, we refer to [Kri91]. □

In other words, adding $M = N$ leads to inconsistency. To prove the theorem, one may assume that M, N are distinct normal forms, and the place where they differ may be "extracted" like above into a context C such that $C[M] =_\beta \lambda xy.x$ and $C[N] =_\beta \lambda xy.y$, from which the theorem follows immediately. As a last remark, we observe that Böhm's theorem gives us already a limited form of "full abstraction".

Corollary 3.2.11 *Let M, N be λ-terms which have a $\beta\eta$-normal form. Then $[\![M]\!] = [\![N]\!]$ in D_∞ iff $M =_{\beta\eta} N$.*

PROOF. • $M =_{\beta\eta} N \Rightarrow [\![M]\!] = [\![N]\!]$: by soundness.

• $[\![M]\!] = [\![N]\!] \Rightarrow M =_{\beta\eta} N$: if M, N have distinct normal forms, then by Böhm's theorem and by soundness $[\![P]\!] = [\![Q]\!]$ for any P, Q; in particular $[\![x]\!] = [\![y]\!]$, which is contradicted by interpreting these terms with a ρ such that $\rho(x) \neq \rho(y)$ (D_∞ contains at least two elements). □

3.3 Filter Models

In this section we introduce the syntax of intersection types, which leads to the definition of a class of reflexive domains. Intersection types provide an extended system of types that allows to type all terms of the λ-calculus. Therefore the philosophy of these "types", which were originally called functional characters in

[CDC80], is quite different from the main stream of type theory, where the slogan is: "types ensure correction in the form of strong normalisation" (cf. theorem 2.2.9). Coppo and Dezani's characters, or types, are the simple types (built with $\rightarrow$ only, cf. definition 2.2.5), supplemented by two new constructions: binary intersection and a special type ω, with the following informal typing rules (see exercise 3.3.14):

any term has type ω,
a term has type $\sigma \wedge \tau$ iff it has both types σ and τ.

As an illustration, to type a self application of a variable x to itself, we can give to x the type $(\sigma \rightarrow \tau) \wedge \sigma$, and we get that xx has type τ. On the other hand, it can be shown that the application of $\Delta = \lambda x.xx$ to itself can only be typed by ω. In section 3.5, we shall further discuss the use of intersection types in the investigation of normalisation properties.

Turning to semantics, functional characters can be used to give meaning to terms, using the following philosophy: characters are seen as properties satisfied by terms, in particular, the property $\sigma \rightarrow \tau$ is the property which holds for a term M if and only if, whenever M is applied to a term N satisfying σ, MN satisfies τ. The meaning of a term is then the collection of properties which it satisfies.

Another way to understand the language of intersection types is to see them as a formal language for presenting the compact elements of the domain D which they serve to define. Recall that the topological presentation of domains (section 1.2) provides us with a collection of properties: the opens of the Scott topology, or more specifically the opens in the basis of Scott topology, that is, the sets of the form $\uparrow d$, where d is a compact of D. Hence, in this approach:

- Types σ, τ represent compact elements d, e of D, the association being bijective, but *antimonotonic* (observe that $d \leq e$ iff $\uparrow e \subseteq \uparrow d$).

- $\sigma \rightarrow \tau$ represents the step function (cf. lemma 1.4.8) $d \rightarrow e : D \rightarrow D$, which is a (compact) element of D, since it is intended that $D = D \rightarrow D$.

- $\sigma \wedge \tau$ represents $d \vee e$, and ω represents $\perp$ (intersection types give a lattice: all finite lub's exist).

These remarks should motivate the following definition.

Definition 3.3.1 (eats) *An extended abstract type structure (eats for short) S is given by a preorder $(S, \leq)$, called the carrier, whose elements are often called types, which:*

- *has all finite glb's, including the empty one, denoted by ω, and*

- *is equipped with a binary operation $\rightarrow$ satisfying:*

$$\begin{array}{ll}
(\rightarrow_1) & (\sigma \rightarrow \tau_1) \wedge (\sigma \rightarrow \tau_2) \leq \sigma \rightarrow (\tau_1 \wedge \tau_2) \\
(\rightarrow_2) & \dfrac{\sigma' \leq \sigma, \tau \leq \tau'}{(\sigma \rightarrow \tau) \leq (\sigma' \rightarrow \tau')} \\
(\omega) & \omega \leq \omega \rightarrow \omega\ .
\end{array}$$

Remark 3.3.2 *1. The structure of a preorder having all finite glb's can be axiomatised as follows:*

$$\begin{array}{lll}
\sigma \leq \sigma & & \dfrac{\sigma_1 \leq \sigma_2, \sigma_2 \leq \sigma_3}{\sigma_1 \leq \sigma_3} \\
\sigma \leq \omega & & \\
\sigma \wedge \tau \leq \sigma & \sigma \wedge \tau \leq \tau & \dfrac{\sigma \leq \sigma' \quad \tau \leq \tau'}{(\sigma \wedge \tau) \leq (\sigma' \wedge \tau')} \\
\sigma \leq \sigma \wedge \sigma & &
\end{array}$$

2. Inequation $(\rightarrow_2)$ expresses contravariance in the first argument and covariance in the second argument (cf. defintion 3.1.1).

3. Thanks to inequation $(\rightarrow_2)$, the two members of $(\rightarrow_1)$ are actually equivalent.

Lemma 3.3.3 *In any eats, the following inequality holds:*

$$(\sigma_1 \rightarrow \tau_1) \wedge (\sigma_2 \rightarrow \tau_2) \leq (\sigma_1 \wedge \sigma_2) \rightarrow (\tau_1 \wedge \tau_2).$$

PROOF. The statement is equivalent to

$$(\sigma_1 \rightarrow \tau_1) \wedge (\sigma_2 \rightarrow \tau_2) \leq ((\sigma_1 \wedge \sigma_2) \rightarrow \tau_1) \wedge ((\sigma_1 \wedge \sigma_2) \rightarrow \tau_2)$$

which holds a fortiori if $\sigma_1 \rightarrow \tau_1 \leq (\sigma_1 \wedge \sigma_2) \rightarrow \tau_1$ and $\sigma_2 \rightarrow \tau_2 \leq (\sigma_1 \wedge \sigma_2) \rightarrow \tau_2$. Both inequalities hold by $(\rightarrow_2)$. □

A way to obtain an eats is via a theory.

Definition 3.3.4 *Let T be the set of types constructed from a non-empty set At of atoms and from a signature $\{\omega^0, \wedge^2, \rightarrow^2\}$ (with the arities in superscript). The formulas have the form $\sigma \leq \tau$, for $\sigma, \tau \in T$. A theory consists of a set Th of formulas closed under the rules defining an eats. Thus a theory produces an eats with carrier T. We denote this eats also by Th. For any set Σ of formulas, $Th(\Sigma)$ denotes the smallest theory containing Σ. We denote with Th_0 the free theory $Th(\emptyset)$.*

Remark 3.3.5 *The assumption $At \neq \emptyset$ is important: otherwise. everything collapses, since $\omega \cong \omega \wedge \omega = \omega \rightarrow \omega$, where $\cong$ is the equivalence associated with the preorder $\leq$.*

Another way to obtain an eats is by means of an applicative structure.

Definition 3.3.6 (ets) *Let $(D, \bullet)$ be an applicative structure. Consider the following operation on subsets of D:*

$$A \to B = \{d \in D \mid \forall e \in A \;\; d \bullet e \in B\}.$$

A subset of $\mathcal{P}(D)$ is called an extended type structure (ets for short) if it is closed under finite set-theoretic intersections and under the operation $\to$ just defined.

Lemma 3.3.7 *An ets, ordered by inclusion, is an eats.*

We have already observed that the order between types is reversed with respect to the order in the abstract cpo semantics. Accordingly, the role of ideals is played here by filters.

Definition 3.3.8 (filter) *A filter of an inf-semi-lattice S is a nonempty subset x of S such that*

$$\begin{array}{l} \sigma, \tau \in x \Rightarrow \sigma \wedge \tau \in x \\ \sigma \in x \text{ and } \sigma \leq \tau \Rightarrow \tau \in x \,. \end{array}$$

The filter domain of an eats S is the set $\mathcal{F}(S)$ of filters of S, ordered by inclusion.

Remark 3.3.9 *Equivalently, in definition 3.3.8, the condition of non-emptyness can be replaced by: $\omega \in x$.*

The following properties are easy to check:

- For each $\sigma \in S$, $\uparrow \sigma$ is a filter.
- Given $A \subseteq S$, the filter $\overline{A}$ generated by A (i.e., the least filter containing A) is the intersection of all filters containing A. It is easily seen that

$$\overline{A} = \bigcup \{\uparrow (\tau_1 \wedge \cdots \wedge \tau_n) \mid \tau_1, \ldots, \tau_n \in A\}.$$

- In particular for a finite $A = \{\tau_1, \ldots, \tau_n\}$, we have $\overline{A} = \uparrow (\tau_1 \wedge \cdots \wedge \tau_n)$.

Proposition 3.3.10 *If S is an eats, then $\mathcal{F}(S)$ is an algebraic complete lattice.*

Proof. The minimum and maximum elements are $\uparrow \omega$ and S, respectively. The nonempty glb's are just set-theoretic intersections. The lub's are obtained by $\bigvee A = \overline{\bigcup A}$. Two instances of lub's can be given explicitly:

• If A is directed, then $\bigcup A$ is a filter, hence $\bigvee A = \bigcup A$. To see this, let $\sigma, \tau \in \bigcup A$. Then $\sigma \in x$, $\tau \in y$ for some $x, y \in A$. By directedness $\sigma, \tau \in z$ for some $z \in A$; since z is a filter, we have $\sigma \wedge \tau \in z$, hence $\sigma \wedge \tau \in \bigcup A$.

• $\uparrow \sigma \vee \uparrow \tau = \overline{\{\sigma, \tau\}} = \uparrow (\sigma \wedge \tau)$.
It follows that $\{\uparrow \sigma \mid \sigma \in x\}$ is directed, for any filter x, and since it is clear that $x = \bigcup \{\uparrow \sigma \mid \sigma \in x\}$, we obtain that $\mathcal{F}(S)$ is algebraic and that the finite elements are the principal filters $\uparrow \sigma$. □

Definition 3.3.11 *Let S be an eats, let $x, y \in \mathcal{F}(S)$, and f be a function $\mathcal{F}(S) \to \mathcal{F}(S)$. We define:*

$$\begin{array}{rcl} x{\bullet}y & = & \{\tau \mid \exists \sigma \in y \ \ \sigma \to \tau \in x\} \\ G(f) & = & \overline{\{\sigma \to \tau \mid \tau \in f(\uparrow \sigma)\}}\,. \end{array}$$

We write F for the curried form of $\bullet$.

Lemma 3.3.12 *For any $x, y \in \mathcal{F}(S)$, $x{\bullet}y$ is a filter, and the operation $\bullet$ is continuous.*

Proof. We check only that $x{\bullet}y$ is a filter:

- $\omega \in x{\bullet}y$: Take $\sigma = \omega$. Then $\omega \in x{\bullet}y$ since $\omega \leq \omega \to \omega$ implies $\omega \to \omega \in x$.
- Closure under intersections: Let $\tau_1, \tau_2 \in x{\bullet}y$, and let $\sigma_1, \sigma_2 \in y$ such that $\sigma_1 \to \tau_1, \sigma_2 \to \tau_2 \in x$. Then $\sigma_1 \wedge \sigma_2 \to \tau_1 \wedge \tau_2 \in x$, by lemma 3.3.3, hence $\tau_1 \wedge \tau_2 \in x{\bullet}y$.
- Upward closedness: by covariance. □

Remark 3.3.13 *Notice the role of the axiom $\omega \leq \omega \to \omega$, which guarantees the non-emptyness of $x{\bullet}y$.*

Exercise 3.3.14 *Consider the following interpretation of λ-terms:*

$$\begin{array}{rcl} [\![x]\!]\rho & = & \rho(x) \\ [\![MN]\!]\rho & = & ([\![M]\!]\rho){\bullet}([\![N]\!]\rho) \\ [\![\lambda x.M]\!]\rho & = & G(\lambda d.[\![M]\!]\rho[d/x]) \end{array}$$

where F and G are as in definition 3.3.11 and where ρ maps variables to filters. (Unlike in definition 3.2.2, we do not suppose $F \circ G = id$.) On the other hand, consider the formal typing system of figure 3.2, involving judgments of the form $\Gamma \vdash M : \sigma$, where M is an untyped λ-term, $\sigma \in S$, and Γ is a partial function from (a finite set of) variables to S, represented as a list of pairs of the form $x : \sigma$. (A slightly different one will be used in section 3.5.) Show $[\![M]\!]\rho = \{\sigma \mid \Gamma \vdash M : \sigma\}$, where $\rho(x) = \uparrow \sigma$ whenever $x : \sigma$ is in Γ.

We next examine how F and G compose.

Lemma 3.3.15 *In an eats, the following holds for any $\sigma, \tau \in S, x \in \mathcal{F}(S)$:*

$$\sigma \to \tau \in x \Leftrightarrow \tau \in x{\bullet} \uparrow \sigma.$$

Proof. ($\Rightarrow$) This follows obviously from $\sigma \in \uparrow \sigma$. Conversely, $\tau \in x{\bullet} \uparrow \sigma$ implies $\sigma' \to \tau \in x$ for some $\sigma' \geq \sigma$, hence $\sigma \to \tau \in x$ by contravariance. □

Lemma 3.3.16 *Let F, G be as in definition 3.3.11. Then $G \circ F \leq id$.*

$$\frac{x:\sigma\in\Gamma}{\Gamma\vdash x:\sigma}$$

$$\frac{\Gamma\vdash M:\sigma\to\tau\quad \Gamma\vdash N:\sigma}{\Gamma\vdash MN:\tau}\qquad \frac{\Gamma\cup\{x:\sigma\}\vdash M:\tau}{\Gamma\vdash\lambda x.M:\sigma\to\tau}$$

$$\frac{}{\Gamma\vdash M:\omega}\qquad \frac{\Gamma\vdash M:\sigma\quad \Gamma\vdash M:\tau}{\Gamma\vdash M:\sigma\wedge\tau}\qquad \frac{\Gamma\vdash M:\sigma\quad \sigma\le\tau}{\Gamma\vdash M:\tau}$$

Figure 3.2: Intersection type assignment

PROOF. Let $f = F(y) = \lambda x.(y\bullet x)$. If $\sigma\in G(f)$, then $\sigma\ge(\sigma_1\to\tau_1)\wedge\cdots\wedge(\sigma_n\to\tau_n)$, where $\tau_i\in y\bullet\uparrow\sigma_i$ $(1\le i\le n)$, or, equivalently, $\sigma_i\to\tau_i\in y$, from which $\sigma\in y$ follows, by definition of a filter. □

Proposition 3.3.17 *Let F, G be as in definition 3.3.11. We have*

$$\begin{array}{ll} F\circ G\ge id & \text{on } \mathcal{F}(S)\to\mathcal{F}(S)\\ F\circ G = id & \text{on } \mathcal{F}(S)\to_{rep}\mathcal{F}(S)\,. \end{array}$$

PROOF. • $F\circ G\ge id$: We have to prove $f(x)\subseteq G(f)\bullet x$, for every $f:\mathcal{F}(S)\to\mathcal{F}(S)$ and every filter x. If $\tau\in f(x)$, then by continuity $\tau\in f(\uparrow\sigma)$ for some $\sigma\in x$. Hence $\sigma\to\tau\in G(f)$ by definition of G, and $\tau\in G(f)\bullet x$ by definition of $\bullet$.

• $F\circ G\le id$: Since $\mathcal{F}(S)\to_{rep}\mathcal{F}(S) = F(\mathcal{F}(S))$, we can reformulate the statement as $F\circ G\circ F\le F$; it then follows from lemma 3.3.16. □

The situation so far is as follows. An eats gives rise to a coadditive prereflexive domain based on the representable functions. This domain does not necessarily give rise to a λ-model, because there may not exist enough representable functions to guarantee that $\lambda d.[\![M]\!]\rho[d/x]$ is always representable. The following proposition characterises the eats' which give rise to reflexive domains, with the above choice of F, G.

Proposition 3.3.18 *For any eats S, $\mathcal{F}(S)\to_{rep}\mathcal{F}(S) = \mathcal{F}(S)\to\mathcal{F}(S)$ (and hence $(\mathcal{F}(S), F, G)$ is reflexive) if and only if (for any types)*

$$(\mathcal{F}\mathit{refl})\quad (\sigma_1\to\tau_1)\wedge\cdots\wedge(\sigma_n\to\tau_n)\le\sigma\to\tau\Rightarrow\bigwedge\{\tau_i\mid\sigma\le\sigma_i\}\le\tau.$$

PROOF. We begin with two observations:

1. By proposition 3.3.17, any representable function f is represented by $G(f)$.

2. The compact continuous functions in $\mathcal{F}(S) \rightarrow_{cont} \mathcal{F}(S)$ are the functions of the form $f = (\uparrow \sigma_1 \rightarrow \uparrow \tau_1) \wedge \cdots \wedge (\uparrow \sigma_n \rightarrow \uparrow \tau_n)$, i.e., such that
$$f(x) = \uparrow \bigwedge \{\tau_i \mid \sigma_i \in x\}$$
(cf. remark 1.4.13). In particular, $\tau_i \in f(\uparrow \sigma_i)$ for every i, hence $\sigma_i \rightarrow \tau_i \in G(f)$.

Suppose first that all continuous functions are representable. Then the above f is represented by $G(f)$. Let σ, τ be as in the assumption of ($\mathcal{F}$*refl*). We have
$$\tau \in G(f) \bullet \uparrow \sigma = f(\uparrow \sigma) = \uparrow \bigwedge \{\tau_i \mid \sigma \leq \sigma_i\}$$
since $\sigma \rightarrow \tau \in G(f)$ and $F \circ G = id$. Hence $\bigwedge \{\tau_i \mid \sigma \leq \sigma_i\} \leq \tau$.

Conversely, suppose that ($\mathcal{F}$*refl*) holds. We show that the compact functions f (cf. observation (2)) are representable. We first show:
$$G(f) = \uparrow (\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n).$$

- $\uparrow (\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n) \subseteq G(f)$: this follows from $\sigma_i \rightarrow \tau_i \in G(f)$, shown above.

- $G(f) \subseteq \uparrow (\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n)$: it is enough to show that $\tau \in f(\uparrow \sigma)$, implies $(\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n) \leq \sigma \rightarrow \tau$. That is, the converse of ($\mathcal{F}$*refl*) holds. This is proved using lemma 3.3.3:
$$\begin{array}{rcl}(\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n) & \leq & \bigwedge_{\{i \mid \sigma \leq \sigma_i\}} (\sigma_i \rightarrow \tau_i) \\ & \leq & \bigwedge_{\{i \mid \sigma \leq \sigma_i\}} \sigma_i \rightarrow \bigwedge_{\{i \mid \sigma \leq \sigma_i\}} \tau_i \\ & \leq & \sigma \rightarrow \tau \; .\end{array}$$

Now we can prove that $G(f)$ represents f, that is, for all σ:
$$G(f) \bullet \uparrow \sigma = \uparrow \bigwedge \{\tau_i \mid \sigma \leq \sigma_i\}.$$

- $G(f) \bullet \uparrow \sigma \subseteq f(\uparrow \sigma)$: Let $\tau \in G(f) \bullet \uparrow \sigma$, that is, let $\sigma' \geq \sigma$ be such that $\sigma' \rightarrow \tau \in G(f)$. Then $(\sigma_1 \rightarrow \tau_1) \wedge \cdots \wedge (\sigma_n \rightarrow \tau_n) \leq \sigma' \rightarrow \tau$.. By ($\mathcal{F}$*refl*), we have $\bigwedge \{\tau_i \mid \sigma' \leq \sigma_i\} \leq \tau$. A fortiori $\bigwedge \{\tau_i \mid \sigma \leq \sigma_i\} \leq \tau$, that is, $\tau \in f(\uparrow \sigma)$.

- $f(\uparrow \sigma) \subseteq G(f) \bullet \uparrow \sigma$: This always holds, by proposition 3.3.17.

Finally, consider an arbitrary continuous function f, and let Δ be the set of its approximants. Then
$$G(\bigvee \Delta) \bullet x = (\bigvee G(\Delta)) \bullet x = \bigvee_{\delta \in \Delta} G(\delta) \bullet x = \bigvee_{\delta \in \Delta} \delta(x) = (\bigvee \Delta)(x).$$

□

Many eats' satisfy ($\mathcal{F}$*refl*), including Th_0 (next exercise), and the theories Th_V defined in section 3.4.

Exercise 3.3.19 *Show that Th_0 satisfies ($\mathcal{F}$refl). Hint: reformulate the goal for an arbitrary formula $\sigma \leq \tau$, exploiting the fact that an arbitrary formula can be written as an intersection of formulas which are each either ω, or an atom, or of the form $\sigma_1 \to \sigma_2$.*

3.4 Some D_∞ Models as Filter Models

We are now interested in recasting some D_∞ models in terms of "logic", that is in terms of filter models built on a theory. We restrict our attention to an initial D_0 which is an algebraic lattice.

Exercise 3.4.1 *Show that if D_0 is an algebraic lattice, then so is D_∞, for an arbitrary choice of (i_0, j_0).*

Definition 3.4.2 *Let (D, F, G) be a reflexive (coadditive) domain. Let $v : D \to D'$ and $w : D' \to D$ be inverse order-isomorphisms. Let F', G' be defined by*

$$\begin{aligned} F'(x') &= v \circ F(w(x')) \circ w \\ G'(f') &= v(G(w \circ f' \circ v)) . \end{aligned}$$

Then (D', F', G'), which is clearly a reflexive (coadditive) domain, is called isomorphic to (D, F, G).

In the following definition we "officialise" the inversion of order involved in the logical treatment.

Definition 3.4.3 *Let D be an algebraic lattice. We set*

$$K(D) = \{\uparrow d \mid d \in \mathcal{K}(D)\}$$

and order it by inclusion. (Hence, up to isomorphism, $K(D)$ is $(\mathcal{K}(D), \geq)$.)

Theorem 3.4.4 *If D is an algebraic lattice and if (D, F', G') is a reflexive coadditive domain, then $K(D)$ can be equipped with an eats structure in such a way that the associated $(\mathcal{F}(K(D)), F, G)$ is isomorphic to (D, F', G').*

Proof. By lemma 3.3.7 applied to the applicative structure D, it is enough to prove that $K(D)$ is closed under finite intersections and under the $\to$ operation. We have

$$\begin{aligned} &(\uparrow d) \wedge (\uparrow e) = \uparrow (d \vee e) \\ &\uparrow \bot = D. \end{aligned}$$

We show:

$$\uparrow d \to \uparrow e = \uparrow G'(d \to e)$$

where the left $\to$ denotes the operation introduced in definition 3.3.6, and where $d \to e$ is a step function. Indeed, we have

$$z \in \uparrow d \to \uparrow e \Leftrightarrow z \bullet d \geq e \Leftrightarrow F'(z) \geq d \to e \Leftrightarrow z \geq G'(d \to e)$$

where the last equivalence follows by coadditivity. Hence $K(D)$ forms an ets (notice that $G'(d \to e)$ is compact by coadditivity). The filters over $K(D)$ are in order-isomorphic correspondence with the ideals over $\mathcal{K}(D)$. We finally check that the operations F, G are the operations F', G', up to isomorphism. For $x, y \in D$, we have

$$\{\uparrow f \mid f \leq x\} \bullet \{\uparrow d \mid d \leq y\} = \{\uparrow e \mid \exists\, d \leq y \ \ (G'(d \to e) \leq x)\}.$$

So what we have to show is

$$e \leq x \bullet y \Leftrightarrow \exists\, d \leq y \ \ (G'(d \to e) \leq x)$$

which by coadditivity is equivalent to

$$e \leq F'(x)(y) \Leftrightarrow \exists\, d \leq y \ \ (d \to e \leq F'(x))$$

which follows by continuity of $F'(x)$. Matching G against G' amounts to show

$$\{\uparrow f \mid f \leq G'(g)\} = \overline{\{\uparrow d \to \uparrow e \mid e \leq g(d)\}}$$

which can be rephrased as

$$\{\uparrow f \mid f \leq G'(g)\} = \overline{A} \quad \text{where } A = \{\uparrow G'(d \to e) \mid d \to e \leq g\}.$$

- $\{\uparrow f \mid f \leq G'(g)\} \subseteq \overline{A}$: By the continuity of G', if $f \leq G'(g)$, then

$$f \leq G'(d_1 \to e_1 \vee \cdots \vee d_n \to e_n)$$

for some $d_1, e_1, \ldots, d_n, e_n$ such that $d_1 \to e_1 \leq g, \ldots, d_n \to e_n \leq g$. Hence

$$\uparrow G'(d_1 \to e_1), \ldots, \uparrow G'(d_n \to e_n) \in A.$$

Then, since G' preserves lub's,

$$\begin{array}{rcl} \uparrow G'(d_1 \to e_1 \vee \cdots \vee d_n \to e_n) & = & \uparrow (G'(d_1 \to e_1) \vee \cdots \vee G'(d_n \to e_n)) \\ & = & \uparrow G'(d_1 \to e_1) \cap \cdots \cap \uparrow G'(d_n \to e_n) \ \ \in \ \ \overline{A} \end{array}$$

from which we get $\uparrow f \in \overline{A}$.

- $\overline{A} \subseteq \{\uparrow f \mid f \leq G'(g)\}$: It is obvious that $A \subseteq \{\uparrow f \mid f \leq G'(g)\}$. □

Now we investigate under which conditions a filter domain can be presented by a theory (recall definition 3.3.4).

Definition 3.4.5 *Suppose that S is an eats, and that an interpretation $V : At \to S$ (which obviously extends to $V : T \to S$) is given. Then S, V induce a theory $S_V = \{\sigma \leq \tau \mid V(\sigma) \leq V(\tau)\}$.*

Lemma 3.4.6 *If S is an eats and $V : At \to S$ is such that its extension $V : T \to S$ is surjective, then S_V is isomorphic to S, in the sense that their collections of filters are isomorphic.*

Proof. The inverse maps are $x \mapsto V(x)$ and $y \mapsto V^{-1}(y)$. The surjectivity of V guarantees that $V(V^{-1}(y)) = y$. In the other direction, $x \subseteq V^{-1}(V(x))$ holds obviously. If $\sigma \in V^{-1}(V(x))$, then $V(\sigma) = V(\tau)$ for some $\tau \in x$, hence $\tau \leq \sigma \in S_V$, from which $\sigma \in x$ follows since x is a filter. □

Summarizing, to get a presentation of a domain by a theory, we should make sure that the domain is an algebraic lattice, is reflexive and coadditive, and we should find a surjection from types to the compact elements of the domain. We apply this discussion to the D_∞ models (D_∞, F, G) constructed from a lattice D_0.

Lemma 3.4.7 *If $V : At \to K(D_\infty)$ is such that for each $d \in K(D_0)$ there exist $\sigma \in T$ such that $V(\sigma) = \uparrow d$, then $V : T \to K(D_\infty)$ is surjective.*

Proof. Recall that by remark 3.1.9 a compact element of D_∞ is a compact element of D_n for some n. We use induction over n. The case $n = 0$ is the assumption. Take a compact element $c = (a_1 \to b_1) \vee \cdots \vee (a_n \to b_n)$ of D_{n+1}. We have by induction

$$\uparrow a_1 = V(\sigma_1), \ldots, \uparrow a_n = V(\sigma_n) \quad \text{and} \quad \uparrow b_1 = V(\tau_1), \ldots, \uparrow b_n = V(\tau_n).$$

Hence:

$$\begin{aligned} V(\sigma_i \to \tau_i) &= \uparrow a_i \to \uparrow b_i \\ &= \uparrow G'(a_i \to b_i) && \text{by theorem 3.4.4} \\ &= \uparrow (a_i \to b_i) && \text{by lemma 3.1.16, and} \\ & && \text{since } i_{n\infty} \circ (a_i \to b_i) \circ j_{n\infty} = a_i \to b_i\ . \end{aligned}$$

We conclude that $\uparrow c = V((\sigma_1 \to \tau_1) \wedge \cdots \wedge (\sigma_n \to \tau_n))$. □

We now define a theory for the D_∞ model based on $D_0 = \{\bot, \top\}$ and the standard pair (i_0, j_0) (cf. definition 3.1.6). We take $At = \{\kappa\}$, and define $V(\kappa) = \uparrow \top$. Then obviously V satisfies the assumption of lemma 3.4.7. So all we need now is a syntactic characterisation of Th_V as $Th(\Sigma)$ for some finite set Σ of axioms.

Theorem 3.4.8 *Let D_∞, V be as above. Set $\Sigma = \{\kappa \leq \omega \to \kappa, \omega \to \kappa \leq \kappa\}$. Then $Th_V = Th(\Sigma)$. Hence D_∞ is isomorphic to $\mathcal{F}(Th(\Sigma))$.*

PROOF. The second part of the statement follows from the first part:

D_∞	is isomorphic to	$\mathcal{F}(K(D_\infty))$	by theorem 3.4.4
$\mathcal{F}(K(D_\infty))$	is isomorphic to	$\mathcal{F}(Th_V)$	by lemma 3.4.6 .

We also deduce from these isomorphisms that Th_V satisfies ($\mathcal{F}$*refl*), by proposition 3.3.18. We now show the first part.

- $Th(\Sigma) \subseteq Th_V$: It suffices to check $V(\kappa) = V(\omega \to \kappa)$:

$$x \in V(\omega \to \kappa)\,(=\uparrow \bot \to \uparrow \top) \;\Leftrightarrow\; \forall y\;\, x{\bullet}y = \top \;\Leftrightarrow\; x = \top .$$

The latter equivalence follows from the fact that D_∞'s $\top$ element $(\top, \ldots, \top, \ldots)$ is D_0's $\top$, since $i_0(\top) = \lambda x.\top$ is D_1's $\top$ element, and since it is easily seen that $i(\top) = \top$ implies $i'(\top) = \top$ where $(i, j') = (i, j) \to (i, j)$.

- $Th_V \subseteq Th(\Sigma)$: We pick $(\sigma \leq \tau) \in Th_V$ and proceed by induction on the sum of the sizes of σ, τ. Clearly, it is enough to prove $\sigma' \leq \tau' \in Th(\Sigma)$ for some σ', τ' such that $\sigma \equiv \sigma', \tau \equiv \tau' \in Th(\Sigma)$ (in particular for $\sigma \equiv_0 \sigma'$, where $\equiv_0$ is the equivalence associated with the preorder $\leq_0$ of Th_0). Clearly, from any σ we can extract $\sigma' \equiv_0 \sigma$ such that σ' has a smaller size than σ and has one of the following forms:

$$\begin{array}{l} \omega \text{ or} \\ (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n)\ (n \geq 1) \text{ or} \\ (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n) \wedge \kappa\ (n \geq 0)\ . \end{array}$$

Similarly for τ. Exploiting this fact, the problem of verifying $(\sigma \leq \tau) \in Th(\Sigma)$ can be limited without loss of generality to $\tau = \kappa$ and $\tau = \tau_1 \to \tau_2$:

- $\tau = \kappa$: We consider the three possible forms of σ:
  - $\sigma = \omega$: this case is impossible, since $\omega \leq \kappa \notin Th_V$.
  - $\sigma = (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n)$: Then $\sigma \leq (\omega \to \kappa) \in Th_V$, since $(\sigma \leq \kappa) \in Th_V$. Let $I = \{i \mid \omega \leq \sigma_1^i \in Th_V\}$. By ($\mathcal{F}$*refl*), we have: $\bigwedge_{i \in I} \sigma_2^i \leq \kappa \in Th_V$. We can apply induction to both $\omega, \bigwedge_{i\in I} \sigma_1^i$ and $\bigwedge_{i\in I} \sigma_2^i, \kappa$, from which $\sigma \leq \tau' \in Th(\Sigma)$ follows by lemma 3.3.3, where $\tau' = \omega \to \kappa \equiv \kappa = \tau$.
  - $\sigma = (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n) \wedge \kappa$:
    Then obviously $\sigma \leq_0 \tau$.
- $\tau = \tau_1 \to \tau_2$: We consider the three possible forms of σ:
  - $\sigma = \omega$: Replacing ω by $\omega \to \omega$, we get $\omega \leq \tau_2$ by ($\mathcal{F}$*refl*), and $\omega \leq \tau_2 \in Th(\Sigma)$ by induction applied to ω, τ_2. Hence $\sigma' \leq \tau \in Th(\Sigma)$, with $\sigma' = \omega \to \omega \equiv \sigma$.

- $\sigma = (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n)$: The reasoning is the same as for the corresponding case for $\tau = \kappa$. We define now $I = \{i \mid \tau_1 \leq \sigma_1^i \in \mathit{Th}_V\}$, and we apply induction to $\tau_1, \bigwedge_{i \in I} \sigma_1^i$ and $\bigwedge_{i \in I} \sigma_2^i, \tau_2$.
- $\sigma = (\sigma_1^1 \to \sigma_2^1) \wedge \cdots \wedge (\sigma_1^n \to \sigma_2^n) \wedge \kappa$:
  The reasoning is a variation of the previous case. We replace κ by $\omega \to \kappa$, we keep the same I, and we apply induction to $\tau_1, \bigwedge_{i \in I} \sigma_1^i$ and $\bigwedge_{i \in I} \sigma_2^i \wedge \kappa, \tau_2$. □

There are more liberal conditions allowing to get $\mathit{Th} = \mathit{Th}(\Sigma)$ for some finite Σ.

Exercise 3.4.9 *Let Th be a theory satisfying ($\mathcal{F}$refl), and in which every atom κ is equivalent to a finite conjunction σ_κ of types of the form $\omega \to \kappa$ or $\kappa_1 \to \kappa_2$. Show that $\mathit{Th} = \mathit{Th}(\Sigma)$, where*

$$\Sigma = (\mathit{Th} \cap \{\kappa_1 \wedge \cdots \wedge \kappa_m \leq \kappa \mid \kappa_1, \ldots, \kappa_m, \kappa \in \mathit{At}\}) \cup \{\kappa \equiv \sigma_\kappa \mid \kappa \in \mathit{At}\}.$$

Hints: (1) Reason by induction on the number of occurrences of $\to$. (2) The inequations $\kappa_1 \wedge \cdots \wedge \kappa_m \leq \kappa$ might lead to loops when replacing $\kappa_1, \ldots, \kappa_m, \kappa$ by $\sigma_{\kappa_1}, \ldots, \sigma_{\kappa_n}, \sigma_\kappa$, hence they are added explicitly.

Exercise 3.4.10 (Park's D_∞) *Apply exercise 3.4.9 to show that the D_∞ model based on $D_0 = \{\bot, \top\}$ and $(i_\top, j_\top)$ (cf. remark 3.1.7) is isomorphic to $\mathcal{F}(\mathit{Th}(\Sigma^{Park}))$, with $\Sigma^{Park} = \{\kappa \equiv \kappa \to \kappa\}$. Hint: use the same function V, but notice that, unlike in the standard D_∞ model, it is not the case here that D_0's $\top$ is D_∞'s $\top$, since $i_\top(\bot) = \lambda x.x$ is not D_1's $\top$ element.*

3.5 More on Intersection Types *

In this section, following original work of Coppo and Dezani, we study intersection types from a syntactic point of view (and without considering an explicit preordering on types, as we did in order to construct a model). Intersection types are used to give characterisations of the following predicates over λ-terms: "has a head normal form", "has a normal form", and "is strongly normalisable". The first two characterisations can be derived as corollaries of a logical formulation of the approximation theorem (cf. section 3.2). Our presentation follows [Kri91].

Systems $\mathcal{D}\Omega$ and $\mathcal{D}$. Recall the set T of intersection types from definition 3.3.4. The typing system $\mathcal{D}\Omega$ is defined in figure 3.3. The only difference with the system presented in figure 3.2 is the replacement of the last rule by the more restricted rules $(\wedge E1)$ and $(\wedge E2)$. Another difference is that we let now M range over Ω-terms (cf. definition 2.3.1). The rule (ω) allows to type Ω.

The restriction of $D\Omega$ obtained by removing ω in the BNF syntax of T, as well as the rule (ω), is called $\mathcal{D}$. In $\mathcal{D}$ only λ-terms, i.e., terms without occurrences of Ω, can be typed.

$$\frac{x:\sigma\in\Gamma}{\Gamma\vdash x:\sigma}$$

$$\frac{\Gamma\vdash M:\sigma\to\tau \quad \Gamma\vdash N:\sigma}{\Gamma\vdash MN:\tau} \qquad \frac{\Gamma\cup\{x:\sigma\}\vdash M:\tau}{\Gamma\vdash \lambda x.M:\sigma\to\tau}$$

$$(\omega)\quad \frac{}{\Gamma\vdash M:\omega} \qquad (\wedge I)\quad \frac{\Gamma\vdash M:\sigma \quad \Gamma\vdash M:\tau}{\Gamma\vdash M:\sigma\wedge\tau}$$

$$(\wedge E1)\quad \frac{\Gamma\vdash M:\sigma\wedge\tau}{\Gamma\vdash M:\sigma} \qquad (\wedge E2)\quad \frac{\Gamma\vdash M:\sigma\wedge\tau}{\Gamma\vdash M:\sigma}$$

Figure 3.3: System $\mathcal{D}\Omega$

Remark 3.5.1 *Because of axiom (ω), in a provable judgement $\Gamma\vdash M:\sigma$ of $\mathcal{D}\Omega$, all free variables of M are not always declared in Γ. This property holds however in $\mathcal{D}$.*

We state a number of syntactic lemmas.

Lemma 3.5.2 (weakening) *If $\Gamma\vdash M:\sigma$ and $\Gamma\subseteq\Gamma'$ (that is, if $x:\sigma$ is in Γ then it is in Γ'), then $\Gamma'\vdash M:\sigma$.*

Lemma 3.5.3 *If $\Gamma, x_1:\sigma_1,\ldots,x_k:\sigma_k:\sigma_k\vdash M:\sigma$, if $N_1,\ldots,N_k$ are λ-terms, and if $\Gamma\vdash N_i:\sigma_i$ for all i's such that $x_i\in FV(M)$, then $\Gamma\vdash M[N_1/x_1,\ldots,N_k/x_k]:\sigma$.*

Proof hint. By induction on the length of the proof of $\Gamma, x_1:\sigma_1,\ldots,x_k:\sigma_k\vdash M:\sigma$. □

Remark 3.5.4 *Lemma 3.5.3 encompasses both substitution and* strengthening. *Substitution corresponds to the situation where $\Gamma\vdash N_i:\sigma_i$ for all $i\leq k$. Strengthening corresponds to the situation where $x_1,\ldots,x_k\notin FV(M)$: then $\Gamma, x_1:\sigma_1,\ldots,x_k:\sigma_k\vdash M:\sigma$ implies $\Gamma\vdash M:\sigma$.*

Lemma 3.5.5 *If $\Gamma, x:\sigma\vdash M:\tau$, then $\Gamma, x:\sigma\wedge\sigma'\vdash M:\tau$ for any σ'.*

Proof. By induction on the length of the proof of $\Gamma, x:\sigma\vdash M:\tau$. We only look at the base cases:

$\Gamma, x:\sigma\vdash x:\sigma$: Then $\Gamma, x:\sigma\wedge\sigma'\vdash x:\sigma\wedge\sigma'$, and $\Gamma, x:\sigma\wedge\sigma'\vdash x:\sigma$ follows by $(\wedge E)$.

$\Gamma, x:\sigma\vdash y:\tau$ $(y\neq x)$: Then also $\Gamma, x:\sigma\wedge\sigma'\vdash y:\tau$. □

Lemma 3.5.6 *If $\Gamma \vdash M : \sigma$ and $\Gamma' \vdash M : \sigma$, then $\Gamma \uplus \Gamma' \vdash M : \sigma$, where the variables declared in $\Gamma \uplus \Gamma'$ are those declared in Γ or Γ', and where (viewing the environments as functions)*

$$\Gamma \uplus \Gamma'(x) = \begin{cases} \tau \wedge \tau' & \textit{if } \Gamma(x) = \tau \textit{ and } \Gamma'(x) = \tau' \\ \tau & \textit{if } \Gamma(x) = \tau \textit{ and } \Gamma'(x) \textit{ is undefined} \\ \tau' & \textit{if } \Gamma'(x) = \tau' \textit{ and } \Gamma(x) \textit{ is undefined}. \end{cases}$$

PROOF. The statement follows from lemma 3.5.2 and from a repeated application of lemma 3.5.5. □

Definition 3.5.7 (prime type) *An intersection type is called prime if it is either an atomic type κ or an arrow type $\sigma \to \tau$. Every type is thus a conjunction of prime types (called its prime factors) and of some ω's, and, by $(\wedge E)$, if $\Gamma \vdash M : \sigma$ and if σ' is a prime factor of σ, then $\Gamma \vdash M : \sigma'$.*

Lemma 3.5.8 *Let $\Gamma \vdash M : \sigma$, with σ prime. Then:*

1. *If $M = x$, then $(x : \sigma') \in \Gamma$, where σ is a prime factor of σ',*
2. *If $M = \lambda x.N$, then $\sigma = \sigma_1 \to \sigma_2$ and $\Gamma, x : \sigma_1 \vdash N : \sigma_2$,*
3. *If $M = M_1M_2$, then $\Gamma \vdash M_2 : \tau$ and $\Gamma \vdash M_1 : \tau \to \sigma'$, for some τ, σ', such that σ is a prime factor of σ'.*

PROOF. First we claim that a proof of $\Gamma \vdash M : \sigma$ contains a proof $\Gamma \vdash M : \sigma'$ which does not end with a $(\wedge I)$ nor $(\wedge E)$ rule and is such that σ is a prime factor of σ'. To prove the claim, we generalise the assumption "a proof of $\Gamma \vdash M : \sigma$" to: "a proof of $\Gamma \vdash M : \sigma''$ where σ is a prime factor of σ''". We proceed by induction on the length of the proof of $\Gamma \vdash M : \sigma''$ and consider the last rule used:

$(\wedge I)$ Then $\sigma'' = \sigma_1 \wedge \sigma_2$, and σ, being a prime factor of σ'', is a prime factor of σ_1 or σ_2, thus we can apply induction to the left or right premise of the $(\wedge I)$ rule.

$(\wedge E)$ Then the premise of the rule is of the form $\Gamma \vdash M : \sigma'' \wedge \tau$ or $\Gamma \vdash M : \tau \wedge \sigma''$, and σ, being a prime factor of σ'', is also prime factor of $\sigma'' \wedge \tau$ or $\tau \wedge \sigma''$.

The claim is proved. Let $\Gamma \vdash M : \sigma'$ be as in the claim; it is a conclusion of one of the three rules of the simply typed λ-calculus (without intersection types):

$M = x$:	Then $(x : \sigma') \in \Gamma$.
$M = \lambda x.N$:	Then $\sigma' = \sigma_1 \to \sigma_2$, hence σ' is prime, which entails $\sigma = \sigma'$.
$M = M_1M_2$:	Obvious.

□

Proposition 3.5.9 (subject reduction) *If $\Gamma \vdash M : \sigma$ and $M \to_\beta M'$, then $\Gamma \vdash M' : \sigma$.*

PROOF HINT. In the crucial axiom case, use lemmas 3.5.8 (case 2) and 3.5.3. □

Lemma 3.5.10 (expansion) *1. In $\mathcal{D}\Omega$, if $\Gamma \vdash M[N/x] : \tau$, then $\Gamma \vdash (\lambda x.M)N : \tau$.*

2. In $\mathcal{D}$, if $\Gamma \vdash M[N/x] : \tau$ and if $\Gamma \vdash N : \sigma$ for some σ, then $\Gamma \vdash (\lambda x.M)N : \tau$.

Proof. We prove (1), indicating where the additional assumption comes in for (2). We may assume by lemma 3.5.3 that x is not declared in Γ. The statement follows obviously from the following claim:

$$\exists \sigma \;\; (\Gamma \vdash N : \sigma \text{ and } \Gamma, x : \sigma \vdash M : \tau).$$

The claim is proved by induction on $(size(M), size(\tau))$:

- $\tau = \omega$: Obvious, taking $\sigma = \omega$.
- $\tau = \tau_1 \wedge \tau_2$: By $(\wedge E)$ and by induction, we have

  $$\begin{array}{ll} \Gamma \vdash N : \sigma_1 & \Gamma, x : \sigma_1 \vdash M : \tau_1 \\ \Gamma \vdash N : \sigma_2 & \Gamma, x : \sigma_2 \vdash M : \tau_2 \,. \end{array}$$

  We set $\sigma = \sigma_1 \wedge \sigma_2$, and we conclude by $(\wedge I)$ and lemma 3.5.5.
- τ prime:
  - $M = x$: Then the assumption is $\Gamma \vdash N : \tau$. Take $\sigma = \tau$.
  - $M = y \neq x$: Then the assumption is $\Gamma \vdash y : \tau$. Take $\sigma = \omega$ (in $\mathcal{D}$, take the assumed type of N).
  - $M = \lambda y.P$: By lemma 3.5.8 (2) we have $\tau = \tau_1 \to \tau_2$ and

    $$\Gamma, y : \tau_1 \vdash P[N/x] : \tau_2.$$

    By induction we have

    $$\Gamma, y : \tau_1 \vdash N : \sigma \quad \text{and} \quad \Gamma, x : \sigma, y : \tau_1 \vdash P : \tau_2.$$

    Since we can (must) make sure that y is not free in N, the conclusion follows from lemma 3.5.3.
  - $M = M_1 M_2$: We can apply induction since $size(M_1), size(M_2) \leq size(M)$. Using lemma 3.5.8 (3), we have

    $$\begin{array}{ll} \Gamma \vdash N : \sigma_1 & \Gamma, x : \sigma_1 \vdash M_1 : \tau'' \to \tau' \\ \Gamma \vdash N : \sigma_2 & \Gamma, x : \sigma_2 \vdash M_2 : \tau'' \end{array}$$

    with τ prime factor of τ'. As above, we set $\sigma = \sigma_1 \wedge \sigma_2$. □

Remark 3.5.11 *Unlike subject reduction, which holds widely in type systems, lemma 3.5.10 is peculiar of intersection types.*

Theorem 3.5.12 (subject equality) *If $\Gamma \vdash M : \sigma$ and $M =_\beta M'$, then $\Gamma \vdash M' : \sigma$.*

PROOF. One direction is proposition 3.5.9 and the other is easily proved by induction using lemmas 3.5.10 and 3.5.8. □

We shall prove the strong normalisation theorem and the approximation theorem making use of an elegant technique called the computability method. The method will be used again in sections 6.3 and 11.5.

Definition 3.5.13 ($\mathcal{N}$-saturated) *Let $\mathcal{N} \subseteq \Lambda$. A subset $\mathcal{X} \subseteq \Lambda$ is called $\mathcal{N}$-saturated if*

$$\forall N \in \mathcal{N}\ \forall M, N_1, \ldots, N_n \in \Lambda \quad (M[N/x]N_1 \ldots N_n \in \mathcal{X} \Rightarrow (\lambda x.M)NN_1 \ldots N_n \in \mathcal{X})$$

(in this implication n is arbitrary, in particular it can be 0).

Proposition 3.5.14 *The $\mathcal{N}$-saturated sets form an ets. (cf. definition 3.3.6).*

PROOF HINT. The statement is obvious for intersections. As for function types, the N_i's in the definition of saturated set serve precisely that purpose. □

Lemma 3.5.15 *For any interpretation V (cf. definition 3.4.5) by $\mathcal{N}$-saturated sets such that $\forall \sigma\ V(\sigma) \subseteq \mathcal{N}$, for any provable $x_1 : \sigma_1, \ldots, x_k : \sigma_k \vdash M : \sigma$, and for any $N_1 \in V(\sigma_1), \ldots, N_k \in V(\sigma_k)$, we have $M[N_1/x_1, \ldots, N_k/x_k] \in V(\sigma)$.*

PROOF. By induction on the length of the proof of $x_1 : \sigma_1, \ldots, x_k : \sigma_k \vdash M : \sigma$.

- $\Gamma \vdash x : \sigma$: The conclusion is one of the assumptions.
- Application: By induction we have

  $$M[N_1/x_1, \ldots, N_k/x_k] \in V(\sigma \to \tau) \quad \text{and} \quad N[N_1/x_1, \ldots, N_k/x_k] \in V(\sigma)$$

  hence $(MN)[N_1/x_1, \ldots, N_k/x_k] \in V(\tau)$ by definition of $V(\sigma \to \tau)$.
- Abstraction: We have to prove $(\lambda x.M[N_1/x_1, \ldots, N_k/x_k])N \in V(\tau)$, for any $N \in V(\sigma)$. By induction we have $M[N_1/x_1, \ldots, N_k/x_k][N/x] \in V(\tau)$, and the concusion follows by the definition of $\mathcal{N}$-saturated set, noticing that a fortiori $N \in \mathcal{N}$ by assumption.
- (ω): Obvious, since $V(\omega) = \Lambda$.
- $(\wedge I)$: Obvious by induction, since $V(\sigma \wedge \tau) = V(\sigma) \wedge V(\tau)$.
- $(\wedge E1)$ (or $(\wedge E2)$): Follows from the implicit order: $V(\sigma \wedge \tau) \subseteq V(\sigma)$. □

Remark 3.5.16 *Notice that we have used induction on types to construct $V(\sigma)$ at all types, and that we have used induction on (typing proofs of) terms to prove the statement. The core of the computability method indeed resides in this elegant separation of inductions, to be contrasted with their combinatorial combination in the proof of theorem 2.2.9.*

The following characterisation of strongly normalisable terms, provides in particular an alternative proof of strong normalisation for the simply typed λ-calculus (theorem 2.2.9).

Theorem 3.5.17 (strong normalisation – intersection types) *Any λ-term M is strongly normalisable iff $\Gamma \vdash M : \sigma$ is provable in $\mathcal{D}$ for some Γ, σ.*

PROOF. ($\Leftarrow$) We take $\mathcal{N} = SN$, the set of strongly normalisable terms, and we interpret the atomic types κ by setting $V(\kappa) = \mathcal{N}$. Our proof plan goes as follows. We show, for all σ:

1. $V(\sigma)$ is $\mathcal{N}$-saturated,
2. $x \in V(\sigma)$ for all variables x,
3. $V(\sigma) \subseteq \mathcal{N}$.

By these conditions lemma 3.5.15 can be applied, with $N_1 = x_1, \ldots, N_k = x_k$, yielding $M \in \mathcal{N}$. Therefore all we have to do is to prove the three conditions.

(1) By lemma 3.5.14, the condition boils down to the verification that $\mathcal{N}$ is $\mathcal{N}$-saturated. We proceed by induction on $depth(N) + depth(M[N/x]N_1 \ldots N_n)$ (cf. definition 2.2.1). It is enough to prove that all the one step reducts P of $(\lambda x.M)NN_1 \ldots N_n$ are in $\mathcal{N}$:

- $P = M[N/x]N_1 \ldots N_n$: By assumption.
- $P = (\lambda x.M')NN_1 \ldots N_n$: By induction, since
$$depth(M'[N/x]N_1 \ldots N_n) < depth(M[N/x]N_1 \ldots N_n).$$
- If the reduction takes place in one of the N_i's, the reasoning is similar.
- $P = (\lambda x.M)N'N_1 \ldots N_n$: By induction, since $depth(N') < depth(N)$ (notice that if $x \notin FV(M)$, then the depth of $M[N/x]N_1 \ldots N_n$ does not change; whence the notion of $\mathcal{N}$-saturated).

(2) and (3) We actually strengthen (2) into

2′. $\mathcal{N}_0 \subseteq V(\sigma)$.

where $\mathcal{N}_0 = \{xM_1 \ldots M_p \mid p \geq 0 \text{ and } \forall\, i \leq p\ \ M_i \in \mathcal{N}\}$. We shall prove (2′) and (3) together, as a consequence of the following properties, which we shall establish first:

$$\begin{array}{ll} \text{(A)} & \mathcal{N}_0 \subseteq \mathcal{N}, \\ \text{(B)} & \mathcal{N}_0 \subseteq (\mathcal{N} \to \mathcal{N}_0), \\ \text{(C)} & (\mathcal{N}_0 \to \mathcal{N}) \subseteq \mathcal{N}. \end{array}$$

(A) Any reduct of $xM_1 \ldots M_p$ is of the form $xN_1 \ldots N_p$ where the N_i's are reducts of the M_i's. Therefore all elements of $\mathcal{N}_0$ are strongly normalisable.

(B) The $M_1, \ldots, M_p$ in the definition of $\mathcal{N}_0$ serve precisely that purpose.

(C) Let $M \in \mathcal{N}_0 \to \mathcal{N}$. Then $Mx \in \mathcal{N}$, and a fortiori $M \in \mathcal{N}$.

Now we can prove (2′) and (3). The two properties hold at basic types because we have chosen $V(\kappa) = \mathcal{N}$, and by (A). The intersection case is obvious. Let thus $\sigma = \sigma_1 \to \sigma_2$. By induction we have $V(\sigma_1) \subseteq \mathcal{N}$ and $\mathcal{N}_0 \subseteq V(\sigma_2)$, hence $\mathcal{N} \to \mathcal{N}_0 \subseteq V(\sigma)$, and (2′) at σ then follows by (B). Similarly, we use induction and (C) to prove (3) at σ.

($\Rightarrow$) By induction on $(depth(M), size(M))$, and by cases:

- $M = \lambda x_1 \ldots x_m.xN_1 \ldots N_n$, with $x \neq x_1, \ldots, x_m$: By induction and by lemmas 3.5.6 and 3.5.2, we have $\Delta \vdash N_1 : \sigma_1, \ldots, \Delta \vdash N_n : \sigma_n$ for some $\Delta = \Gamma, x_1 : \tau_1, \ldots, x_m : \tau_m, x : \tau$. Then we have, using lemma 3.5.5:

$$\Gamma, x : \tau \wedge (\sigma_1 \to \cdots \to \sigma_n \to \kappa) \vdash M : \tau_1 \to \cdots \to \tau_m \to \kappa.$$

- $M = \lambda x_1 \ldots x_m.x_iN_1 \ldots N_n$: Similar to the previous case.

- $M = \lambda x_1 \ldots x_m.(\lambda x.N)PN_1 \ldots N_n$: Then by induction $\Delta \vdash P : \sigma$ and $\Delta \vdash N[P/x]N_1 \ldots N_n : \tau$ for some $\Delta = \Gamma, x_1 : \tau_1, \ldots, x_m : \tau_m$. We claim that $\Delta \vdash (\lambda x.N)PN_1 \ldots N_n : \tau$, from which $\Gamma \vdash M : \tau_1 \to \cdots \to \tau_m \to \tau$ follows. This is proved by induction on n (cf. theorem 3.5.12), the base case $n = 0$ being lemma 3.5.10 (2). □

In the following three exercises, we present the logical approximation theorem, and derive as corollaries characterisations of the terms having a head normal form and of the terms having a normal form. We follow [RdR93].

Exercise 3.5.18 (logical approximation) * *We define the following notion of "parallel normal reduction", denoted with* $\overset{norm}{\Longrightarrow}$*:*

$$\frac{}{(\lambda x.P)QM_1 \ldots M_n \overset{norm}{\Longrightarrow} P[Q/x]M_1 \ldots M_n}$$

$$\frac{P \overset{norm}{\Longrightarrow} Q}{xM_1 \ldots M_{i-1}PM_{i+1} \ldots M_n \overset{norm}{\Longrightarrow} xM_1 \ldots M_{i-1}QM_{i+1} \ldots M_n} \qquad \frac{M \overset{norm}{\Longrightarrow} N}{\lambda x.M \overset{norm}{\Longrightarrow} \lambda x.N}$$

Show that the following implication holds, for any Γ, M, σ*:*

$$\Gamma \vdash M : \sigma \quad \Rightarrow \quad \exists N \;\; M \overset{norm}{\Longrightarrow}{}^* N \text{ and } \Gamma \vdash \omega(N) : \sigma.$$

Hints (refering to the proof of theorem 3.5.20): (1) The following easy property is useful: in $\mathcal{D}\Omega$*, if* $\Gamma \vdash M : \sigma$ *and* $M \leq N$*, then* $\Gamma \vdash N : \sigma$*. (2) One should now deal with typed versions of the predicates* $\mathcal{N}$ *and* $\mathcal{N}_0$*. Specifically, set*

$$\begin{array}{rcl} \mathcal{N}(\Gamma, \sigma) & = & \{M \in \Lambda \mid \exists N \;\; (M \overset{norm}{\Longrightarrow}{}^* N \text{ and } \Gamma \vdash \omega(N) : \sigma)\} \\ \mathcal{N}_0(\Gamma, \sigma) & = & \{M \in \mathcal{N}(\Gamma, \sigma) \mid M \text{ has the form } xM_1 \ldots M_p\}\,. \end{array}$$

(3) Formulate and prove typed versions of properties (A) (B), and (C) (plus a property saying that the predicates at $\sigma \wedge \tau$ *are the intersections of the predicates at* σ *and* τ*),*

as well as of properties (1), (2′), and (3) (in the latter the type should be restricted to a non-trivial one, see exercise 3.5.19. (4) In the proof of (C), observe that if

$$Mx \overset{norm}{\Longrightarrow}{}^* (\lambda y.M)x \overset{norm}{\Longrightarrow} M[x/y] \overset{norm}{\Longrightarrow}{}^* P$$

then

$$M \overset{norm}{\Longrightarrow}{}^* \lambda y.M \overset{norm}{\Longrightarrow}{}^* \lambda y.P[y/x].$$

(5) Formulate interpretations of the form $V(\Gamma,\sigma)$ (making use of the operation $\uplus$ defined in proposition 3.5.6 for the arrow case), and prove a version of lemma 3.5.15.

Exercise 3.5.19 *Show that the following are equivalent for a λ-term M:*

1. *$M =_\beta N$ for some head normal form N,*
2. *$M \overset{norm}{\longrightarrow}{}^* N$ for some head normal form N (cf. definition 2.1.19),*
3. *M is typable with a non-trivial type in $\mathcal{D}\Omega$.*

where the non-trivial types are defined as follows: atomic types are non-trivial, $\sigma \wedge \tau$ is non-trivial provided one of σ or τ is non-trivial, and $\sigma \to \tau$ is non-trivial provided τ is non-trivial. Hints: Ω can ony have a trivial type. Any term $xM_1 \ldots M_n$ is typable in any environment $x : \omega \to \cdots \to \omega \to \sigma$.

Exercise 3.5.20 *Show that the following are equivalent for a λ-term M:*

1. *M is normalisable,*
2. *the leftmost reduction from M terminates (cf. proposition 2.2.18),*
3. *$\Gamma \vdash M : \sigma$ in $\mathcal{D}\Omega$ for some Γ,σ where ω does not occur.*

On the way, show the following properties:

- *If $\Gamma \vdash M : \sigma$, where M is a β-normal form and where ω does not occur in Γ,σ, then Ω does not occur in M.*

- *Every β normal form M is typable in $\mathcal{D}$.*

Exercise 3.5.21 *Show that the logical approximation theorem still holds, replacing the type system $\mathcal{D}\Omega$ by the type system of figure 3.2. (Warning: this involves revisiting a number of syntactic lemmas, typically lemma 3.5.8.)*

Chapter 4

Interpretation of λ-Calculi in CCC's

In first approximation, typed λ-calculi are *natural deduction* presentations of certain fragments of minimal logic (a subsystem of intuitionistic logic). These calculi have a natural computational interpretation as core of typed functional languages where computation, intended as $\beta\eta$-reduction, corresponds to proof normalization. In this perspective, we reconsider in section 4.1 the simply typed λ-calculus studied in chapter 2. We exhibit a precise correspondence between the simply typed λ-calculus and a natural deduction formalization of the implicative fragment of propositional implicative logic.

Next, we address the problem of *modelling* the notions of $\beta\eta$-reduction and equivalence. It turns out that simple models can be found by interpreting types as sets and terms as functions between these sets. But, in general, which are the structural properties that characterize such models? The main problem considered in this chapter is that of understanding what is the *model theory* of simply typed and untyped λ-calculi. In order to answer this question, we introduce in section 4.2 the notion of cartesian closed category (CCC). We present CCC's as a natural categorical generalization of certain adjunctions found in Heyting algebras. As a main example, we show that the category of directed complete partial orders and continuous functions is a CCC.

The description of the models of a calculus by means of category-theoretical notions will be a central and recurring topic of this book. We will not always fully develop the theory but in this chapter we can take advantage of the simplicity of the calculus to go into a complete analysis. In section 4.3, we describe the interpretation of the simply typed λ-calculus into an arbitrary CCC, and we present some basic properties such as the substitution theorem. The interpretation into a categorical language can be seen as a way of implementing α-renaming and substitution. This eventually leads to the definition of a *categorical abstract machine.*

In section 4.4, we address the problem of understanding which equivalence is induced on terms by the interpretation in a CCC. To this end, we introduce the notion of λ-theory. Roughly speaking, a λ-theory is a congruence over λ-terms which includes $\beta\eta$-equivalence. It turns out that every CCC induces a λ-theory. Vice versa, one may ask: does any λ-theory come from the interpretation in a CCC? We answer this question positively by showing how to build a suitable CCC from any λ-theory. This concludes our development of a *model theory* for the simply typed λ-calculus. Related results will be presented in chapter 6 for PCF, a simply typed λ-calculus extended with arithmetical operators and fixed point combinators.

In section 4.5 we introduce *logical relations* which are a useful tool to establish links between syntax and semantics. In particular, we apply them to the problem of characterizing equality in the set-theoretical model of the simply typed λ-calculus, and to the problem of understanding which elements of a model are definable by a λ-term.

In section 4.6 we regard the untyped λ-calculus as a typed λ-calculus with a *reflexive type*. We show that that every CCC with a reflexive object gives rise to an untyped λ-theory. We present a general method to build a category of retractions out of a reflexive object in a CCC. We give two applications of this construction. First, we hint to the fact that every untyped λ-theory is induced by a reflexive object in a CCC (this is the analogue of the result presented in section 4.4 for the simply typed λ-calculus). Second, we adopt the category of retractions as a frame for embedding algebraic structures in λ-models. Following Engeler, we describe a method to encode standard mathematical structures in λ-models.

This chapter is mainly based on [LS86, Sco80, Cur86] to which the reader seeking more advanced results is addressed.

4.1 Simply Typed λ-Calculus

In chapter 2, we have presented a simply typed λ-calculus in which every subterm is labelled by a type. This was well-suited to our purposes but it is probably not the most illuminating treatment. So far, we have (mainly) discussed the λ-calculus as a core formalism to compute functions-as-algorithms. The simply typed λ-calculus receives an additional interpretation: it is a language of proofs for minimal logic. Let us revisit simple types first, by considering basic types as atomic propositions and the function space symbol as implication

$$\begin{array}{ll} At & ::= \kappa \mid \kappa' \mid \cdots \\ \sigma & ::= At \mid (\sigma \to \sigma)\ . \end{array}$$

Forgetting the terms for a while, we briefly describe the provability of formulas for this rudimentary logic. We use a deduction style called *natural deduction* [Pra65]. A formula σ is proved relatively to a list $\sigma_1, \ldots, \sigma_n$ of assumptions. The

$$\frac{1 \leq i \leq n}{\sigma_1, \ldots, \sigma_n \vdash \sigma_i}$$
$$\frac{\sigma_1, \ldots, \sigma_n, \sigma \vdash \tau}{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau}$$
$$\frac{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau \quad \sigma_1, \ldots, \sigma_n \vdash \sigma}{\sigma_1, \ldots, \sigma_n \vdash \tau}$$

Figure 4.1: Natural deduction for minimal implicative logic

formal system described in figure 4.1 allows us to derive judgments of the form $\sigma_1, \ldots, \sigma_n \vdash \sigma$, which are called sequents.

An important remark with a wide range of possible applications [How80] is that proofs in natural deduction can be encoded precisely as λ-terms. To this aim hypotheses are named by variables. Raw terms are defined by the following BNF (in the following, we feel free to spare on parentheses):

$$\begin{array}{ll} v & ::= x \mid y \mid \ldots \\ M & ::= v \mid (\lambda v : \sigma.M) \mid (MM) \ . \end{array}$$

A *context* Γ is a list of pairs, $x : \sigma$, where x is a variable, all variables are distinct, and σ is a type. We write $x : \sigma \in \Gamma$ to express that the pair $x : \sigma$ occurs in Γ. A judgment has the shape $\Gamma \vdash M : \sigma$. Whenever we write $\Gamma \vdash M : \sigma$ it is intended that the judgment is provable. We also write $M : \sigma$ to say that there exists a context Γ such that $\Gamma \vdash M : \sigma$. A term M with this property is called well-typed. Provable judgments are inductively defined in figure 4.2. We may omit the labels on the λ-abstractions when the types are obvious from the context. It is easily seen that any derivable judgment admits a unique derivation, thus yielding a one-to-one correspondence between proofs and terms.

Yet another presentation of the typing rules omits all type information in the λ-terms. The corresponding typing system is obtained from the one in figure 4.2 by removing the type σ in $\lambda x : \sigma.M$. In this case a term in a given context can be given several types. For instance the term $\lambda x.x$ can be assigned in the empty context any type $\sigma \to \sigma$, for any σ. To summarize, we have considered three styles of typing:

(1) A totally explicit typing where every subterm is labelled by a type (see section 2.2).

(2) A more economic typing, where only the variables bound in abstractions are labelled by a type. This style is known as "typing à la Church".

$$(\mathrm{Asmp})\quad \frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

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

$$(\to_E)\quad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

Figure 4.2: Typing rules for the simply typed λ-calculus

(3) A type assignment system, where an untyped term receives a type. This is known as "typing à la Curry".

In the first system, the term itself carries all the typing information. We note that once we have labelled free variables and λ-abstractions the label of each subterm can be reconstructed in a unique way. In the two other systems, à la Church and à la Curry, a separate context carries type information. In the system à la Church, the context together with the types of bound variables carry all the necessary information to reconstruct uniquely the type of the term. In the system à la Curry, a term, even in a given environment, may have many types. In general, the problem of deciding if an untyped λ-term has a type in a given context is a non-trivial one. This is referred to as the *type-inference* or, equivalently, *type reconstruction* problem.

Type reconstruction algorithms are quite relevant in practice as they relieve the programmer from the burden of explicitly writing all type information and allow for some form of polymorphism. For the simply typed discipline presented here, it can be shown that the problem is decidable and that it is possible to represent by a type schema (a type with type variables) all derivable solutions to a given type reconstruction problem [Hin69]. On the other hand, the type-inference problem turns out to be undecidable in certain relevant type disciplines (e.g. second order [Wel94]).

In this chapter, we concentrate on the interpretation of λ-terms with *explicit* type information. We regard these calculi à la Church as central, by virtue of their strong ties with category theory and proof theory. The interpretation of type assignment systems has already been considered in chapter 3, and it will be further developed in chapter 15.

Exercise 4.1.1 *This exercise gives a more precise relation between the three systems mentioned above. Let M^σ be a totally explicitly typed term. Let $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ be its free variables. Let erase be the function that erases all type information in a λ-term.*

Show that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash erase(M) : \sigma$ is derivable. Define a function semi-erase such that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash semi\text{-}erase(M) : \sigma$ is à la Church derivable. Conversely, from a derivation à la Curry of $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma$, construct a totally explicitly typed term N^σ, whose free variables are $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$, and such that $erase(N^\sigma) = M$. Design a similar transformation from a derivation à la Church. Investigate how these transformations compose.

Exercise 4.1.2 *Show that the structural rules of exchange, weakening, and contraction are derived in the system above, in the sense that, if the premises are provable, then the conclusion is provable.*

$$\begin{array}{lll}(exch) & \Gamma, x : \sigma, y : \tau, \Gamma' \vdash M : \rho & \Rightarrow \;\; \Gamma, y : \tau, x : \sigma, \Gamma' \vdash M : \rho \\ (weak) & \Gamma \vdash M : \tau \text{ and } x : \sigma \notin \Gamma & \Rightarrow \;\; \Gamma, x : \sigma \vdash M : \tau \\ (contr) & \Gamma, x : \sigma, y : \sigma \vdash M : \tau & \Rightarrow \;\; \Gamma, z : \sigma \vdash M[z/x, z/y] : \tau \;\; (z \text{ fresh}) \; .\end{array}$$

We consider two basic axioms for the reduction of terms (cf. chapter 2)

$$\begin{array}{ll}(\beta) & (\lambda x : \sigma.M)N \to M[N/x] \\ (\eta) & \lambda x : \sigma.(Mx) \to M \;\; \text{if } x \notin FV(M)\end{array}$$

we denote with $\to_{\beta\eta}$ their compatible (or contextual) closure (cf. figure 2.4), and with $\to^*_{\beta\eta}$ the reflexive and transitive closure of $\to_{\beta\eta}$.

Exercise 4.1.3 (subject reduction) *Show that well-typed terms are closed under reduction, formally:*

$$\Gamma \vdash M : \sigma \text{ and } M \to_{\beta\eta} N \;\; \Rightarrow \;\; \Gamma \vdash N : \sigma \; .$$

Theorem 4.1.4 (confluence and normalization) *(1) The reduction relation $\to^*_{\beta\eta}$ is confluent (both on typed and untyped λ-terms).*
(2) The reduction system $\to_{\beta\eta}$ is strongly normalizing on well-typed terms, that is if $M : \sigma$ then all reduction sequences lead to a $\beta\eta$-normal form.

We have already proved these properties in chapter 2 for the untyped β-reduction. Using subject reduction (exercise 4.1.3) the proof-techniques can be easily adapted to the typed case. The following exercise provides enough guidelines to extend the results to $\beta\eta$-reduction.

Exercise 4.1.5 *In the following $\to^{\leq 1}$ means reduction in 0 or 1 step.*
(1) If $M \to_\eta M_1$ and $M \to_\eta M_2$ then there is an N such that $M_1 \to^{\leq 1}_\eta N$ and $M_2 \to^{\leq 1}_\eta N$.
(2) If $M \to_\eta M_1$ and $M \to_\beta M_2$ then there is an N such that $M_1 \to^{\leq 1}_\beta N$ and $M_2 \to^*_\eta N$.
(3) If $M \to_\eta \cdot \to_\beta N$ then $M \to_\beta \cdot \to_\beta N$ or $M \to_\beta \cdot \to^*_\eta N$, where $M \to_{R_1} \cdot \to_{R_2} N$ stands for $\exists P\,(M \to_{R_1} P$ and $P \to_{R_2} N)$.

$$
(Asmp)\ \frac{x:\sigma\in\Gamma}{\Gamma\vdash x:\sigma}
\qquad
(\times_I)\ \frac{\Gamma\vdash M:\sigma\quad \Gamma\vdash N:\tau}{\Gamma\vdash\langle M,N\rangle:\sigma\times\tau}
$$

$$
(\times_{E1})\ \frac{\Gamma\vdash M:\sigma\times\tau}{\Gamma\vdash\pi_1 M:\sigma}
\qquad
(\times_{E2})\ \frac{\Gamma\vdash M:\sigma\times\tau}{\Gamma\vdash\pi_2 M:\tau}
$$

Figure 4.3: Typing rules for a calculus of conjunction

4.2 Cartesian Closed Categories

The reader will find in appendix B some basic notions of category theory. Next, we motivate the introduction of CCC's as the combination of two more elementary concepts.

Example 4.2.1 (conjunction and binary products) *Let us consider a simple calculus in which we can pair two values or project a pair to one of its components.*

$$
\begin{array}{lll}
\textit{Types:} & At & ::= \kappa \mid \kappa' \mid \cdots \\
 & \sigma & ::= At \mid (\sigma\times\sigma) \\
\textit{Terms:} & v & ::= x \mid y \mid \cdots \\
 & M & ::= v \mid \langle M,M\rangle \mid \pi_1 M \mid \pi_2 M
\end{array}
$$

This calculus corresponds to the conjunctive fragment of minimal logic. Its typing rules are shown in figure 4.3.

It is intuitive that a cartesian category (i.e. a category with a terminal object and binary products) has something to do with this calculus. Let us make this intuition more precise:

(1) We interpret a type σ as an object $[\![\sigma]\!]$ of a cartesian category **C**. The interpretation of the type $\sigma\times\tau$ is the cartesian product $[\![\sigma]\!]\times[\![\tau]\!]$.

(2) If types are objects, it seems natural to associate terms to morphisms. If M is a closed term of type σ we may expect that its interpretation is a morphism $f:1\to[\![\sigma]\!]$, where 1 is the terminal object. But what about a term M such that $x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash M:\sigma$? The idea is to interpret this term as a morphism $f:(\cdots(1\times[\![\sigma_1]\!])\times\cdots\times[\![\sigma_n]\!])\to[\![\sigma]\!]$.

This example suggests that types can be seen as objects and terms as morphisms. We do not wish to be more precise at the moment (but see section 3) and leave the following as an exercise.

Exercise 4.2.2 *Define an interpretation of the typed terms of the calculus of conjunction into a cartesian category.*

There is a well-known correspondence between classical propositional logic and boolean algebras: a formula is provable iff it is valid in every boolean algebra interpretation. *Heyting algebras* play a similar role for intuitionistic (or minimal) logic.

Definition 4.2.3 (Heyting algebras) *A Heyting algebra H is a lattice with lub operation $\vee$, glb operation $\wedge$, greatest element 1, least element 0, and with a binary operation $\rightarrow$ that satisfies the condition*

$$x \wedge y \leq z \quad \textit{iff} \quad x \leq y \rightarrow z \ .$$

Exercise 4.2.4 *Heyting algebras abound in nature. Show that the collection Ω of open sets of a topological space (X, Ω) ordered by inclusion can be seen as a Heyting algebra by taking*

$$U \rightarrow V = \bigcup \{W \in \Omega \mid W \subseteq (X \backslash U) \cup V\} \ .$$

For our purposes the important point in the definition of Heyting algebra is that the implication is characterized by an adjoint situation (in a poset case), as for any $y \in H$ the function $\_ \wedge y$ is left adjoint to the function $y \rightarrow \_$

$$\forall y \in H \, (\_ \wedge y) \dashv (y \rightarrow \_) \ .$$

In poset categories the interpretation of proofs is trivial. For this reason Heyting algebras cannot be directly applied to the problem of interpreting the simply typed λ-calculus. However combined with our previous example they suggest a natural generalization: consider a cartesian category in which each functor $\_ \times A$ has a right adjoint $(\_)^A$. In this way we arrive at the notion of CCC. The adjunction condition can be reformulated in a more explicit way, as shown in the following definition.

Definition 4.2.5 (CCC) *A category* $\mathbf{C}$ *is called* cartesian closed *if it has:*

(1) *A terminal object* 1.

(2) *For each $A, B \in \mathbf{C}$ a product given by an object $A \times B$ with projections $\pi_A : A \times B \rightarrow A$ and $\pi_B : A \times B \rightarrow B$ such that:*

$$\forall C \in \mathbf{C} \forall f : C \rightarrow A \forall g : C \rightarrow B \, \exists! h : C \rightarrow A \times B \, (\pi_A \circ h = f \textit{ and } \pi_B \circ h = g) \ .$$

h is often denoted by $\langle f, g \rangle$, where $\langle \_, \_ \rangle$ is called the pairing operator. π_1 and π_2 are equivalent notations for π_A and π_B respectively.

(3) *For each $A, B \in \mathbf{C}$ an* exponent *given by an object B^A with $ev : B^A \times A \rightarrow B$ such that*

$$\forall C \in \mathbf{C} \forall f : C \times A \rightarrow B \, \exists! h : C \rightarrow B^A \, (ev \circ (h \times id) = f) \ .$$

h is often denoted by $\Lambda(f)$, Λ is called the currying operator, and ev the evaluation morphism.

In the following B^A and $A \Rightarrow B$ are interchangeable notations for the exponent object in a category.

Exercise 4.2.6 *Given a CCC* $\mathbf{C}$*, extend the functions* $Prod(A,B) = A \times B$ *and* $Exp(A,B) = B^A$ *to functors* $Prod : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ *and* $Exp : \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$*.*

Exercise 4.2.7 *Show that a CCC can be characterized as a category* $\mathbf{C}$ *such that the following functors have a right adjoint: (i) the unique functor* $! : \mathbf{C} \to \mathbf{1}$*, (ii) the diagonal functor* $\Delta : \mathbf{C} \to \mathbf{C} \times \mathbf{C}$ *defined by* $\Delta(c) = (c, c)$ *and* $\Delta(f) = (f, f)$*, (iii) the functors* $\_ \times A : \mathbf{C} \to \mathbf{C}$*, for any object* A*.*

It is possible to *skolemize* the definition of CCC, that is to eliminate the existential quantifications, using the type operators 1, $(\_ \times \_)$, $(\_)^{(\_)}$ and the term operators $*$, $\langle \_, \_ \rangle$, $\Lambda(\_)$. In this way the theory of CCC's can be expressed as a typed equational theory as shown in the following.

Exercise 4.2.8 *Show that a CCC can be characterized as a category* $\mathbf{C}$ *such that the following equations hold.*

- *There are* $1 \in \mathbf{C}$ *and* $*_A : A \to 1$*, such that for all* $f : A \to 1$*,*

$$(!) \quad f = *_A \ .$$

- *There are* $\pi_1 : A \times B \to A$ *and* $\pi_2 : A \times B \to B$*, for any* $A, B \in \mathbf{C}$*, and* $\langle f, g \rangle : C \to A \times B$ *for any* $f : C \to A$*,* $g : C \to B$*, such that for all* $f : C \to A$*,* $g : C \to B$*,* $h : C \to A \times B$*,*

$$\begin{array}{ll} (Fst) & \pi_1 \circ \langle f, g \rangle = f \\ (Snd) & \pi_2 \circ \langle f, g \rangle = g \\ (SP) & \langle \pi_1 \circ h, \pi_2 \circ h \rangle = h \ . \end{array}$$

- *There are* $ev : B^A \times A \to B$ *for any* $A, B \in \mathbf{C}$*, and* $\Lambda(f)$ *for any* $f : C \times A \to B$*, such that for all* $f : C \times A \to B$*,* $h : C \to B^A$*,*

$$\begin{array}{ll} (\beta_{cat}) & ev \circ (\Lambda(f) \times id) = f \\ (\eta_{cat}) & \Lambda(ev \circ (h \times id)) = h \end{array}$$

where $f \times g = \langle f \circ \pi_1, g \circ \pi_2 \rangle$*.*

Exercise 4.2.9 *Referring to exercise 4.2.8 prove that (SP) is equivalent to*

$$\begin{array}{ll} (DPair) & \langle f, g \rangle \circ h = \langle f \circ h, g \circ h \rangle \\ (FSI) & \langle \pi_1, \pi_2 \rangle = id \end{array}$$

and that (β_{cat}) *and* (η_{cat}) *are equivalent to*

$$\begin{array}{ll} (Beta) & ev \circ \langle \Lambda(f), g \rangle = f \circ \langle id, g \rangle \\ (D\Lambda) & \Lambda(f) \circ h = \Lambda(f \circ (h \times id)) \\ (AI) & \Lambda(ev) = id \ . \end{array}$$

Exercise 4.2.10 *Show that the following categories are cartesian closed: (a) (Finite) Sets. (b) (Finite) Posets and monotonic functions. On the other hand prove that the category* **pSet** *of sets and partial functions is not cartesian closed. Hint: consider the existence of an isomorphism between* $\mathbf{pSet}[2 \times 2, 1]$ *and* $\mathbf{pSet}[2, 4]$.

One can now formally prove that the category of directed complete partial orders (dcpo's) and maps preserving lub's of directed sets is cartesian closed using propositions 1.4.1 and 1.4.4. Exercise 1.4.6 does not say directly that the product construction in **Dcpo** yields a categorical product. This follows from the following general (meta)-property.

Exercise 4.2.11 *Let* $\mathbf{C}, \mathbf{C}'$ *be categories, and* $F : \mathbf{C} \to \mathbf{C}'$ *be a faithful functor. Suppose that* $\mathbf{C}'$ *has products, and that for any pair of objects* A *and* B *of* $\mathbf{C}$ *there exists an object* C *and two morphisms* $\alpha : C \to A$ *and* $\beta : C \to B$ *in* $\mathbf{C}$ *such that:*

$$F(C) = F(A) \times F(B), \quad F(\alpha) = \pi_1, \quad F(\beta) = \pi_2$$

and for any object D *and morphisms* $f : D \to A$, $g : D \to B$, *there exists a morphism* $h : D \to C$ *such that* $F(h) = \langle F(f), F(g)\rangle$. *Show that* $\mathbf{C}$ *has products. Explain why this general technique applies to* **Dcpo**.

In a similar way one can verify that the function space construction in **Dcpo** yields a categorical exponent. The check is slightly more complicated than for the product, due to the fact that the underlying set of the function space in **Dcpo** is a proper subset of the function space in **Set**.

Exercise 4.2.12 *Let* $\mathbf{C}$, $\mathbf{C}'$ *be categories, and* $F : \mathbf{C} \to \mathbf{C}'$ *be a faithful functor. Suppose that the assumptions of exercise 4.2.11 hold, and use* $\times$ *to denote the cartesian product in* $\mathbf{C}$. *Suppose that* $\mathbf{C}'$ *has exponents, and that for any pair of objects* A *and* B *of* $\mathbf{C}$ *there exists an object* C *of* $\mathbf{C}$, *a mono* $m : FC \to FB^{FA}$ *and a morphism* $\gamma : C \times A \to B$ *such that: (1)* $F(\gamma) = ev \circ (M \times id)$, *and (2) for any object* D *and arrow* $f : D \times A \to B$, *there exists a morphism* $k : D \to C$ *such that* $m \circ F(k) = \Lambda(F(f))$. *Show that* $\mathbf{C}$ *has exponents. Apply this to* **Dcpo**.

Theorem 4.2.13 (Dcpo CCC) Dcpo *is a cartesian closed category. The order for products is componentwise, and the order for exponents is pointwise.* **Cpo** *is cartesian closed too.*

PROOF. We can apply the exercises 1.4.6 and 4.2.12. A direct proof of cartesian closure is also possible and easy. For the last part of the statement, notice that $(\bot, \bot)$ is the minimum of $D \times E$, and that the constant function $\lambda d.\bot$ is the minimum of $D \to E$. $\square$

(Asmp)	$[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash x_i : \sigma_i]\!]$	$= \pi_{n,i}$
$(\to_I)$	$[\![\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau]\!]$	$= \Lambda([\![\Gamma, x : \sigma \vdash M : \tau]\!])$
$(\to_E)$	$[\![\Gamma \vdash MN : \tau]\!]$	$= ev \circ \langle [\![\Gamma \vdash M : \sigma \to \tau]\!], [\![\Gamma \vdash N : \sigma]\!] \rangle$

Figure 4.4: Interpretation of the simply typed λ-calculus in a CCC

4.3 Interpretation of λ-Calculi

We explain how to interpret the simply typed λ-calculus in an arbitrary CCC. Suppose that **C** is a CCC. Let us choose a terminal object 1, a product functor $\times : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and an exponent functor $\Rightarrow: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{C}$. Then there is an obvious interpretation for types as objects of the category which is determined by the interpretation of the atomic types. The arrow is interpreted as exponentiation in **C**. Hence given an interpretation $[\![\kappa]\!]$ for the atomic types, we have:

$$[\![\sigma \to \tau]\!] = [\![\sigma]\!] \Rightarrow [\![\tau]\!] \, .$$

Consider a provable judgment of the shape $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma$. Its interpretation will be defined *by induction on the length of the proof* as a morphism from $[\![\Gamma]\!]$ to $[\![\sigma]\!]$, where we set $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$ and $[\![\Gamma]\!] = 1 \times [\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!]$. We will take the convention that $\times$ associates to the left. We denote with $\pi_{n,i} : [\![\Gamma]\!] \to [\![\sigma_i]\!]$ $(i = 1, \ldots, n)$ the morphism: $\pi_2 \circ \pi_1 \circ \cdots \circ \pi_1$, where π_1 is iterated $(n - i)$ times.

The interpretation is defined in figure 4.4. The last two rules need some explanation. Suppose $C = [\![\Gamma]\!]$, $A = [\![\sigma]\!]$, and $B = [\![\tau]\!]$.

• $(\to_I)$ If there is a morphism $f : C \times A \to B$ then there is a uniquely determined morphism $\Lambda(f) : C \to B^A$.

• $(\to_E)$ If there are two morphisms $f : C \to B^A$ and $g : C \to A$, then one can build the morphism $\langle f, g \rangle : C \to B^A \times A$ and composing with ev one gets $ev \circ \langle f, g \rangle : C \to A$.

Sometimes, we write $[\![M]\!]$ as an abbreviation for $[\![\Gamma \vdash M : \sigma]\!]$. When composing the interpretation of the judgment $\Gamma \vdash M : \tau$ with an environment, that is a morphism in $\mathbf{C}[1, [\![\Gamma]\!]]$, we will freely use the notation $[\![M]\!] \circ \langle d_1, \ldots, d_n \rangle$ which relies on an n-ary product.

In section 4.5 we will work with a simply typed λ-calculus enriched with a set of constants C. We suppose that each constant is labelled with its type, say c^σ. The typing system is then enriched with the rule:

$$\frac{}{\Gamma \vdash c^\sigma : \sigma} \tag{4.1}$$

We denote with $\Lambda(C)$ the collection of well-typed terms. The interpretation is fixed by providing for each constant c^σ a morphism $f_c : 1 \to [\![\sigma]\!]$. The judgment $\Gamma \vdash c : \sigma$ is then interpreted by composing with the terminal morphism:

$$[\![\Gamma \vdash c^\sigma : \sigma]\!] = f_c \circ ! \tag{4.2}$$

The interpretation in figure 4.4 is defined by induction on the structure of a proof of a judgment $\Gamma \vdash M : \sigma$. In the simple system we presented here a judgment has a *unique* proof. However, in general, there can be several ways of deriving the same judgment, therefore a problem of *coherence* of the interpretation arises, namely one has to show that different proofs of the same judgment receive the same interpretation. Note that in the simply typed calculus the coherence problem is avoided by getting rid of the structural rules. This trick does not suffice in more sophisticated type theories like LF (see chapter 11) where the derivation is not completely determined by the structure of the judgment. In this case term judgments and type judgments are inter-dependent.

Exercise 4.3.1 *Show that if* $\Gamma \vdash M : \tau$ *and* $x : \sigma \notin \Gamma$ *then* $\Gamma, x : \sigma \vdash M : \tau$ *(cf. exercise 4.1.2) and* $[\![\Gamma, x : \sigma \vdash M : \tau]\!] = [\![\Gamma \vdash M : \tau]\!] \circ \pi_1$.

Exercise 4.3.2 *Given two contexts* $\Gamma, x : \sigma, y : \tau, \Gamma'$ *and* $\Gamma, y : \tau, x : \sigma, \Gamma'$ *define an isomorphism* ϕ *between the corresponding objects. Hint: if* $\Gamma \equiv z : \rho$ *and* Γ' *is empty then* $\phi \equiv \langle\langle \pi_1 \circ \pi_1, \pi_2\rangle, \pi_2 \circ \pi_1\rangle : (C \times A) \times B \to (C \times B) \times A$. *Show that (cf. exercise 4.1.2)*

$$[\![\Gamma, x : \sigma, y : \tau, \Gamma' \vdash M : \rho]\!] = [\![\Gamma, y : \tau, x : \sigma, \Gamma' \vdash M : \rho]\!] \circ \phi \ .$$

The next step is to analyse the interpretation of substitution in a category.

Theorem 4.3.3 (substitution) *If* $\Gamma, x : \sigma \vdash M : \tau$, *and* $\Gamma \vdash N : \sigma$ *then (1)* $\Gamma \vdash M[N/x] : \tau$, *and (2)* $[\![\Gamma \vdash M[N/x] : \tau]\!] = [\![\Gamma, x : \sigma \vdash M : \tau]\!] \circ \langle id, [\![\Gamma \vdash N : \sigma]\!]\rangle$.

PROOF. (1) By induction on the length of the proof of $\Gamma, x : \sigma \vdash M : \tau$. The interesting case arises when the last deduction is by $(\to_I)$:

$$\frac{\Gamma, x : \sigma, y : \tau \vdash M : \tau'}{\Gamma, x : \sigma \vdash \lambda y : \tau.M : \tau \to \tau'}$$

We observe $(\lambda y : \tau.M)[N/x] \equiv \lambda y : \tau.M[N/x]$. We can apply the inductive hypothesis on $\Gamma, y : \tau, x : \sigma \vdash M : \tau'$ (note the exchange on the assumptions) to get $\Gamma, y : \tau \vdash M[N/x] : \tau'$ from which $\Gamma \vdash (\lambda y : \tau.M)[N/x] : \tau \to \tau'$ follows by $(\to_I)$.

(2) We will use the exercises 4.3.1 and 4.3.2 on the interpretation of weakening and exchange. Again we proceed by induction on the length of the proof of

$\Gamma, x : \sigma \vdash M : \tau$ and we just consider the case $(\to_I)$. Let:

$$\begin{cases} f_1 = [\![\Gamma \vdash \lambda y : \tau.M[N/x] : \tau \to \tau']\!] & : C \to B'^B \\ g_1 = [\![\Gamma, y : \tau \vdash M[N/x] : \tau']\!] & : C \times B \to B' \\ f_2 = [\![\Gamma, x : \sigma \vdash \lambda y : \tau.M : \tau \to \tau']\!] & : C \times A \to B'^B \\ g_2 = [\![\Gamma, y : \tau, x : \sigma \vdash M : \tau']\!] & : (C \times B) \times A \to B' \\ f_3 = [\![\Gamma \vdash N : \sigma]\!] & : C \to A \\ g_3 = [\![\Gamma, y : \tau \vdash N : \sigma]\!] & : C \times B \to A \\ g_2' = [\![\Gamma, x : \sigma, y : \tau \vdash M : \tau']\!] & : (C \times A) \times B \to B' \ . \end{cases}$$

We have to show $f_1 = f_2 \circ \langle id, f_3 \rangle$, knowing by induction hypothesis that $g_1 = g_2 \circ \langle id, g_3 \rangle$. We observe that $f_1 = \Lambda(g_1)$, $f_2 = \Lambda(g_2')$, and $g_2' = g_2 \circ \phi$, where $\phi = \langle \langle \pi_1 \circ \pi_1, \pi_2 \rangle, \pi_2 \circ \pi_1 \rangle$ is the iso given by exercise 4.3.2. Moreover $g_3 = f_3 \circ \pi_1$. We then compute

$$\begin{aligned} f_2 \circ \langle id, f_3 \rangle &= \Lambda(g_2') \circ \langle id, f_3 \rangle \\ &= \Lambda(g_2' \circ (\langle id, f_3 \rangle \times id)) \ . \end{aligned}$$

So it is enough to show $g_1 = g_2' \circ (\langle id, f_3 \rangle \times id)$. We compute on the right hand side

$$\begin{aligned} g_2' \circ (\langle id, f_3 \rangle \times id) &= g_2 \circ \phi \circ \langle \langle \pi_1, f_3 \circ \pi_1 \rangle, \pi_2 \rangle \\ &= g_2 \circ \phi \circ \langle \langle \pi_1, g_3 \rangle, \pi_2 \rangle \\ &= g_2 \circ \langle id, g_3 \rangle \ . \end{aligned}$$

□

The categorical interpretation can be seen as a way of compiling a language with variables into a language without variables. The slogan is that *variables are replaced by projections*, for instance $[\![\phi \vdash \lambda x : \sigma.x : \sigma \to \sigma]\!] = \Lambda(\pi_2)$. In other words, rather than giving a symbolic reference in the form of a variable, one provides a path for accessing a certain information in the context. [1] As a matter of fact the *compilation* of the λ-calculus into the categorical language has been taken as a starting point for the definition of an abstract machine (the *Categorical Abstract Machine* (CAM), see [CCM87]) in the style of Landin's classical *SECD* machine [Lan64] (see [Cur86] for a comparison). The purpose of these machines is to provide a high-level description of data structures and algorithms used to reduce efficiently λ-terms. In the CAM approach, a fundamental problem is that of orienting the equations that characterize CCC's as defined in exercise 4.2.8. In the following we drop all type-information and we restrict our attention to the simulation of β-reduction (the treatment of the extensional rules raises additional problems). Hardin [Har89] has studied the term rewriting system described in figure 4.5. The most important results are:

- $\mathcal{E}$ is confluent and strongly normalizing .

[1] de Bruijn conventions for the representation of variables as distances from the respective binders, as well as standard implementations of environments in abstract machines (cf. chapter 8) follow related ideas.

$$(Beta) \quad ev \circ \langle \Lambda(f), g\rangle \to f \circ \langle id, g\rangle$$

$$(\mathcal{E}) \quad \left\{ \begin{array}{ll} (f \circ g) \circ h & \to f \circ (g \circ h) \\ id \circ f & \to f \\ \pi_1 \circ id & \to \pi_1 \\ \pi_2 \circ id & \to \pi_2 \\ \pi_1 \circ \langle f, g\rangle & \to f \\ \pi_2 \circ \langle f, g\rangle & \to g \\ \langle f, g\rangle \circ h & \to \langle f \circ h, g \circ h\rangle \\ \Lambda(f) \circ h & \to \Lambda(f \circ \langle h \circ \pi_1, \pi_2\rangle) \end{array} \right.$$

Figure 4.5: A rewriting system for the β-categorical equations

• $\mathcal{E} + Beta$ is confluent (on a subset of categorical terms which is large enough to contain all the compilations of λ-terms).

The proof of strong normalization of $\mathcal{E}$ is surprisingly difficult [CHR92]. Simpler results have been obtained with a related calculus called $\lambda\sigma$-calculus [ACCL92]. More results on abstract machines which are related to the CAM are described in chapter 8.

4.4 From CCC's to λ-Theories and back

We study the equivalence induced by the interpretation of the simply typed λ-calculus in a CCC. It turns out that the equivalence is closed under $\beta\eta$-conversion and forms a congruence.

Definition 4.4.1 (λ-theory) *Let T be a collection of judgments of the shape $\Gamma \vdash M = N : \sigma$ such that $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$. T is called a λ-theory if it is equal to the smallest set containing T and closed under the rules in figure 4.6.*

We note that the congruence generated by the axioms β and η is the smallest λ-theory. To every CCC we can associate a λ-theory.

Theorem 4.4.2 *Let* **C** *be a CCC and let $[\![\ ]\!]$ be an interpretation in the sense of figure 4.4 of the simply typed λ-calculus defined over* **C**. *Then the following collection is a λ-theory.*

$$Th(\mathbf{C}) = \{\Gamma \vdash M = N : \sigma \mid \Gamma \vdash M : \sigma, \Gamma \vdash N : \sigma, [\![\Gamma \vdash M : \sigma]\!] = [\![\Gamma \vdash N : \sigma]\!]\} \ .$$

$$(\alpha)\quad \frac{\Gamma \vdash \lambda x:\sigma.N : \sigma \to \tau \quad y \notin FV(N)}{\Gamma \vdash \lambda y:\sigma.N[y/x] = \lambda x:\sigma.N : \sigma \to \tau}$$

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

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

$$(\mathit{weak})\quad \frac{\Gamma \vdash M = N : \sigma \quad x:\tau \notin \Gamma}{\Gamma, x:\tau \vdash M = N : \sigma}$$

$$(\mathit{refl})\quad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma}$$

$$(\mathit{sym})\quad \frac{\Gamma \vdash M = N : \sigma}{\Gamma \vdash N = M : \sigma}$$

$$(\mathit{trans})\quad \frac{\Gamma \vdash M = N : \sigma \quad \Gamma \vdash N = P : \sigma}{\Gamma \vdash M = P : \sigma}$$

$$(\xi)\quad \frac{\Gamma, x:\sigma \vdash M = N : \tau}{\Gamma \vdash \lambda x:\sigma.M = \lambda x:\sigma.N : \sigma \to \tau}$$

$$(\mathit{appl})\quad \frac{\Gamma \vdash M = N : \sigma \to \tau \quad \Gamma \vdash M' = N' : \sigma}{\Gamma \vdash MM' = NN' : \tau}$$

$$(\mathit{asmp})\quad \frac{\Gamma \vdash M = N : \sigma \in T}{\Gamma \vdash M = N : \sigma}$$

Figure 4.6: Closure rules for a typed λ-theory

PROOF. We have to check that $Th(\mathbf{C})$ is closed under the rules presented in figure 4.6. For (α) we observe that $[\![\ ]\!]$ is invariant with respect to the names of bound variables.

(β) Let $[\![\Gamma \vdash (\lambda x : \sigma.M)N : \tau]\!] = ev \circ \langle \Lambda(f), g\rangle$, where $f = [\![\Gamma, x : \sigma \vdash M : \tau]\!]$ and $g = [\![\Gamma \vdash N : \sigma]\!]$. By the substitution theorem $[\![\Gamma \vdash M[N/x] : \tau]\!] = f \circ \langle id, g\rangle$, and we compute

$$f \circ \langle id, g\rangle = ev \circ (\Lambda(f) \times id) \circ \langle id, g\rangle = ev \circ \langle \Lambda(f), g\rangle \ .$$

(η) By the following equations:

$$\begin{array}{l}
[\![\Gamma \vdash \lambda x : \sigma.Mx : \sigma \to \tau]\!] \\
= \Lambda(ev \circ \langle [\![\Gamma, x : \sigma \vdash M : \sigma \to \tau]\!], [\![\Gamma, x : \sigma \vdash x : \sigma]\!]\rangle) \\
= \Lambda(ev \circ \langle [\![\Gamma \vdash M : \sigma \to \tau]\!] \circ \pi_1, \pi_2\rangle) \\
= \Lambda(ev \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times id)) \\
= [\![\Gamma \vdash M : \sigma \to \tau]\!] \ .
\end{array}$$

For $(weak)$ we use the exercise 4.3.1. The rules $(refl)$, (sym), $(trans)$ hold since $Th(\mathbf{C})$ is an equivalence. Finally, (ξ), (apl) follow by the definition of the interpretation of abstraction and application. □

Exercise 4.4.3 *Show that there are infinitely many λ-theories. Hints: Interpret the atomic types as finite sets and consider the resulting λ-theory. Then analyse the $\beta\eta$-normal forms of type $(\kappa \to \kappa) \to (\kappa \to \kappa)$.*

Next, we show how to generate a CCC starting from a λ-theory. The construction consists essentially in taking types as objects of the category and (open) terms quotiented by the λ-theory as morphisms (cf. Henkin's term model [Hen50]). We take the following steps:

(1) We extend the language with constructors for terminal object and product, as well as the relative equations:

$$\begin{array}{lll}
\text{Types:} & At & ::= \kappa \mid \kappa' \mid \ldots \\
 & \sigma & ::= At \mid 1 \mid \sigma \times \sigma \mid \sigma \to \sigma \\
\text{Terms:} & v & ::= x \mid y \mid \ldots \\
 & M & ::= v \mid * \mid \langle M, M\rangle \mid \pi_1 M \mid \pi_2 M \mid \lambda v : \sigma.M \mid MM \ .
\end{array}$$

1. Typing Rules. The rules of the simply typed calculus (figure 4.2), plus the rules for conjunction (figure 4.3), plus:

$$(*)\ \frac{}{\Gamma \vdash * : 1} \ .$$

2. Equations. The rules of the pure $\lambda\beta\eta$-theory (figure 4.6) plus:

$$(*)\ \frac{\Gamma\vdash M:1}{\Gamma\vdash M=*}\qquad (SP)\ \frac{\Gamma\vdash M:\sigma\times\tau}{\Gamma\vdash\langle\pi_1 M,\pi_2 M\rangle=M:\sigma\times\tau}$$
$$(\pi_1)\ \frac{\Gamma\vdash\langle M,N\rangle:\sigma\times\tau}{\Gamma\vdash\pi_1\langle M,N\rangle=M:\sigma}\qquad (\pi_2)\ \frac{\Gamma\vdash\langle M,N\rangle:\sigma\times\tau}{\Gamma\vdash\pi_2\langle M,N\rangle=N:\tau}\ .$$

We denote with T' the collection of judgments provable in the λ-theory T extended with the constructors and rules given above.

(2) We now associate to T' a CCC $\mathbf{C}(T')$ as follows.

1. The objects are the types of the extended language.

2. The morphisms are equivalences classes of terms according to the equivalence induced by T'. More precisely

$$\mathbf{C}(\mathbf{T}')[\sigma,\tau]=\{[x:\sigma\vdash M:\tau]\mid x:\sigma\vdash M:\tau\}$$
$$[x:\sigma\vdash M:\tau]=\{y:\sigma\vdash N:\tau\ |\!\vdash\lambda x:\sigma.M=\lambda y:\sigma.N:\sigma\to\tau\in T'\}\ .$$

3. The structure associated to every CCC is defined as follows:

(id)	$[x:\sigma\vdash x:\sigma]$
$(comp)$	$[x:\tau\vdash M:\rho]\circ[y:\sigma\vdash N:\tau]=[y:\sigma\vdash M[N/x]:\rho]$
$(term)$	$!_\sigma=[x:\sigma\vdash *:1]$
$(proj)$	$\pi_1=[x:\sigma\times\tau\vdash\pi_1 x:\sigma]\ \ \pi_2=[x:\sigma\times\tau\vdash\pi_2 x:\tau]$
$(pair)$	$\langle[x:\rho\vdash M:\sigma],[x:\rho\vdash N:\tau]\rangle=[x:\rho\vdash\langle M,N\rangle:\sigma\times\tau]$
$(eval)$	$ev_{\sigma,\tau}=[x:(\sigma\to\tau)\times\sigma\vdash(\pi_1 x)(\pi_2 x):\tau]$
$(curry)$	$\Lambda([x:\rho\times\sigma\vdash M:\tau])=[y:\rho\vdash\lambda z:\sigma.M[\langle y,z\rangle/x]:\sigma\to\tau]$.

4. We leave to the reader the verification of the equations associated to a CCC.

(3) Finally we have to verify that the λ-theory associated to $\mathbf{C}(T')$ is exactly T'. To this end one checks that $[\![x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash M:\sigma]\!]=[x:\tau\vdash M[\pi_{n,i}x/x]:\sigma]$, where $\tau\equiv(\cdots(1\times\sigma_1)\times\cdots\times\sigma_n)$.

We can summarize our constructions as follows.

Theorem 4.4.4 (from λ-theories to CCC's) *Given any λ-theory T over the simply typed calculus with products and terminal object we can build a CCC $\mathbf{C}(T)$ such that the λ-theory associated to $\mathbf{C}(T)$ coincides with T.*

Remark 4.4.5 *(1) It is possible to see the constructions described here as representing an equivalence between a* category of CCC's *and a* category of λ-theories *[LS86].*

(2) It is possible to strengthen the previous theorem by considering a theory T over the simply typed λ-calculus without *products and terminal object. Then one needs to show that it is possible to add* conservatively *to T the equations $(*)$, (π_1), (π_2), and (SP) (see [Cur86], chapter 1, for a description of suitable proof techniques).*

4.5 Logical Relations

Logical relations are a quite ubiquitous tool in semantics and in logic. They are useful to establish links between syntax and semantics. In this section, logical relations are defined and applied to the proof of three results of this sort: Friedman's completeness theorem [Fri73], which characterizes $\beta\eta$-equality, and Jung-Tiuryn's and Siebers's theorems [JT93, Sie92] on the characterization of λ-definability.

Logical relations are predicates defined by induction over types, relating models of a given λ-calculus with constants $\Lambda(C)$. To simplify matters, throughout the rest of this section, we make the assumption that there is only one basic type κ. We define next (binary) logical relations, to this end we fix some terminology. Recall that an interpretation of simply typed λ-calculus in a CCC $\mathbf{C}$ is given as soon as the basic type κ is interpreted by an object D^κ of $\mathbf{C}$. We shall summarize this by calling the pair $\mathcal{M} = (\mathbf{C}, D^\kappa)$ a model. We write $[\![\sigma]\!] = D^\sigma$, hence $D^{\sigma\to\tau} = D^\sigma \Rightarrow D^\tau$.

If there are constants, then the constants must also be interpreted, but we leave this implicit to keep notation compact.

We shall make repeated use of the hom-sets of the form $\mathbf{C}[1, D]$. It is thus convenient to use a shorter notation. We shall write, for any object D of $\mathbf{C}$:

$$\mathbf{C}[1, D] = \underline{D} \ .$$

As a last preliminary to our definition, we point out the following instrumental isomorphisms which hold in any cartesian closed category, for any objects A and B:

$$\mathbf{C}[A, B] \cong \mathbf{C}[1, B^A] \ .$$

Here is the right-to-left direction (the other is left to the reader):

$$\hat{f} = \Lambda(f \circ \pi_2) \ .$$

Definition 4.5.1 (logical relation) *Let $\mathcal{M} = (\mathbf{C}, D^\kappa)$ and $\mathcal{M}' = (\mathbf{C}', D'^\kappa)$ be two models. A logical relation is a relation $R^\kappa \subseteq \underline{D^\kappa} \times \underline{D'^\kappa}$. These relations are extended to all types (including product types) by the following definitional equivalences:*

$$\begin{array}{rcl}
\mathcal{R}^1 & = & \{(id, id)\} \\
\langle d, e\rangle\, \mathcal{R}^{\sigma\times\tau}\, \langle d', e'\rangle & \Leftrightarrow & (d\,\mathcal{R}^\sigma\, d' \text{ and } e\,\mathcal{R}^\tau\, e') \\
f\,\mathcal{R}^{\sigma\to\tau}\, f' & \Leftrightarrow & \forall\, d, d'\ (d\,\mathcal{R}^\sigma\, d' \Rightarrow (ev \circ \langle f, d\rangle)\,\mathcal{R}^\tau\,(ev \circ \langle f', d'\rangle)) \ .
\end{array}$$

Thus, at every type, $R^\sigma \subseteq \underline{D^\sigma} \times \underline{D'^\sigma}$. We shall write, for any $f, f' : D^\sigma \to D^\tau$:

$$f\,\mathcal{R}^{\sigma,\tau}\, f' \quad \textit{whenever} \quad \hat{f}\,\mathcal{R}^{\sigma\to\tau}\,\hat{f'} \ .$$

A logical relation is called C-logical if $[\![c]\!]_{\mathcal{M}}\,\mathcal{R}^{\sigma}\,[\![c]\!]_{\mathcal{M}'}$, for all constants (where σ is the type of c). Notice that if C is empty, i.e. if our language is the simply typed λ-calculus without constants, then C-logical is the same as logical.

Remark 4.5.2 *The above definition is much in the spirit of the computability predicates used in section 3.5. The only difference is that computability predicates are defined on terms, while logical relations are defined on elements of models.*

The following is known as the fundamental lemma of logical relations.

Lemma 4.5.3 *Let R be a C-logical relation, then, for any closed term M of type σ:*

$$[\![\vdash M]\!]_{\mathcal{M}}\,\mathcal{R}^{\sigma}\,[\![\vdash M]\!]_{\mathcal{M}'}\ .$$

PROOF HINT. We extend the statement to open terms as follows. For any $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, and for any $d_1\,\mathcal{R}^{\sigma_1}\,d'_1, \ldots, d_n\,\mathcal{R}^{\sigma_n}\,d'_n$:

$$[\![M]\!] \circ \langle d_1, \ldots, d_n\rangle \quad \mathcal{R}^{\tau} \quad [\![M]\!] \circ \langle d'_1, \ldots, d'_n\rangle\ .$$

The proof of this extended statement is by induction on the size of M. For the abstraction case, one uses the following equation, which is consequence of the equations characterizing CCC's (cf. exercise 4.2.8): $ev\circ\langle\Lambda(f)\circ d, e\rangle = f\circ\langle d, e\rangle$. $\Box$

A more concrete description of models, and of logical relations, can be given when the models are extensional, i.e. when the morphisms of the model can be viewed as functions.

Definition 4.5.4 (enough points) *A category $\mathbf{C}$ with terminal object 1 is said to have enough points if the following holds, for any a, b and any $f, g \in \mathbf{C}[a, b]$:*

$$\forall\, h : 1 \to a\ \ (f \circ h = g \circ h)\ \ \Rightarrow\ \ f = g\ .$$

If the underlying category of a model $\mathcal{M}$ has enough points, we say that $\mathcal{M}$ is extensional.

Extensional models and logical relations can be described without using the vocabulary of category theory. The price to pay is that the definition is syntax-dependent. We leave it to the reader to convince himself that the following constructions indeed define (all) extensional models and logical relations.

A simply-typed extensional applicative structure (cf. section 3.1) $\mathcal{M}$ consists of a collection of sets D^{σ}, such that, for all σ, τ, $D^{\sigma\to\tau}$ is (in bijection with) a set of functions from D^{σ} to D^{τ}.

With any sequence $\sigma_1, \ldots, \sigma_n$ of types we associate a set $D^{\sigma_1,\ldots,\sigma_n}$ (abbreviated as $D^{\vec{\sigma}}$) as follows. With the empty sequence we associate a singleton set $\{*\}$, and we define $D^{\vec{\sigma},\tau} = D^{\vec{\sigma}} \times D^{\tau}$.

An interpretation function is a function $[\![\_]\!]$ mapping any typing judgement $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ to a function from $D^{\vec{\sigma}}$ to D^τ (for closed terms $\vdash M : \tau$, we consider freely $[\![\vdash M : \tau]\!]$ as an element of D^τ), which has to satisfy the following (universally quantified) properties:

$$\begin{array}{ll}
[\![\vdash c : \sigma]\!] & \in D^\sigma \\
[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash x_i : \sigma_i]\!](\vec{d}) & = d_i \\
[\![\Gamma \vdash MN : \tau]\!](\vec{d}) & = ([\![\Gamma \vdash M : \sigma \to \tau]\!](\vec{d}))([\![\Gamma \vdash N : \sigma]\!](\vec{d})) \\
([\![\Gamma \vdash \lambda x : \sigma .\, M : \sigma \to \tau]\!](\vec{d}))(e) & = [\![\Gamma, x : \sigma \vdash M : \tau]\!](\vec{d}, e) \ .
\end{array}$$

There is an additional condition dealing with weakening, which we omit here, and which serves in the proof of theorem 4.5.13. These clauses characterize the interpretation function except for constants.

An extensional model $\mathcal{M}$ consists of an extensional applicative structure together with an interpretation function $[\![\_]\!]_{\mathcal{M}}$ (the subscript is omitted when clear from the context).

In this concrete framework, a logical relation is now a relation $R^\kappa \subseteq D^\kappa \times D'^\kappa$, extended to all types (using function types only) by the following definitional equivalence:

$$f \, \mathcal{R}^{\sigma \to \tau} \, f' \quad \Leftrightarrow \quad \forall \, d, d' \ \ (d \, \mathcal{R}^\sigma \, d') \Rightarrow (f(d) \, \mathcal{R}^\tau \, f(d')) \ .$$

We shall freely use this concrete presentation in the sequel, when dealing with extensional models.

Exercise 4.5.5 *Establish formal links between the above formulations and categories with enough points. Hint: given a model described concretely, consider the category whose objects are sequences* $(\sigma_1, \ldots, \sigma_m)$ *of types and whose morphisms are vectors* $(d_1, \ldots, d_n) : (\sigma_1, \ldots, \sigma_m) \to (\tau_1, \ldots, \tau_n)$ *where* $d_i \in D^{\sigma_1 \to \cdots \to \sigma_m \to \tau_i}$ *for all* i*.*

Exercise 4.5.6 (extensional collapse) *Let* $\mathcal{M} = (\mathbf{C}, D^\kappa)$ *be a model, and consider the logical relations* R *defined by* $R^\kappa = \{(d, d) \mid d \in \underline{D^\kappa}\}$*. Consider the category* $[\mathbf{C}]$ *whose objects are the types built freely from* κ *using finite products and function types, and whose arrows from* σ *to* τ *are equivalence classes with respect to* $R^{\sigma,\tau}$*. Show that* $[\mathbf{C}]$ *is a CCC with enough points, called the extensional collapse of* $\mathbf{C}$*.*

We next give two applications of logical relations. The first application is in fact itself a family of applications. Logical relations may be useful to prove inclusions of theories, thanks to the following lemma.

Lemma 4.5.7 *Let* R *be a logical relation between two models* $\mathcal{M}$ *and* $\mathcal{M}'$*. Suppose that* R^σ *is functional for all* σ*, i.e. for all* d, d', d''*:*

$$d \, \mathcal{R}^\sigma \, d' \text{ and } d \, \mathcal{R}^\sigma \, d'' \quad \Rightarrow \quad d' = d'' \ .$$

Then, for any closed terms M, N *of the same type:*

$$[\![M]\!]_{\mathcal{M}} = [\![N]\!]_{\mathcal{M}} \quad \Rightarrow \quad [\![M]\!]_{\mathcal{M}'} = [\![N]\!]_{\mathcal{M}'} \ .$$

PROOF. Let σ be the common type of M, N and let d be the common value of $[\![M]\!]_{\mathcal{M}}$ and $[\![N]\!]_{\mathcal{M}}$. By two applications of lemma 4.5.3 we have $d\,\mathcal{R}^\sigma\,[\![M]\!]_{\mathcal{M}'}$ and $d\,\mathcal{R}^\sigma\,[\![N]\!]_{\mathcal{M}'}$, and the conclusion follows from the assumption that R is functional. $\square$

We would like to have sufficient conditions on R^κ to prove that R is functional. Clearly, one should require R^κ to be functional to start with. In practice R^κ may be more than functional. Often we want to compare (extensional) models which interpret basic types the same way and which differ in the choice of the functions (or morphisms) in the interpretation of function types. In other words, R^κ is often the identity (an example is provided by exercise 4.5.6). It turns out that surjectivity (a property a fortiori enjoyed by the identity) is a useful property to carry along an induction. Indeed, let us attempt to prove that if R^κ is functional and surjective, then R^σ is functional and surjective for all σ.

Once we know that R^σ is functional, we freely say that $R^\sigma(d)$ is defined and equal to d' if $d\,\mathcal{R}^\sigma\,d'$. Also, we assume that we have proved surjectivity at type σ by building a function $i^\sigma : D'^\sigma \to D^\sigma$ such that $i^\sigma(d')\mathcal{R}^\sigma d'$ for all d'. Equivalently, we assume that at type σ, there exists a partial function $R^\sigma : D^\sigma \rightharpoonup D'^\sigma$ and a total function $i^\sigma : D'^\sigma \to D^\sigma$ such that $R^\sigma \circ i^\sigma = id$. Similarly, we suppose that these data exist at type τ. We want to show that $R^{\sigma\to\tau}$ is functional, and to construct $i^{\sigma\to\tau}$ such that $R^{\sigma\to\tau} \circ i^{\sigma\to\tau} = id$.

- Functional: Using the formulation of R^σ and R^τ as partial functions, the definition of $f\,\mathcal{R}^{\sigma\to\tau}\,f'$ reads:

  $$\forall d \in D^\sigma \quad R^\sigma(d)\downarrow \Rightarrow f'(R^\sigma(d)) = R^\tau(f(d)) \ .$$

  Suppose that $f\,R^{\sigma\to\tau}\,f'$. Then, for any $d' \in D'^\sigma$:

  $$\begin{array}{rcll} f'(d') & = & f'(R^\sigma(i^\sigma(d'))) & \text{by surjectivity at } \sigma \\ & = & R^\tau(f(i^\sigma(d'))) & \text{by the definition of } R^{\sigma\to\tau} \ . \end{array}$$

  Hence f' is unique. More precisely:

  $$R^{\sigma\to\tau}(f)\downarrow \ \Rightarrow \ R^{\sigma\to\tau}(f) = R^\tau \circ f \circ i^\sigma \ .$$

- Surjective: We claim that all we need to know about $i^{\sigma\to\tau}$ is the following, for any $f' \in D'^{\sigma\to\tau}$:

  $$(\dagger) \quad \forall d \in D^\sigma \quad R^\sigma(d)\downarrow \Rightarrow i^{\sigma\to\tau}(f')(d) = i^\tau(f'(R^\sigma(d))) \ .$$

  Let us prove that if $i^{\sigma\to\tau}$ satisfies $(\dagger)$, then $R^{\sigma\to\tau} \circ i^{\sigma\to\tau} = id$, that is, by the definition of $R^{\sigma\to\tau}$, for any $f' : D'^{\sigma\to\tau}$:

  $$\forall d \in D^\sigma \quad R^\sigma(d)\downarrow \Rightarrow f'(R^\sigma(d)) = R^\tau(i^{\sigma\to\tau}(f')(d)) \ .$$

Indeed, we have, under the assumption that $R^\sigma(d)\downarrow$:

$$\begin{array}{rcll} R^\tau(i^{\sigma\to\tau}(f')(d)) & = & R^\tau(i^\tau(f'(R^\sigma(d)))) & \text{by } (\dagger) \\ & = & f'(R^\sigma(d)) & \text{by surjectivity at } \tau\ . \end{array}$$

In the above proof schema, we have used extensionality of both $\mathcal{M}$ and $\mathcal{N}$. The proof schema is easily instantiated in the following theorem.

Theorem 4.5.8 (Friedman) *Let $\mathcal{F} = \{\mathcal{D}^\sigma\}$ be the full type hierarchy over an infinite set D^κ, i.e. $\mathcal{F} = \{\mathcal{D}^\sigma\}$ is the extensional model for which $D^{\sigma\to\tau}$ is the set of all functions from D^σ to D^τ. Then, for any λ-terms:*

$$[\![M]\!]_{\mathcal{F}} = [\![N]\!]_{\mathcal{F}} \quad\Leftrightarrow\quad M =_{\beta\eta} N\ .$$

PROOF. ($\Leftarrow$) Since $\mathcal{F}$ is extensional, it validates $\beta\eta$.

($\Rightarrow$) We turn syntax into an extensional model, by setting D'^σ to be the set of all $\beta\eta$ equivalence classes of (open) terms of type σ. That this makes an extensional model is proved as follows. We define an application function $\bullet$ by $[M]{\bullet}[N] = [MN]$. Suppose that for all $[P]$, $[M]{\bullet}[P] = [N]{\bullet}[P]$. Then in particular, for a fresh x, $[Mx] = [Nx]$, i.e. $Mx =_{\beta\eta} Nx$. Then:

$$M =_{\beta\eta} \lambda x.Mx =_{\beta\eta} \lambda x.Nx =_{\beta\eta} N\ .$$

Therefore we have an extensional model. Now we are ready to use the proof schema, with the term model as $\mathcal{M}'$, and $\mathcal{F}$ as $\mathcal{M}$. We need a surjection from D^κ to D'^κ. It is clearly enough to live with a denumerable set of variable names. Therefore we can consider the set of terms as denumerable. The sets D'^σ are then at most denumerable. They are not finite, since we can have infinitely many different $\beta\eta$-normal forms x, $xx_1, xx_1\cdots x_n, \ldots$ at any given type σ. In particular, D'^κ is infinite and denumerable. Then, given any infinite D^κ, we can pick a (total) surjection $R^\kappa : D^\kappa \to D'^\kappa$. More precisely, we can pick a function $i^\kappa : D'^\kappa \to D^\kappa$ such $R^\kappa \circ i^\kappa = id$. We are left to exhibit a definition of $i^{\sigma\to\tau}$ satisfying ($\dagger$). But property ($\dagger$) actually gives a definition of the restriction of $i^{\sigma\to\tau}(f')$ to the domain of R^σ. Since we are in the full type hierarchy, we can choose any value for $i^{\sigma\to\tau}(f')(d)$ when $R^\sigma(d)\uparrow$. □

The above proof, which is essentially the original proof of [Fri73], uses the fullness of model $\mathcal{F}$ quite crudely. There exists another proof of Friedman's theorem, as a corollary of a powerful theorem known as Statman's 1-section theorem, whose range of applications to various completeness theorems seems wider than what can be achieved by the above proof schema.

Statman's theorem states that in order to prove a completeness theorem, it suffices to prove it for terms of type $\alpha = (\kappa \to \kappa \to \kappa) \to \kappa \to \kappa$. What is special about this type? If there are no constants, the closed normal forms of type α are exactly the (encodings of) binary trees constructed over a signature $\{f, c\}$ consisting of a symbol f or arity 2 and a symbol c of arity 0.

Theorem 4.5.9 (1-section) *Let $\mathcal{C}$ be a class of models of the simply-typed λ-calculus. The following properties are equivalent:*

(1) *For any type σ, and any closed terms M, N of type σ:*

$$\forall \mathcal{M} \in \mathcal{C} \;\; [\![\vdash M]\!]_{\mathcal{M}} = [\![\vdash N]\!]_{\mathcal{M}} \quad \Leftrightarrow \quad M =_{\beta\eta} N \; .$$

(2) *For any closed terms M, N of type $\alpha = (\kappa \to \kappa \to \kappa) \to \kappa \to \kappa$:*

$$\forall \mathcal{M} \in \mathcal{C} \;\; [\![\vdash M]\!]_{\mathcal{M}} = [\![\vdash N]\!]_{\mathcal{M}} \quad \Leftrightarrow \quad M =_{\beta\eta} N \; .$$

Proof. Statement (1) obviously implies (2). The other direction is an immediate consequence of the following lemma 4.5.11. □

Definition 4.5.10 (rank) *The rank of a simple type τ is defined as follows:*

$$\begin{array}{l} rank(\kappa) = 0 \quad (\kappa \textit{ base type}) \\ rank(\tau_1 \to \cdots \to \tau_k \to \kappa) = 1 + (\max\{rank(\tau_i) \mid i = 1 \ldots k\}) \; . \end{array}$$

We say that a type τ is at most rank n if $rank(\tau) \leq n$. If there is only one base type κ, the types of rank at most 1 are thus the types κ^n, where:

$$\kappa^0 = \kappa \quad \kappa^{n+1} = \kappa \to \kappa^n \; .$$

Lemma 4.5.11 *If M, N are closed simply-typed λ-terms of type σ, and if $M \neq_{\beta\eta} N$, then there exists a closed simply-typed λ-term P of type $\sigma \to \alpha$ (where α is as in the statement of theorem 4.5.9) such that $PM \neq_{\beta\eta} PN$.*

Proof hint. The proof makes use of the notion of extended $\beta\eta$-normal form. An extended $\beta\eta$-normal form of type $\sigma_1 \to \cdots \to \sigma_n \to \kappa$ is a term of the form $\lambda x_1 \cdots x_n . u(M_1 x_1 \cdots x_n) \cdots (M_k x_1 \cdots x_n)$, where each M_j is itself an extended $\beta\eta$-normal form. Note that extended $\beta\eta$-normal forms are not normal forms in general, whence the qualification "extended". Using strong $\beta\eta$ normalisation, it is easily shown that for any M there exists a unique extended $\beta\eta$-normal form N such that $M =_{\beta\eta} N$. The proof of the statement is decomposed in two claims.

1. Let M, N be two different closed extended $\beta\eta$-normal forms of a type σ of rank at most 2. Then there exists a closed term L of type $\sigma \to \alpha$ depending only on σ, such that $LM \neq_{\beta\eta} LN$.

2. Let M, N be two different closed extended $\beta\eta$-normal forms of type $\sigma_1 \to \cdots \to \sigma_n \to \kappa$. Then there exist terms $V_1, \ldots, V_n$, whose free variables have types of rank at most 1, such that $MV_1 \cdots V_n \neq_{\beta\eta} NV_1 \cdots V_n$.

The statement follows from these claims, taking:

$$P = \lambda x.L(\lambda \vec{y}.xV_1 \cdots V_n)$$

where $\vec{y}$ is a list of the free variables in $V_1, \ldots, V_n$, and where L is relative to $\lambda \vec{y}.MV_1, \ldots V_n$ and $\lambda \vec{y}.NV_1, \ldots V_n$.

Both claims are proved by induction on the size of M, N. For claim 1 let us fix $\sigma = \sigma_1 \to \cdots \to \sigma_n \to \kappa$. We pick free variables $x : \kappa$ and $g : \kappa \to \kappa \to \kappa$ and define the following terms:

$$\mathbf{0} = x \qquad (\mathbf{i}+\mathbf{1}) = (gx\mathbf{i}) \ .$$

Suppose, say, that $\sigma_i = \kappa \to \kappa \to \kappa$. Then we set:

$$P_i = \lambda y_1 y_2.g\mathbf{i}(gy_1y_2) \ .$$

Finally, we take:

$$L = \lambda w.\lambda g.\lambda x.wP_1 \cdots P_n$$

which depends on σ only. Suppose, say, that $M = \lambda \vec{x}.\, x_i(M_1\vec{x})(M_2\vec{x})$. Then:

$$LM =_{\beta\eta} \lambda gx.g\mathbf{i}(g(M_1\vec{P})(M_2\vec{P})) \ .$$

If N has a different head variable, say x_j, then similarly LN is $\beta\eta$-equal to a term which starts with $\lambda gx.g\mathbf{j}$, which cannot be $\beta\eta$ equal to a term starting with $\lambda gx.g\mathbf{i}$ (these prefixes are preserved until the normal form is reached). Suppose thus that N has the same head variable, i.e. $N = \lambda \vec{x}.\, x_i(N_1\vec{x})(N_2\vec{x})$. Since the type of x_i has rank 1, $M_1\vec{x}$ has type κ, i.e. M_1 has type σ. Similarly, M_2, N_1, and N_2 have type σ. On the other hand $M \neq_{\beta\eta} N$ implies, say, $M_1 \neq_{\beta\eta} N_1$. We can thus apply induction to M_1 and N_1:

$$M_1\vec{P} =_{\beta\eta} LM_1gx \neq_{\beta\eta} LN_1gx =_{\beta\eta} N_1\vec{P} \ .$$

On the other hand, since:

$$LN =_{\beta\eta} \lambda gx.g\mathbf{i}(g(N_1\vec{P})(N_2\vec{P}))$$

$LM =_{\beta\eta} LN$ would imply $M_1\vec{P} =_{\beta\eta} N_1\vec{P}$ and $M_2\vec{P} =_{\beta\eta} N_2\vec{P}$. Contradiction.

We turn to the second claim. The interesting case is again when M and N have the same head variable. We suppose that M, N, and σ_i are as above. Now $M_1\vec{x}$ is not necessarily of base type. By induction applied to M_1 and N_1, there is a vector $\vec{U}$ of terms $U_1, \ldots, U_m$ such that $M_1\vec{U} \neq_{\beta\eta} N_1\vec{U}$ at type κ. In particular $m \geq n$, where n is the length of $\vec{x}$. We set (with h, y_1, y_2 fresh and of rank at most 1):

$$V_i = \lambda y_1 y_2.h(y_1 U_{n+1} \cdots U_m)(U_i y_1 y_2)$$

and $V_p = U_p$ if $p \neq i$ and $p \leq n$. The trick about V_i is the following property:

$$(\star) \quad V_i[\lambda uv.v/h] =_{\beta\eta} U_i$$

from which $(M_1\vec{V}U_{n+1}\cdots U_m)[\lambda uv.v/h] = M_1\vec{U}$ follows, and similarly for N_1. Therefore:

$$M_1\vec{V}U_{n+1}\cdots U_m \neq_{\beta\eta} N_1\vec{V}U_{n+1}\cdots U_m$$

which entails the claim, since $M\vec{V}$ reduces to a term which starts with

$$h(M_1\vec{V}U_{n+1}\cdots U_m)$$

and similarly for N_1. □

We now show how theorem 4.5.9 yields Friedman's theorem as a corollary. All we have to check is that if two trees built only using application from variables $g : \kappa \to \kappa \to \kappa$ and $c : \kappa$ are different, then their semantics in $\mathcal{F}$ are different. We assume more precisely that D^κ is ω, and we pick a pairing function *pair* which is injective and such that:

$$\forall m, n \ \ (m < pair(m,n) \text{ and } n < pair(m,n)) \ .$$

Then we interpret g by (the curried version of) *pair*, and c by any number, say 0. It is easy to prove by induction that if s, t are two distinct trees, then their interpretations are different, using these two properties of *pair*. This shows the hard implication in the equivalence stated in theorem 4.5.9, and completes thus our second proof of Friedman's theorem.

We now come back to logical relations. As a second kind of application, we consider the so-called definability problem. Given an extensional model $\mathcal{M}$, is it possible to characterize the elements d such that $d = [\![M]\!]$ for some closed M? A positive answer was given by Jung and Tiuryn [JT93]. It is an elegant construction, based on Kripke logical relations, which we define next (in the particular setting we need).

Definition 4.5.12 *A Kripke logical relation over an extensional model $\mathcal{M}$ is given by a family of sets $R^\kappa_{\vec{\sigma}} \subseteq (D^{\vec{\sigma}} \to D^\tau)$ satisfying the following so-called Kripke-monotonicity condition:*

$$f \in R^\kappa_{\vec{\sigma}} \ \ \Rightarrow \ \ (\forall \vec{\sigma'} \ f \circ \pi \in R^\kappa_{\vec{\sigma},\vec{\sigma'}}) \ .$$

where $\pi(\vec{d}, \vec{d'}) = \vec{d}$. A Kripke logical relation is extended to all types as follows:

$$f \in R^{\tau_1 \to \tau_2}_{\vec{\sigma}} \ \ \Leftrightarrow \ \ \forall \vec{\sigma'}, \ g \in R^{\tau_1}_{\vec{\sigma},\vec{\sigma'}} \ \ \lambda \vec{d}\vec{d'}. \ f(\vec{d})(g(\vec{d},\vec{d'})) \in R^{\tau_2}_{\vec{\sigma},\vec{\sigma'}} \ .$$

We write R^τ for $R^\tau_{\vec{\sigma}}$ when $\vec{\sigma}$ is of length 0. An element $d \in D^\sigma$ is called invariant under R if $d \in R^\sigma$. A Kripke C-logical relation is a Kripke logical relation such that $[\![c]\!]$ is invariant for all $c \in C$.

Each set $R^\tau_{\vec{\sigma}}$ can be viewed as a relation over D^τ whose arity is the cardinality of $D^{\vec{\sigma}}$. In this sense, Kripke logical relations are of variable arities. It is easy to check that Kripke-monotonicity extends to all types:

$$f \in R^\tau_{\vec{\sigma}} \;\Rightarrow\; (\forall \vec{\sigma'}\; f \circ \pi \in R^\tau_{\vec{\sigma},\vec{\sigma'}}) \; .$$

Theorem 4.5.13 *Let $\mathcal{M}$ be an extensional model. Then d is definable, i.e. $d = [\![M]\!]$ for some closed term M, if and only if d is invariant under all Kripke C-logical relations.*

PROOF. $(\Rightarrow)$ This is a variant of the fundamental lemma of logical relations. The statement needs to be extended to subscripts $\vec{\sigma}$ and to open terms. The following extended statement is proved straightforwardly by induction on M.

For any $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M : \tau$, for any $\vec{\sigma}$, for any $f_1 \in R^{\tau_1}_{\vec{\sigma}}, \ldots, f_n \in R^{\tau_n}_{\vec{\sigma}}$:

$$[\![M]\!] \circ \langle f_1, \ldots, f_n\rangle \in R^\tau_{\vec{\sigma}} \; .$$

$(\Leftarrow)$ We actually prove a stronger result. We exhibit a single Kripke logical relation which characterizes definability. This relation S is defined as follows:

$$S^\kappa_{\vec{\sigma}} = \{[\![M]\!] \mid \vec{x} : \vec{\sigma} \vdash M : \kappa\} \; .$$

The proof that S satisfies Kripke-monotonicity requires an additional assumption on the interpretation function in the concrete description of extensional model, which we detail now. If $\vec{x} : \vec{\sigma} \vdash M : \tau$ is a provable judgment, then $\vec{x} : \vec{\sigma}, \vec{x'} : \vec{\sigma'} \vdash M : \tau$ is also provable, for any $\vec{\sigma'}$. We require:

$$[\![\vec{x} : \vec{\sigma}, \vec{x'} : \vec{\sigma'} \vdash M : \tau]\!] = [\![\vec{x} : \vec{\sigma} \vdash M : \tau]\!] \circ \pi \; .$$

Kripke monotonicity follows straightforwardly. We next claim:

$$\forall \tau, \vec{\sigma}\;\; S^\tau_{\vec{\sigma}} = \{[\![M]\!] \mid \vec{x} : \vec{\sigma} \vdash M : \tau\} \; .$$

Then the statement follows, taking the empty $\vec{\sigma}$. We prove the two inclusions of the claim by mutual induction on the size of the type τ, for $\tau = \tau_1 \to \tau_2$:
$(\subseteq)$ By induction applied at τ_1 $(\supseteq)$, we have:

$$[\![\vec{x} : \vec{\sigma}, y : \tau_1 \vdash y : \tau_1]\!] \in S^{\tau_1}_{\vec{\sigma},\tau_1} \; .$$

Let $f \in S^{\tau_1 \to \tau_2}_{\vec{\sigma}}$. By definition of S at $\tau_1 \to \tau_2$, we have

$$\lambda \vec{d} d_1 .\, f(\vec{d})(d_1) \in S^{\tau_2}_{\vec{\sigma},\tau_1} \; .$$

By induction applied at τ_2 $(\subseteq)$, there exists $\vec{x} : \vec{\sigma}, y : \tau_1 \vdash M : \tau_2$ such that, for all $\vec{d}, d_1$:

$$[\![\vec{x} : \vec{\sigma}, y : \tau_1 \vdash M : \tau_2]\!](\vec{d}, d_1) = f(\vec{d})(d_1) \; .$$

It follows that $[\![\lambda x.M]\!] = f$.
$(\supseteq)$ Let $\vec{x} : \vec{\sigma} \vdash M : \tau_1 \to \tau_2$. Let $g \in S^{\tau_1}_{\vec{\sigma},\vec{\sigma'}}$. By induction applied at τ_1 $(\subseteq)$, there exists $\vec{x} : \vec{\sigma}, \vec{x'} : \vec{\sigma'} \vdash N : \tau_1$ such that $g = [\![N]\!]$. We have to prove:

$$[\![\vec{x} : \vec{\sigma}, \vec{x'} : \vec{\sigma'} \vdash MN : \tau_2]\!] \in S^{\tau_2}_{\vec{\sigma},\vec{\sigma'}}$$

which holds by induction applied at τ_2 $(\supseteq)$. □

If we restrict our attention to terms whose type has at most rank 2, and if we require that the constants in C also have a type of rank at most 1, then we can get around variable arities. The following theorem is due to Sieber. It was actually obtained before, and provided inspiration for, theorem 4.5.13. Given an extensional model $\mathcal{M} = \{D^\sigma\}$, and a function $f \in D^\sigma$, given a matrix $\{d_{ij}\}_{i \le p, j \le n}$, we wonder whether there exists a closed term M such that, for each line of the matrix, i.e., for all $i \le p$:

$$[\![M]\!](d_{i1}) \cdots (d_{in}) = f(d_{i1}) \cdots (d_{in})$$

and we consider to this aim C-logical relations of arity p, containing the columns of the matrix. If we can exhibit such a logical relation which moreover does not contain the vector $(f(d_{11}) \cdots (d_{1n}), \ldots, f(d_{p1}) \cdots (d_{pn}))$, then the answer to our question is negative, by the fundamental lemma. Sieber's theorem asserts that this method is complete.

Theorem 4.5.14 *Consider a set C of constants whose types have at most rank 1. Let $\mathcal{M}$ be an extensional model for $\Lambda(C)$, and let $\sigma = \sigma_1 \to \cdots \to \sigma_n \to \kappa$ be a type with rank at most 2. Let $\{d_{ij}\}_{i \le p, j \le n}$ be a matrix where $d_{ij} \in D^{\sigma_j}$, for any i, j, and let, for all $i \le p$:*

$$e_i = f(d_{i1}) \cdots (d_{in}) \ .$$

The following properties are equivalent:
(1) *There exists $\vdash M : \kappa$ such that, for all $i \le p$:*

$$[\![M]\!](d_{i1}) \cdots (d_{in}) = e_i \ .$$

(2) *For every p-ary C-logical relation R, the following implication holds:*

$$(\ddagger) \quad (\forall j \le n \ \ (d_{1j}, \ldots, d_{pj}) \in R^{\sigma_j}) \ \Rightarrow \ (e_1, \ldots, e_p) \in R^\kappa \ .$$

(3) *The logical relation S defined at κ by:*

$$S^\kappa = \{(a_1, \ldots, a_p) \mid \exists \vdash N : \kappa \ \ \forall i \le p \ \ [\![N]\!](d_{i1}) \cdots (d_{in}) = a_i\}$$

contains $(e_1, \ldots, e_p)$.

PROOF. (1) $\Rightarrow$ (2) This implication holds by the fundamental lemma.

(2) $\Rightarrow$ (3) Suppose that we have shown that S is C-logical. Then, by (2), S satisfies (‡). The conclusion will follow if we prove that S in fact satisfies the hypothesis of (‡). If N and $(a_1, \ldots, a_p)$ are as in the definition of S^κ, it is convenient to call N a witness of $(a_1, \ldots, a_p)$. If $(d_{1j}, \ldots, d_{pj})$ is at base type, then we take $\lambda\vec{x}.x_j$ as witness. If $(d_{1j}, \ldots, d_{pj})$ have types of rank at most 1, say κ^2, we have to check, for any $(a_1, \ldots, a_p), (b_1, \ldots, b_p) \in S^\kappa$:

$$(d_{1j}(a_1)(b_1), \ldots, d_{pj}(a_p)(b_p)) \in S^\kappa \ .$$

Since $\vec{a}$ and $\vec{b}$ are at base type (this is where we use the restriction on types of rank 2 in the statement), by definition of S^κ there are witnesses N and P for them. Then $\lambda\vec{x}.x_j(N\vec{x})(P\vec{x})$ is a witness for $(d_{1j}(a_1)(b_1), \ldots d_{pj}(a_p)(b_p))$.

The argument to show that S is C-logical is similar to the argument just used (left to the reader).

(3) $\Rightarrow$ (1) Obvious by definition of S. □

If the base domain D^κ is finite, then all the D^σ's are finite, and the complete behaviour of f can be described by a matrix. By the characterization (2) of theorem 4.5.14, the definability problem is then decidable at rank at most 2: try successively all C-logical relations R (there are finitely many of them).

The following proposition, due to Stoughton [Sto94], paves the way towards a more realistic decision procedure. We call intersection of two logical relations S_1 and S_2 the relation S_3 defined at κ by:

$$S_3^\kappa = S_1^\kappa \cap S_2^\kappa \ .$$

We write $S_3 = S_1 \cap S_2$. We can similarly define an arbitrary intersection of logical relations. When C consists of constants whose types have at most rank 1, then any intersection of C-logical relations is C-logical.

Proposition 4.5.15 *The relation S in theorem 4.5.14(3), is the intersection of all the C-logical relations satisfying $\forall j \leq n\, (d_{1j}, \ldots, d_{pj}) \in R^{\sigma^j}$.*

PROOF. Let S' be the intersection mentioned in the statement. We have $S'^\kappa \subseteq S^\kappa$. Conversely, if $(a_1, \ldots, a_p) \in S^\kappa$, then, by definition of S, there exists an N such that $\forall\, i \leq p\ [\![\vdash N]\!](d_{i1}) \cdots (d_{in}) = a_i$, which implies $(a_1, \ldots, a_p) \in S^\kappa$ by the fundamental lemma, and by the definition of S'. □

By this proposition, it is enough to construct progressively S', starting from the assumptions that S' contains the vectors $(d_{1j}, \ldots, d_{pj})$ $(j \leq n)$. If the construction finishes without having met $\vec{e}$, then there is no M for f and $\{d_{ij}\}_{i \leq p, j \leq n}$. We illustrate this with an example which will be important in the sequel.

Very Finitary PCF. We set $C = \{\bot, \top, \mathit{if}\ \_\ \mathit{then}\ \_\}$. We set $D^\kappa = \mathbf{O}$, the flat domain $\{\bot, \top\}$, and we build the model in the category of partial orders and monotonic functions. The meanings of $\vdash \bot : \kappa$ and $\vdash \top : \kappa$ are $\bot \in D^\kappa$ and $\top \in D^\kappa$. The last constant is a unary test function:

$$\mathit{if}\ \bot\ \mathit{then}\ d = \bot \qquad \mathit{if}\ \top\ \mathit{then}\ d = d \quad .$$

This language is known as Very Finitary PCF (Finitary PCF is coming next, and PCF is the subject of chapter 6). We claim that there is no closed term M in this language such that:

$$\begin{array}{l} [\![M]\!](\bot)(\top) = \top \\ [\![M]\!](\top)(\bot) = \top \\ [\![M]\!](\bot)(\bot) = \bot \ . \end{array}$$

We begin the construction of S'. It should contain the two columns $(\bot, \top, \bot)$ and $(\top, \bot, \bot)$. Since it has to be a C-logical relation, it also has to contain $(\bot, \bot, \bot)$ and $(\top, \top, \top)$. It is easy to see that this set of pairs makes *if* _ *then* _ invariant. We have completed the definition of S'. Since S' does not contain $(\top, \top, \bot)$, there is no M meeting the above specification.

On the other hand, if the decision procedure yields a positive answer, it would be nice to produce the defining term as well. Here is an indication of how this may be done. We refer to [Sto94] for details. We now consider a function $F : (\kappa \to \kappa \to \kappa) \to \kappa$ such that:

$$F(g_1) = \top \quad F(g_2) = \top \quad F(\bot) = \bot$$

where g_1 is a function such that $g_1(\top)(\bot) = \top$, g_2 is a function such that $g_2(\bot)(\top) = \top$, and $\bot$ is the constant $\bot$ function. We exhibit a closed term M such that:

$$[\![M]\!](g_1) = \top \quad [\![M]\!](g_2) = \top \quad [\![M]\!](\bot) = \bot \ .$$

We begin the construction of S' as we did in the previous example, but now we build pairs $(\vec{d}, P)$, where $\vec{d}$ is a member of S', and where P is a term. We start with $((\bot, \bot, \bot), \bot)$, $((\top, \top, \top), \top)$, and $((g_1, g_2, \bot), g)$, where g is a variable of type $\kappa \to \kappa \to \kappa$. By definition of a logical relation, S' has to contain:

$$(g_1(\top)(\bot), g_2(\top)(\bot), \bot(\top)(\bot)) = (\top, \bot, \bot) \ .$$

We form the pair $((\top, \bot, \bot), g(\top)(\bot))$ to keep track of how we obtained this new vector. Similarly, we obtain $((\bot, \top, \bot), g(\bot)(\top))$. Finally, taking these two new vectors as arguments for g, we build the vector which we are seeking:

$$((\top, \top, \bot), g(g(\top)(\bot))(g(\bot)(\top))) \ .$$

The term $M = \lambda g.g(g(\top)(\bot))(g(\bot)(\top))$ satisfies the requirements, by construction.

Finitary PCF. The same counter-example and example also live in the following language, called Finitary PCF. We now set $C = \{\bot, tt, f\!f, if\ \_\ then\ \_\ else\ \_\}$. We set $D^\kappa = \mathbf{B}_\bot$, the flat domain $\{\bot, tt, f\!f\}$. Again, we build the model in the category of partial orders and monotonic functions. The meanings of $\vdash \bot : \kappa$, $\vdash tt : \kappa$, and $\vdash f\!f : \kappa$ are $\bot \in D^\kappa$, $tt \in D^\kappa$, and $f\!f \in D^\kappa$. The last constant is the usual conditional:

$$if\ \bot\ then\ d\ else\ e = \bot \quad if\ tt\ then\ d\ else\ e = d \quad if\ f\!f\ then\ d\ else\ e = e\ .$$

The advantage of Finitary PCF (and of its associated model) over Very Finitary PCF is that it allows for an elegant characterization of the C-logical relations (also due to Sieber), which we give now.

Definition 4.5.16 ((Sieber-)sequential) *The C-logical relations for Finitary* PCF *and its monotonic function model are called sequential relations. Consider the following n-ary relations $S^n_{A,B}$ over $\mathbf{B}_\bot$, where $A \subseteq B \subseteq \{1, \ldots, n\}$:*

$$(d_1, \ldots, d_n) \in S^n_{A,B} \quad \Leftrightarrow \quad (\exists\, i \in A\ \ d_i = \bot)\ or\ (\forall\, i, j \in B\ \ d_i = d_j)\ .$$

A Sieber-sequential relation is an n-ary logical relation S such that S^κ is an intersection of relations of the form $S^n_{A,B}$.

The word "sequential" will be justified in section 6.5.

Theorem 4.5.17 *A logical relation over Finitary* PCF *is sequential if and only if it is Sieber-sequential.*

($\Leftarrow$) All base type constants are invariant, since constant vectors satisfy trivially the second disjunct in the definition of $S^n_{A,B}$. We check only that the conditional is invariant. If its first argument $d = (d_1, \ldots, d_n)$ satisfies the first disjunct, or the second disjunct with the common value equal to $\bot$, then the result vector $e = (e_1, \ldots, e_n)$ satisfies the same property by strictness of the conditional function. Otherwise, if, say, $d_i = 0$ for all $i \in B$, then the coordinates e_i for $i \in B$ are those of the second argument. Since $A \subseteq B$, this entails $e \in S^n_{A,B}$.

($\Rightarrow$) Let R be n-ary and sequential, and let S be the intersection of all $S^n_{A,B}$'s containing R^κ. We prove $S \subseteq R^\kappa$. We pick $d = (d_1, \ldots, d_n) \in S$, and we show, by induction on the size of $C \subseteq \{1, 2, \ldots, n\}$:

$$\exists\, e = (e_1, \ldots, e_n) \in R^\kappa \quad (\forall\, i \in C\ \ e_i = d_i)\ .$$

If $C = \{i\}$ is a singleton, then we choose $e = (d_i, \ldots, d_i)$. We suppose now that $\sharp C \geq 2$, and we distinguish cases.

1. $d_i = \bot$ for some $i \in C$:

(a) $R^\kappa \subseteq S^n_{C\backslash\{i\},C}$: Then $d \in S^n_{C\backslash\{i\},C}$, by definition of S, and either $d_k = \perp$ for some $k \in C\backslash\{i\}$, or all d_j's are equal, for $j \in C$. Since $d_i = \perp$, the latter case is rephrased as: all d_j's are $\perp$. Recalling that $\sharp C \geq 2$, we have thus, in either case:

$$\exists k \in C\backslash\{i\} \;\; d_k = \perp \; .$$

We apply induction to both $C\backslash\{i\}$ and $C\backslash\{k\}$, obtaining $v \in R^\kappa$ and $w \in R^\kappa$, respectively. There are again two cases:

i. If $v_i = d_i (= \perp)$, then v works for C.

ii. If $v_i \neq \perp$, say, $v_i = tt$, consider the term:

$$M = \lambda xy.\ \mathit{if}\ x\ \mathit{then}\ y\ \mathit{else}\ x$$

and set:

$$f = [\![M]\!] \quad \text{and} \quad e = (f(v_1)(w_1), \ldots, f(v_n)(w_n)) \; .$$

First, $e \in R^\kappa$ by sequentiality of R. Second, we show that e coincides with d over C. By definition of M, for any x, y, we have $f(x)(y) = x$ or $f(x)(y) = y$. This implies that e does the job over $C\backslash\{i,k\}$, over which v and w coincide. Since v coincides with d over $C\backslash\{i\}$, we have $v_k = d_k = \perp$, hence $f(v_k)(w_k) = \perp = d_k$, since f is strict in its first argument. Finally, since $v_i = tt$, we have $f(v_i)(w_i) = w_i = d_i$.

(b) $R^\kappa \not\subseteq S^n_{C\backslash\{i\},C}$: Let $u \in R^\kappa \backslash S^n_{C\backslash\{i\},C}$. By definition of $S^n_{C\backslash\{i\},C}$, there exists $k \in C$ such that

$$\begin{array}{ll} u_k \neq u_i & (\text{negation of } \forall j,k \in C \;\; u_j = u_k) \\ u_k \neq \perp & (\text{negation of } \exists j \in C\backslash\{i\} \;\; u_j = \perp) \; . \end{array}$$

We suppose, say, that $u_k = tt$. Let, as in the previous case, v and w be relative to $C\backslash\{i\}$ and $C\backslash\{k\}$, respectively. We now consider the term $N = \lambda xyz.\ \mathit{if}\ x\ \mathit{then}\ y\ \mathit{else}\ z$, and we set:

$$g = [\![N]\!] \quad \text{and} \quad e = (g(u_1)(v_1)(w_1), \ldots, g(u_n)(v_n)(w_n)) \; .$$

We check that e works for C. Let $j \in C\backslash\{i\}$, since $u \notin S^n_{C\backslash\{i\},C}$, we have $u_j \neq \perp$. Hence $g(u_j)(v_j)(w_j)$ passes the test, and is either v_j or w_j. For $j \in C\backslash\{i,k\}$, both are equal to d_j. Suppose now that $j = k$. We have $g(u_k)(v_k)(w_k) = v_k = d_k$. Finally, let $j = i$. We know from above that $u_i \neq u_k$. If $u_i = \perp$, then $g(u_i)(v_i)(w_i) = \perp = d_i$ since g is strict in its first argument. If $u_i \neq \perp$, then $u_i = \mathit{ff}$, hence $g(u_i)(v_i)(w_i) = w_i = d_i$.

2. $\forall i \in C \;\; d_i \neq \perp$: We treat this case more briefly. There are two cases:

   (a) $R^\kappa \subseteq S^n_{C,C}$: Then $d \in S^n_{C,C}$ by the definition of S, and we can take a constant vector e.

   (b) $R^\kappa \not\subseteq S^n_{C,C}$: We take $u \in R^\kappa$ such that $u_i \neq \perp$ for all $i \in C$ and such that $u_i \neq u_j$ are different for some i, j in C, say, $u_i = tt$ and $u_j = f\!f$. Let:
   $$C_1 = \{k \in C \mid u_k = u_i\} \quad C_2 = \{k \in C \mid u_k = u_j\} \; .$$
   We can apply induction to C_1 and C_2, obtaining v and w. Then the proof is completed with the help of $N = \lambda xyz.\ \mathit{if}\ x\ \mathit{then}\ y\ \mathit{else}\ z$. □

Here is an example of the use of Sieber-sequential relations, taken from [Sie92].

Exercise 4.5.18 *Show that there is no term of Finitary* PCF *of type* $(\kappa \to \kappa \to \kappa) \to \kappa$ *such that:*
$$F(g_1) = tt \quad F(g_2) = tt \quad F(g_3) = tt \quad F(\perp) = \perp$$
where g_1, g_2, g_3 *are such that:*
$$\begin{array}{ll} g_1(\perp)(f\!f) = tt & g_1(f\!f)(tt) = tt \\ g_2(tt)(\perp) = tt & \\ g_3(\perp)(tt) = tt\ . & \end{array}$$
Hints: (1) Notice that $(0,0,0,\perp) \notin S^4_{\{1,2,3\},\{1,2,3,4\}}$. *(2) The relations* $S^4_{\{1,2\},\{1,2\}}$ *and* $S^4_{\{1,3\},\{1,3\}}$ *arise when trying to prove that* $(g_1, g_2, g_3, \perp) \in S^4_{\{1,2,3\},\{1,2,3,4\}}$.

Exercise 4.5.19 *Let* g_1, g_2 *be the functions whose minimal points are described by:*
$$\begin{array}{ll} g_1(f\!f)(\perp)(\perp) = f\!f & g_1(tt)(tt)(tt) = tt \\ g_2(\perp)(f\!f)(\perp) = f\!f & g_2(tt)(tt)(f\!f) = tt\ . \end{array}$$
Show that there is no closed term M *of Finitary* PCF *of type* $(\kappa \to \kappa \to \kappa \to \kappa) \to \kappa$ *such that* g_1 *and* g_2 *are the only minimal points of* $[\![M]\!]$. *Generalise this to first-order functions* g_1 *et* g_2 *(of the same type) whose minimal points are described by:*
$$\begin{array}{ll} g_1(A_0) = a_0 & g_1(B_0) = b_0 \\ g_2(A_1) = a_0 & g_2(B_1) = b_1 \end{array}$$
where $a_0 \neq b_0$, $a_0 \neq b_1$, $A_0 \uparrow A_1$ *(i.e.* $\exists A \;\, A \geq A_0, A_1$*),* $B_0 \not\uparrow B_1$, $B_0 \not\uparrow A_1$ *and* $A_0 \not\uparrow B_1$. *(Hint: Find* g_3 *such that* $(g_1, g_2, g_3) \in S^3_{\{1,2\},\{1,2,3\}}$.*)*

It is presently unknown whether the definability problem for Finitary PCF is decidable at all types. (See also the related open problem at the end of section 6.4.) On the other hand, Loader has shown that definability is undecidable for "Finitary λ-calculus", that is, λ-calculus without constants, and whose base types are interpreted by finite sets [Loa94].

4.6 Interpretation of the Untyped λ-Calculus

We recall the grammar and the basic computation rule of the untyped λ-calculus (cf. chapter 2).

$$\begin{array}{ll} \textit{Terms}: & v ::= x \mid y \mid \ldots \\ & M ::= v \mid (\lambda v.M) \mid (MM) \\ \beta\textit{-reduction}: & (\lambda x.M)N \to M[N/x] \; . \end{array}$$

In order to use the work done in the typed case, it is useful to represent the untyped calculus as a typed calculus with a special type δ, and the following typing rules:

$$\frac{x : \delta \in \Gamma}{\Gamma \vdash x : \delta} \quad \frac{\Gamma, x : \delta \vdash M : \delta}{\Gamma \vdash \lambda x : \delta.M : \delta} \quad \frac{\Gamma \vdash M : \delta \quad \Gamma \vdash N : \delta}{\Gamma \vdash MN : \delta}$$

Observe that if a type $\delta \to \delta$ could contract into a type δ in the introduction rule, and vice versa, if the type δ could expand into a type $\delta \to \delta$ in the elimination rule, then we would have the same rules as in the simply typed λ-calculus. In other words, we can apply the standard apparatus provided we have a type whose elements can be seen both as arguments and as functions. The following definition makes this intuition formal.

Definition 4.6.1 *An object D in a CCC* **C** *is called* reflexive *if there is a pair (i, j), called retraction pair, such that:*

$$i : D^D \to D, \quad j : D \to D^D, \quad j \circ i = id \; .$$

We simply write $D^D \lhd D$ to indicate the existence of a retraction pair.

The domain-theoretical construction described in section 3.1 can be used to build reflexive objects in the category of cpo's. Next, we describe a different construction with a set-theoretical flavour that serves the same purpose. The resulting models are usually called *graph-models.*

Example 4.6.2 (graph-model) *Let A be a non-empty set equipped with an injective coding $\langle \_, \_ \rangle : \mathcal{P}_{fin}(A) \times A \to A$ In the following we denote with $a, b, \ldots$ elements of A; with $\alpha, \beta, \ldots$ elements of $\mathcal{P}_{fin}(A)$; and with $X, Y, \ldots$ elements of the powerset $\mathcal{P}(A)$. We define for $f \in$* **Dcpo**$[\mathcal{P}(A), \mathcal{P}(A)]$*:*

$$\textit{Graph}(f) = \{\langle \alpha, a \rangle \mid a \in f(\alpha)\} \in \mathcal{P}(A) \; .$$

Vice versa for $X \in \mathcal{P}(A)$, we define $Fun(X) : \mathcal{P}(A) \to \mathcal{P}(A)$ as follows:

$$Fun(X)(Y) = \{a \mid \exists \alpha \, (\langle \alpha, a \rangle \in X \textit{ and } \alpha \subseteq Y)\} \; .$$

Proposition 4.6.3 *Given any non-empty set A with an injective coding $\langle \_,\_ \rangle : \mathcal{P}_{fin}(A) \times A \rightarrow A$ the complete lattice $\mathcal{P}(A)$, ordered by inclusion, is a reflexive object in* **Dcpo**, *via the morphisms Graph and Fun.*

PROOF. We take the following elementary steps:

- $Graph$ is monotonic: $f \leq g \Rightarrow Graph(f) \subseteq Graph(g)$.
- $Graph$ preserves directed lub's, namely $\{f_i\}_{i\in I}$ directed implies $Graph(\bigvee_{i\in I}\{f_i\}) \subseteq \bigcup_{i\in I} Graph(f_i)$. We observe $\langle \alpha, a\rangle \in Graph(\bigvee_{i\in I}\{f_i\})$ iff $a \in (\bigvee_{i\in I}\{f_i\})(\alpha) = \bigcup_{i\in I} f_i(\alpha)$ iff $a \in f_i(\alpha)$, for some $i \in I$.
- Fun is monotonic in both arguments: if $X \subseteq X'$, $Y \subseteq Y'$ then $Fun(X)(Y) \subseteq Fun(X')(Y')$.
- Fun is continuous in both arguments: if $\{X_i\}_{i\in I}$, $\{Y_j\}_{j\in J}$ are directed then $Fun(\bigcup_{i\in I} X_i)(\bigcup_{j\in J} Y_j) \subseteq \bigcup_{i\in I, j\in J} Fun(X_i)(Y_j)$.
- $(Graph, Fun)$ is a retraction: $Fun(Graph(f))(X) = f(X)$. Notice that this is the only condition that depends on the assumption that the coding is injective. □

The construction of the graph-model is parametric with respect to the choice of the set A and of the coding $\langle \_,\_ \rangle$. If we define a coding $\langle \_,\_ \rangle : \mathcal{P}_{fin}(\omega) \times \omega \rightarrow \omega$ where ω is the collection of natural numbers, then we obtain a family of reflexive objects also known as $\mathcal{P}(\omega)$ graph-models.

Exercise 4.6.4 *Build an example of a coding $\langle \_,\_ \rangle : \mathcal{P}_{fin}(\omega) \times \omega \rightarrow \omega$.*

The so called D_A *graph-models* are obtained by a "free construction" of the coding function. Let At be a non empty set. We suppose that elements in At are atomic, in particular they cannot be regarded as pairs. Define:

$$\begin{cases} A_0 & = At \\ A_{n+1} & = A_n \cup \{(\alpha, a) \mid \alpha \in \mathcal{P}_{fin}(A_n), a \in A_n\} \\ A & = \bigcup_{n<\omega} A_n \ . \end{cases}$$

Exercise 4.6.5 *Verify that $\langle \_,\_ \rangle : \mathcal{P}_{fin}(A) \times A \rightarrow A$ defined as $\langle \alpha, a\rangle = (\alpha, a)$ is the desired coding.*

Having verified the existence of various techniques to build reflexive objects in CCC's, we introduce in figure 4.7 the interpretation of the untyped $\lambda\beta$-calculus in these structures. This is the same as the interpretation of the simply typed λ-calculus up to insertion of the maps i, j which collapse the hierarchy of types to D. The notion of λ-theory (cf. definition 4.4.1) is readily adapted to the untyped case.

$$\begin{array}{lll}
(\text{Asmp}) & [\![x_1 : \delta, \ldots, x_n : \delta \vdash x_i : \sigma_i]\!] & = \pi_{n,i} \\
(\to_I) & [\![\Gamma \vdash \lambda x : \delta . M : \delta]\!] & = i \circ \Lambda([\![\Gamma, x : \delta \vdash M : \delta]\!]) \\
(\to_E) & [\![\Gamma \vdash MN : \delta]\!] & = ev \circ \langle j \circ [\![\Gamma \vdash M : \delta]\!], [\![\Gamma \vdash N : \delta]\!] \rangle
\end{array}$$

Figure 4.7: Interpretation of the untyped λ-calculus in a CCC

Definition 4.6.6 *Let T be a collection of judgments of the shape $\Gamma \vdash M = N : \delta$ such that $\Gamma \vdash M : \delta$ and $\Gamma \vdash N : \delta$. T is an untyped λ-theory if it is equal to the smallest set containing T and closed under the rules obtained from figure 4.6 by replacing all types with the type δ.*

Every interpretation induces an untyped λ-theory.

Theorem 4.6.7 *Let **C** be a CCC with a reflexive object D and $[\![\ ]\!]$ be an interpretation of the untyped λ-calculus defined over **C** in the sense of figure 4.7. Then the following collection is an untyped λ-theory*

$$Th(\mathbf{C}) = \{\Gamma \vdash M = N : \delta \mid \Gamma \vdash M : \delta, \Gamma \vdash N : \delta, [\![\Gamma \vdash M : \delta]\!] = [\![\Gamma \vdash N : \delta]\!]\} \ .$$

Proof hint. The proof of this result follows the same schema as in the typed case. The crucial point is to verify the substitution theorem. □

Next we describe a general construction, called *Karoubi envelope* that given a reflexive object D in a CCC, produces a new CCC of *retractions over D*. Apart for its intrinsic interest, this construction can be adapted to reverse the previous theorem, namely to show that every untyped λ-theory is the theory induced by a reflexive object in a CCC; a result very much in the spirit of theorem 4.4.4. The construction is similar to that described next in the proof of theorem 4.6.9. The difference is that rather than starting with a reflexive object in a CCC one starts from a λ-theory and the related monoid of terms and composition (see [Sco80] for details).

Definition 4.6.8 (Karoubi envelope) *Let **C** be a CCC and D be a reflexive object in **C**. The Karoubi envelope is the category $\mathbf{Ret}(D)$ of retractions over D defined as follows:*

$$\begin{array}{rl}
\mathbf{Ret}(D) & = \{r : D \to D \mid r \circ r = r\} \\
\mathbf{Ret}(D)[r,s] & = \{f : D \to D | s \circ f \circ r = f\} \ .
\end{array}$$

Theorem 4.6.9 *If **C** is a CCC and (D, i, j) is a reflexive object in **C** then $\mathbf{Ret}(D)$ is a CCC.*

PROOF. The proof is a matter of translating λ-calculus encodings into the categorical language and checking that everything goes through. Here we just remind the reader of the encoding. When we write λ-terms for building morphisms it is intended that one has to take the *interpretations* of such λ-terms.

In the untyped $\lambda\beta$-calculus we can define a fixed point combinator $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, and terms for pairing and projections:

$$[\_,\_] = \lambda x.\lambda y.\lambda p.pxy, \quad p_1 = \lambda p.p(\lambda x.\lambda y.x), \quad p_2 = \lambda p.p(\lambda x.\lambda y.y)$$

so that $p_1[x, y] = x$, $p_2[x, y] = y$.

• Terminal Object. In a CCC we have a unique morphism $* : D \to 1$. Moreover if we take $Y(\lambda x.x)$ we get a morphism from 1 to D. From this follows $1 \lhd D$. Then we take the retraction determined by 1 as the terminal object in $\mathbf{Ret}(D)$.

• Product. The pairing and projections defined above show that $D \times D \lhd D$ via a retraction that we denote with $\langle\_,\_\rangle : D \times D \to D$, $(\pi_1, \pi_2) : D \to D \times D$. If r and s are retractions then we define their product as:

$$r \times s = \lambda x.\langle r(\pi_1(x)), s(\pi_2(x))\rangle \ .$$

• Exponent. If r and s are retractions then define their exponent as:

$$r \to s = \lambda x.i(s \circ j(x) \circ r) \ .$$

□

As a second application of the category of retractions, we use $\mathbf{Ret}(D)$ as a frame for an abstract formulation of Engeler's theorem [Eng81] on the embedding of algebras in λ-models. In the following, $\mathbf{C}$ is a CCC with enough points (cf. definition 4.5.4) and D is a reflexive object in $\mathbf{C}$. Let $\Sigma \equiv \{\sigma_i^{n_i}\}_{i \in I}$ be a *finite signature*, that is a finite collection of names of operators σ_i with the relative arity n_i. We are interested in a notion of Σ-algebra in which the carriers are objects in $\mathbf{Ret}(D)$ and the operators are maps in $\mathbf{Ret}(D)$ of the appropriate type.

Definition 4.6.10 (Σ_D-algebra) *A* Σ_D-algebra *is a pair $(r, \{f_i\}_{i\in I})$ where $r \in \mathbf{Ret}(D)$, and for all $i \in I$, $f_i : D^{n_i} \to D$, r^{n_i} stands for $r \times \cdots \times r$ n_i times, and $r \circ f_i \circ r^{n_i} = f_i$. A* morphism *of Σ_D-algebras $h : (r, \{f_i\}_{i\in I}) \to (r', \{g_i\}_{i\in I})$ is a morphism $h : D \to D$ such that $r' \circ h \circ r = h$, and for all $i \in I$, $h \circ f_i \circ r^{n_i} = g_i \circ (h \circ r)^{n_i}$.*

Theorem 4.6.11 (embedding Σ_D-algebras) *There is a Σ_D-algebra $(id, \{F_i\}_{i\in I})$ such that any other Σ_D-algebra $(r, \{f_i\}_{i\in I})$ can be embedded into it by a monomorphism.*

PROOF. For the sake of simplicity we just consider the case $\Sigma = \{\sigma^2\}$. Assume that $\langle \_, \_ \rangle$, and π_1, π_2 are, respectively, the pairing and the projections definable in the λ-calculus as in the proof of theorem 4.6.9. We take $F = \lambda(x_1, x_2).(\pi_2 x_1)x_2$. We note that recursive definitions are available in the untyped λ-calculus thanks to the fixed point combinator. Given the Σ_D-algebra $(r, \{f\})$ we define recursively a morphism $\rho : D \rightarrow D$ as follows:

$$\rho(a) = \langle a, \lambda x.\rho(f(a, \pi_1 x))\rangle \ .$$

The basic idea of the embedding is to put into the data the information on the behaviour of the operations defined on them. In the first place let us observe that ρ is a mono as (we use the enough points hypothesis):

$$\rho(a) = \rho(b) \ \Rightarrow \ a = \pi_1(\rho(a)) = \pi_2(\rho(b)) = b \ .$$

Clearly $\rho \circ r : r \rightarrow id$ in $\mathbf{Ret}(D)$ as $\rho \circ r = id \circ \rho \circ r \circ r$. Also since $r(f(rx, ry)) = f(rx, ry)$ we have:

$$\begin{aligned} F(\rho(ra), \rho(rb)) \ &= (\lambda x.\rho(f(ra, \pi_1 x)))\rho(rb) \ &&= \rho(f(ra, \pi_1\rho(rb))) \\ &= \rho(f(ra, rb)) &&= (\rho \circ r)(f(ra, rb)). \end{aligned}$$

Therefore $\rho \circ r : (r, \{f\}) \rightarrow (id, \{F\})$ is a Σ_D-algebras mono-morphism. □

Remark 4.6.12 *The following describes a schema for coding finite signatures which suggests how to generalize the previous proof. Given* $\Sigma = \{\sigma, f_1^1, \ldots, f_n^n\}$ *define:*

$$\begin{aligned} \rho(a) = \ & \langle a, \\ & \langle \rho(f_1 a), \\ & \langle \lambda x.\rho(f_2 a(\pi_1 x)), \\ & \cdots\cdots \\ & \langle \lambda x_1 \ldots \lambda x_n.\rho(f_n a)(\pi_1 x_1)\cdots(\pi_1 x_n), * \rangle \cdots \rangle \end{aligned}$$

where:

$$\begin{aligned} F_1 &= \lambda x.\pi_1(\pi_2(\pi_2 x)) \\ F_2 &= \lambda x.\pi_1(\pi_2(\pi_2(\pi_2 x))) \\ &\cdots\cdots \\ F_n &= \lambda x.\pi_1(\pi_2 \cdots (\pi_2 x)\cdots) \ \textit{with } n+1 \ \pi_2\textit{'s} \ . \end{aligned}$$

Chapter 5

CCC's of Algebraic Dcpo's

In this chapter, we provide a finer analysis of algebraicity. The central result – which was conjectured by Plotkin and was first proved in [Smy83a] – is that there exists a maximum cartesian closed full subcategory (full sub-CCC) of ω**Acpo** (the category of ω-algebraic cpo's). Jung has extended this result: he has characterised the maximal cartesian closed full subcategories of **Acpo** and **Adcpo** (and of ω**Adcpo** as well).

In section 5.1, we define continuous dcpo's. Theyr are dcpo's where approximations exist without being necessarily compact. Continuous lattices have been investigated in depth from a mathematical perspective [GHK$^+$80]. Our interest in continuous dcpo's arises from the fact that retracts of algebraic dcpo's are not algebraic in general, but are continuous. Much of the technical work involved in our quest of maximal full cartesian closed subcategories of (d)cpo's involves retracts: they are smaller, hence easier to work with. In section 5.2, we introduce two cartesian closed categories: the category of profinite dcpo's and the category of L-domains, both with continuous functions as morphisms. In section 5.3, we show that the algebraic L-domains and the bifinite domains form the two maximal cartesian closed full subcategories of **Acpo**, and derive Smyth's result for ω**Acpo** with little extra work. In section 5.4, we treat more sketchily the situation with **Adcpo**. The material of sections 5.3 and 5.4 follows [Jun88]. In section 5.5, we show a technical result needed in section 5.3: a partial order is a dcpo if and only if all its well-founded subsets have a lub.

5.1 Continuous Dcpo's

In order to define algebraic dcpo's, we first introduced the notion of compact element, and then we defined algebraicity. The definition of continuous dcpo's is more direct, and more general.

Definition 5.1.1 (continuous dcpo) *Let D be a dcpo. For elements $x, y \in D$, we say that x is way-below y, and write $x \ll y$, if*

$$\Delta \textit{ directed},\ y \leq \bigvee \Delta \Rightarrow \exists\, \delta \in \Delta \;\; x \leq \delta.$$

D is called continuous if for any x in D, $\Downarrow x = \{y \mid y \ll x\}$ is directed and has x as lub.

Notice that by definition x is compact iff $x \ll x$. We leave the proof of the following easy properties as an exercise:

$$\begin{array}{l} \text{If } x \ll y, \text{ then } x \leq y. \\ \text{If } x' \leq x \ll y \leq y', \text{ then } x' \ll y'. \end{array}$$

Clearly, algebraic dcpo's are continuous, but the converse does not hold.

Exercise 5.1.2 (cont-nonalg) *Show that the interval $[0,1]$ of real numbers is continuous but not algebraic. Hint: Prove that $x \ll y$ iff $x = 0$ or $x < y$.*

Lemma 5.1.3 *In a continuous dcpo, $x \ll y$ holds iff the following implication holds:*

$$\Delta \textit{ directed},\ y = \bigvee \Delta \Rightarrow \exists\, \delta \in \Delta \;\; x \leq \delta.$$

Proof. Suppose $y = \bigvee \Delta \Rightarrow \exists\, \delta \in \Delta \;\; x \leq \delta$, for all Δ. Since $y = \bigvee(\Downarrow y)$, we have $x \leq y'$ for some $y' \ll y$. Hence $x \ll y$ since $x \leq y' \ll y$. □

Lemma 5.1.4 *Let D be a dcpo and $x \in D$. If $\Delta \subseteq \Downarrow x$ is directed and $x = \bigvee \Delta$, then $\Downarrow x$ is directed and $x = \bigvee(\Downarrow x)$.*

Proof. If $y \ll x$, $y' \ll x$, then by definition $y \leq a$, $y' \leq a'$ for some $a, a' \in \Delta$. By directedness we have $a, a' \leq y''$ for some $y'' \in \Delta$. Hence $y, y' \leq y'' \in \Downarrow x$. The inequality $x \leq \bigvee(\Downarrow x)$ follows from the obvious inequality $\bigvee \Delta \leq \bigvee(\Downarrow x)$. □

The density property of the real line is generalised as follows.

Lemma 5.1.5 (interpolation) *In a continuous dcpo D, if $x \ll y$, then there exists $z \in D$ such that $x \ll z \ll y$.*

Proof. Consider $\Delta = \{a \in D \mid \exists\, a' \in D \;\; a \ll a' \ll y\}$. If we show that Δ is directed and $\bigvee \Delta = y$, then we can conclude, since by definition $x \ll y$ implies $x \leq a$ for some $a \in \Delta$, hence $x \in \Delta$. The set Δ is non-empty, since the directedness of $\Downarrow y$ implies a fortiori its non emptyness. Thus one can find at least an $a' \ll y$, and then at least an $a \ll a'$. Suppose that $a \ll a' \ll y$ and $b \ll b' \ll y$. By directedness, there exists $c' \in D$ such that $a', b' \leq c' \ll y$. Hence $a, b \ll c'$, and by directedness again $a, b \leq c \ll c'$ for some c, which is in Δ since $c \ll c' \ll y$. Hence Δ is directed. Since $a \ll a' \ll y$ implies $a \ll y$, we have $\bigvee \Delta \leq \bigvee \Downarrow y$. Conversely, if $y' \ll y$, then $\Downarrow y' \subseteq \Delta$, hence $\bigvee \Downarrow y = \bigvee_{y' \in \Downarrow y} \bigvee \Downarrow y' \leq \bigvee \Delta$. □

Lemma 5.1.6 *In a continuous dcpo D, minimal upper bounds (mub's) of finite sets of compact elements are compact.*

PROOF. Let $A \subseteq \mathcal{K}(D)$ be finite, and $x \in MUB(A)$. Let $a \in A$. Since $a \ll a \leq x$, we have $a \ll x$. By directedness there exists $x' \in \Downarrow x \cap UB(A)$. Then $x' \leq x$ and $x \in MUB(A)$ imply $x' = x$. Finally, $x = x' \ll x$ means exactly that x is compact. □

We move on to retractions and projections (cf. definition 3.1.2, and exercise 4.6.9, which shows important ways of constructing retractions out of other retractions).

Definition 5.1.7 (retraction, projection) *In a category, an arrow $r : D \to D$ is a retraction, or an idempotent, if $r \circ r = r$. In* **Dcpo** *the image of a retraction, endowed with the induced ordering, is called a retract. If a retraction r is such that $r \leq id$, we say that r is a projection.*

Projections are determined by their images.

Proposition 5.1.8 *For two projections p, p' over the same dcpo D, one has $p \leq p'$ iff $p(D) \subseteq p(D')$.*

PROOF. Suppose $p \leq p'$. If $y \in p(D)$, then $y = p(y) \leq p'(y) \leq y$ since $p \leq p' \leq id$, hence $y = p'(y) \in p'(D)$. Conversely, notice that for any projection p and any $x \in D$ one has $p(x) = \max\{y \in p(D) \mid y \leq x\}$. Then $p \leq p'$ follows obviously. □

Lemma 5.1.9 *Fix a dcpo D and $x \in D$.*

1. *$\downarrow x$ is a retract of D.*
2. *If x is compact, then $\uparrow x$ is a retract of D.*

PROOF. (1) $\downarrow x = r(D)$ where

$$r(y) = \begin{cases} y & \text{if } y \leq x \\ x & \text{otherwise .} \end{cases}$$

We check that r is continuous. If $\bigvee \Delta \leq x$, then $\forall \delta \in \Delta \;\; \delta \leq x$, hence $r(\bigvee \Delta) = \bigvee \Delta = \bigvee r(\Delta)$. If $\bigvee \Delta \not\leq x$, then $\exists \delta \in \Delta \;\; \delta \not\leq x$. Then we have $r(\delta) = x$, which implies $\bigvee r(\Delta) = x = r(\bigvee \Delta)$.

(2) If $x = d$ is compact, we have $\uparrow d = s(D)$ where

$$s(x) = \begin{cases} x & \text{if } x \geq d \\ d & \text{otherwise .} \end{cases}$$

If $\bigvee \Delta \geq d$, then $\exists \delta \in \Delta \;\; d \leq \delta$. We set $\Delta' = \Delta \cap \uparrow \delta$; since obviously $s(\bigvee \Delta') = \bigvee s(\Delta')$, we deduce $s(\bigvee \Delta) = \bigvee s(\Delta)$. If $\bigvee \Delta \not\geq d$, then $\forall \delta \in \Delta \;\; \delta \not\geq d$, so that $s(\bigvee \Delta) = d$ and $s(\Delta) = \{d\}$, hence $s(\bigvee \Delta) = \bigvee s(\Delta)$. □

Retractions are at the heart of our interest in continuous dcpo's. Indeed, retracts of algebraic dcpo's are not algebraic in general, but only continuous (see exercise 5.1.11).

Proposition 5.1.10 (continuous retracts) *A retract $r(D)$ of a dcpo D is a subdcpo. If D is continuous, then $r(D)$ is continuous.*

Proof. Let $\Delta \subseteq r(D)$ be directed. Then $r(\bigvee \Delta) = \bigvee r(\Delta) = \bigvee \Delta$, since $\forall \delta \in \Delta \;\; r(\delta) = \delta$. Suppose that $x \ll y \in r(D)$. We show that $r(x)$ is way-below y in $r(D)$. If $y \leq \bigvee \Delta$, with $\Delta \subseteq r(D)$, then $x \ll y$ implies $x \leq \delta$ for some $\delta \in \Delta$; hence $r(x) \leq r(\delta) = \delta$. Since $y = r(y) = r(\bigvee \Downarrow y) = \bigvee \{r(x) \mid x \ll y\}$, we conclude by lemma 5.1.4. □

Exercise 5.1.11 *Show that any continuous dcpo D is isomorphic to a projection of $Ide(D)$.*

We end the section with a topological exercise. Continuous lattices were met (and named so) by Scott in his quest of spaces whose topology could be entirely understood from an underlying partial order [Sco72].

Exercise 5.1.12 *Let D be a continuous cpo. Show the following properties:*

1. *$\Uparrow x$ is Scott open, and these opens form a basis of Scott topology.*
2. *If D is a complete lattice, then $\forall x \in D \;\; x = \bigvee \{\bigwedge U \mid x \in U\}$.*
3. *$x \ll y \Leftrightarrow y \in (\uparrow x)^\circ$ (the interior of $\uparrow x$).*

Exercise 5.1.13 (injective spaces) *A topological space D is called injective if whenever $X \in$ **Top**, $Y \subseteq X$, and $f : Y \to D$ are given, with f continuous for the induced subspace topology, there exists a continuous extension $\overline{f} : X \to D$ of f. Show that the following properties are equivalent for a T_0 space:*

1. *D is injective,*
2. *D is a retract of a product of copies of* **O**,
3. *D is a continuous lattice and its topology is Scott topology.*

Hints: Every space X is homeomorphic to a subspace of a product $\Pi_{U \in \Omega X} \mathbf{O}$ of copies of **O**. *An injective subspace Y of a space X is a retract of X: take $\overline{id_Y} : X \to Y$.* **O** *is continuous, and continuous lattices are stable under products and retractions (cf. proposition 5.1.10). If D is a continuous lattice, $Y \subseteq X$, and $f : Y \to D$, then define $\overline{f}$ by: $\overline{f}(x) = \bigvee \{\bigwedge \{f(y) \mid y \in Y \cap U\} \mid x \in U\}$.*

5.2 Bifinite Domains and L-Domains

Recall that, if D and E are algebraic, then the step functions $d \to e$ are compact (lemma 1.4.8) , but $D \to E$ need not be algebraic (exercise 1.4.15). One way to ensure algebraicity is to impose bounded completeness (theorem 1.4.12) on D and E. But this assumption can be weakened.

Definition 5.2.1 *Let $(P, \leq)$ be a preorder, and let $A \subseteq P$. The set of minimal upper bounds (mub's) of A is denoted $MUB(A)$, and is called* complete *if*

$$\forall y \in UB(A)\ \exists x \in MUB(A)\ x \leq y.$$

Consider a continuous function f, and two step functions $d \to e \leq f$, $d' \to e' \leq f$. We want to constuct a compact upper bound h of $d \to e$ and $d' \to e'$ such that $h \leq f$. Suppose that $MUB(d, d')$ is complete. Then we may choose e'' such that $e'' \leq f(d'')$ for each $d'' \in MUB(d, d')$, and set $h_1(d'') = e''$. In general, one has to consider in turn the compatible pairs $d''_1, d''_2 \in MUB(d, d')$, leading to the construction of a new function h_2, and so on. At each step we have by construction $h_n \leq f$. There are two different further assumptions that allow us to stop this chain, and to ensure that each h_n is monotonic (which implies its continuity by construction) and compact.

1. Strengthen the completeness assumption to:

   $$\forall y \in UB(d, d')\ \exists ! x \in MUB(d, d')\ x \leq y.$$

   Then the sequence of the h_n's stops at h_1, since there are no compatible distinct minimal upper bounds of $\{d, d'\}$. Moreover, the above e'' can be defined canonically as the only member of $MUB(e, e')$ below $f(d'')$. This canonicity allows us to prove that h_1 is compact (hint: if $h_1 \leq \bigvee \Delta$, take $f \in \Delta$ such that $e \leq f(d)$ and $e' \leq f(d')$, and show $h_1 \leq f$).

2. Impose finiteness conditions on minimal upper bounds: if $MUB(d, d')$ is finite, and if the process of taking minimal upper bounds of minimal upper bounds, and so on, terminates, then the above construction stops at some h_n. Moreover the finiteness of the description of h_n allows to prove that it is compact.

This discussion had only a motivating purpose, since we only addressed the construction of a compact upper bound of compacts of the form $d \to e$, not of any pair of compact approximations of f. The two kinds of assumptions correspond to L-domains and profinite domains, respectively. The rest of the section is devoted to their study. We first introduce the profinite dcpo's (a terminology due to Gunter) and show that they form a cartesian closed full subcategory of **Adcpo**. We recall that **Dcpo** is a cartesian closed category and that lub's of functions are defined pointwise.

Definition 5.2.2 (profinite) *Let D be a dcpo D. A projection is called finite if its image is finite. We say that D is profinite if the finite projections $p : D \to D$ form a directed set whose lub is the identity. We denote with* **Prof** *the category of profinite dcpo's and continuous functions. A profinite dcpo which is moreover a cpo is called* bifinite. *We denote with* **Bif** *the full subcategory of bifinite cpo's.*

The terminology "bifinite", due to Taylor, comes from a more categorical characterisation of profinite and bifinite dcpo's to be found in chapter 7: they are limits and colimits at the same time (whence "bi") of families of finite (d)cpo's. The bifinite domains have been first explored in [Plo76], under the name SFP (Sequence of Finite Projections). The following proposition justifies the name of finite projections.

Proposition 5.2.3 *Let D be a cpo and $p : D \to D$ be a projection such that $im(p)$ is finite. Then every element in $im(p)$ is compact in D and p is compact in $D \to D$. Moreover, if D is bifinite, then all compact projections over D have a finite image.*

Proof. We suppose $x = p(x)$ and $x \leq \bigvee \Delta$, with Δ directed in $D \to D$. Then:

$$x \leq \bigvee \Delta \quad \Rightarrow \quad x = p(x) \leq p(\bigvee \Delta) = \bigvee p(\Delta)$$

Since $im(p)$ is finite, there is $\delta \in \Delta$ such that $\bigvee p(\Delta) = p(\delta)$, hence $x \leq p(\delta) \leq \delta$. Next, we suppose $p \leq \bigvee \Delta$, with Δ directed set in D. We have just proven that:

$$\forall x \in im(p)\, \exists \delta_x \in \Delta\, (x = p(x) \leq \delta_x(x))$$

Since $im(p)$ is finite, we have that $\exists \delta \in \Delta \forall x \in im(p)\, (x \leq \delta(x))$. Hence $\forall y\, (p(y) \leq \delta(p(y)) \leq \delta(y))$. Finally, suppose D is bifinite, say $id = \bigvee_{i \in I} p_i$ with $im(p_i)$ finite. Let p be a compact projection. Then $p = \bigvee_{i \in I}(p_i \circ p)$, and there is i such that $p \leq p_i \circ p \leq id \circ p \leq p$. Hence $p = p_i \circ p$ and $im(p) \subseteq im(p_i)$ which is finite. □

Proposition 5.2.4 (profinite - CCC) *1. Every profinite dcpo D is algebraic, with:*

2. Profinite dcpo's (bifinite cpo's, respectively) and continuous maps form a cartesian closed category.

Proof. (1) If D is profinite, then $x = \bigvee\{p(x) \mid p \text{ finite projection}\}$, for any x. It is enough to show that $p(x)$ is compact, for any finite projection p. If $p(x) \leq \bigvee \Delta$, then $p(x) = p(p(x)) \leq \bigvee p(\Delta)$. Since $p(\Delta)$ is finite, there exists $\delta \in \Delta$ such that $\bigvee p(\Delta) = p(\delta)$. Then $p(x) \leq p(\delta) \leq \delta$.

(2) It is enough to check that if $D, E \in$ **Prof** are profinite, then $D \times E, D \to E \in$ **Prof**. For $D \times E$, take the set of projections $p \times q = \langle p \circ \pi_1, q \circ \pi_2 \rangle$, where

p, q are finite projections. For $D \to E$, define, for any pair of finite projections p, q on D, E:

$$r(f) = (x \mapsto q(f(p(x))))$$

(cf. exercise 4.6.9) Clearly f is the lub of these $r(f)$'s. As for the finiteness of $im(r)$, observe:

- there are finitely many functions from $p(D)$ to $q(E)$;
- every f such that $r(f) = f$ restricts to a function $f : p(D) \to q(E)$, and is determined by this restriction, because $f(x) = f(p(x))$ for any x. □

When there are only denumerably many finite projections, a profinite dcpo is called ω-profinite. This name is justified by the following exercise.

Exercise 5.2.5 *Show that an ω-profinite dcpo is ω-algebraic.*

We shall give an alternative characterisation of profinite dcpo's.

Definition 5.2.6 (properties m, M) *We say that a partial order $(Y, \leq)$*

- *satisfies property m (notation $Y \models m$) if for all $X \subseteq_{fin} Y$ the set $MUB(X)$ of mub's of X is complete, i.e., $(\forall y \in UB(X)\ \exists x \in MUB(X)\ x \leq y)$.*
- *satisfies property M (notation $Y \models M$) if it satisfies property m, with the additional condition that $MUB(X)$ is finite for any finite subset X.*

Theorem 5.2.7 *Let D be an algebraic cpo. D is a bifinite domain iff the following properties hold:*

1. *$\mathcal{K}(D) \models m$,*
2. *$U^\infty(X) = \bigcup_{n\in\omega} U^n(X)$ is finite for any finite subset of compact elements X,where U is an operator on subsets defined by*

$$U(X) = \bigcup\{MUB(Y) \mid Y \subseteq_{fin} X\}.$$

PROOF. Notice that in particular, $X \subseteq U(X)$ for any X, and $MUB(X) \subseteq U(X)$ if X is finite. Therefore properties (1) and (2) imply that $\mathcal{K}(D) \models M$ [1]. .

$(\Rightarrow)$ Let D be a bifinite domain. We recall that $\mathcal{K}(D) = \bigcup\{p(D) \mid p$ finite projection$\}$. If $X \subseteq_{fin} \mathcal{K}(D)$, then $X \subseteq p(D)$ for some p, by directedness and proposition 5.1.8. Call Z the set of mub's of X in $p(D)$, which exists, is finite, and is complete in

[1] Algebraic dcpo's D such that $\mathcal{K}(D)$ satisfies property M are sometimes called 2/3 SFP.

$p(D)$, since $p(D)$ is finite. We first show: $Z \subseteq MUB(X)$. Indeed suppose $z \in Z$, $z' \in UB(X)$, and $z' < z$. Then

$$\begin{array}{ll} p(z') \in UB(X) & \text{since } p(X) = X \\ p(z') < z & \text{since } p(z') \leq z' \,. \end{array}$$

This contradicts the definition of Z. Thus $Z \subseteq MUB(X)$. We next show that Z is a complete set of mub's of X in D. Take $y \in UB(X)$; then, as argued above, $p(y) \in UB(X)$, and by completeness of Z one may find $z \in Z$ such that $z \leq p(y)$, and a fortiori $z \leq y$. The completeness of Z forces $Z = MUB(X)$. Therefore $MUB(X)$ is finite and complete, and $MUB(X) \subseteq p(D)$. Similarly, $MUB(Y) \subseteq p(D)$ for any $Y \subseteq_{fin} X$. From there we deduce that $U^n(X) \subseteq p(D)$ for any n, observing that each subset of X is a fortiori included in $p(D)$.

$(\Leftarrow)$ Let A be a finite set of compacts. Then we claim:

$$\forall y \in D \quad U^\infty(A) \cap \downarrow y \text{ is directed}$$

(i.e., according to a terminology which will be introduced in definition 7.4.6, $U^\infty(A)$ is normal).

This is shown as follows: if $x, x' \in U^\infty(A) \cap \downarrow y$, then $MUB(x, x') \subseteq U^\infty(A)$, and by completeness $MUB(x, x') \cap \downarrow y \neq \emptyset$. By the claim we can set

$$p_A(y) = \bigvee (U^\infty(A) \cap \downarrow y).$$

It is left to the reader to check that this gives a directed set of finite projections having *id* as lub. □

Notice that, in the proof of $(\Leftarrow)$, we have used only mub's of pairs. The following result goes in the same direction.

Lemma 5.2.8 *Let $(D, \leq)$ be a partial order. If $MUB(X)$ is complete and finite for every subset X such that $\sharp Y \leq 2$, then $MUB(X)$ is complete and finite for every finite subset X.*

Proof. Let $X = \{a_1, \ldots, a_n\}$. We construct

$$M_2 = MUB(a_1, a_2)\ , \ldots,\ M_n = \bigcup_{x \in M_{n-1}} MUB(x, a_n).$$

If x is an upper bound of X, then by completeness x dominates an element of M_2. Continuing in the same way, we find an element y of M_n below x. Suppose moreover $x \in MUB(X)$: then $x = y$, since by construction $M_n \subseteq UB(X)$. We have proved $MUB(X) \subseteq M_n$. Since M_n is finite, $MUB(X)$ is a fortiori finite. □

Exercise 5.2.9 *Let X be finite. Show that if there exists $Y \subseteq_{fin} \uparrow X$ such that $\forall x \in \uparrow X\ \exists y \in Y\ \ y \leq x$, then $MUB(X)$ is finite and complete.*

(A) $D = \overline{\omega} \cup \{\bot, a, b\}$ (where $\overline{\omega} = \{\overline{n} \mid n \in \omega\}$ is a copy of ω), ordered as follows:

$$x \le y \text{ iff } \begin{cases} x = \bot \text{ or} \\ x = a \text{ and } y \in \overline{\omega} \text{ or} \\ x = b \text{ and } y \in \overline{\omega} \text{ or} \\ x = \overline{n}, y = \overline{m}, \text{ and } m \le n \end{cases}$$

In this example, $U(\{a, b\})$ fails to be complete.

(B) $D = \{a, b\} \cup \omega$, ordered as follows:

$$\forall n \;\; a, b < n$$

In this example, $U(\{a, b\})$ fails to be finite.

(C) $D = \{a, b\} \cup \omega_L \cup \omega_R$ (where $\omega_L = \{n_L \mid n \in \omega\}$ and $\omega_R = \{n_R \mid n \in \omega\}$), ordered as follows:

$$\begin{array}{l} \forall m, n \;\; a, b < m_L, n_R \\ m \le n \Rightarrow m_L \le n_L \text{ and } m_R \le n_R \\ m < n \Rightarrow m_L < n_R \text{ and } m_R < n_L \end{array}$$

In this example, $U^{\infty}(\{a, b\})$ fails to be finite.

Figure 5.1: Dcpo's that fail to be profinite

Exercise 5.2.10 *Show that D is bifinite iff the conditions stated in theorem 5.2.7 hold, replacing the operator U by the operator $U'(X) = \bigcup\{MUB(Y) \mid Y \subseteq X$ and $\sharp Y \le 2\}$. Show that if D is bifinite, then $U^{\infty}(X) = U'^{\infty}(X)$.*

Figure 5.1 illustrates how an algebraic dcpo may fail to be profinite. The function spaces of examples (A) and (C) are not algebraic (cf. exercise 1.4.15 and proposition 5.3.7). The function space of example (B) is algebraic, but not ω-algebraic. It is an example of L-domain, which we shall introduce next.

Definition 5.2.11 (L-domain) *An L-domain*[2] *is a cpo D such that*

$$\forall A \subseteq_{fin} D \;\; \forall x \in UB(A) \;\; \exists! y \le x \;\; y \in MUB(A).$$

Notice that in the definiton of L-domain we have traded the finiteness condition of property M against a uniqueness assumption.

[2]See also definition 12.5.4 and exercise 12.5.6.

Example 5.2.12 *The following is a minimal example of a finite partial order which is not an L-domain: $D = \{a,b,c,d,e\}$ with $a,b \leq c,d \leq e$.*

For algebraic cpo's, we can limit ourselves to finite sets of compact elements.

Exercise 5.2.13 *Show that an algebraic cpo is an L-domain iff*

$$\forall A \subseteq_{fin} \mathcal{K}(D)\ \forall x \in UB(A)\ \exists! y \leq x\ \ y \in MUB(A).$$

Hint: if $A = \{x_1, \ldots, x_n\}$, consider the sets $\{e_1, \ldots, e_n\}$, where the e_i's approximate the x_i's.

Exercise 5.2.14 *(1) Show that one can restrict definition 5.2.11 to the A's which have cardinal 2 without loss of generality (hint: the uniqueness is essential). (2) Show that, if D is algebraic, we may restrict ourselves to compacts, i.e., $A \subseteq_{fin} \mathcal{K}(D)$.*

Hence L-domains are "locally" bounded complete: any bounded subset is bounded complete.

Proposition 5.2.15 *A cpo D is an L-domain iff*

$$D \models m \text{ and } U^\infty(A) = U(A) \text{ for all finite subsets } A \text{ of } D.$$

Proof. ($\Rightarrow$) Property m holds a fortiori. To show $U^\infty(A) = U(A)$. it is enough to prove $U^2(A) \subseteq U(A)$. Let $x \in MUB(B)$, for a finite $B \subseteq U(A)$, and let A_b be a finite subset of A of which b is a mub, for any $b \in B$. We show $x \in MUB(\bigcup_{b\in B} A_b)$. By construction $x \in UB(\bigcup_{b\in B} A_b)$. Suppose $x \geq y \in UB(\bigcup_{b\in B} A_b)$. By property m, $y \geq b'$ for some mub b' of A_b. By uniqueness of the mub of A_b below x, we get $b' = b$. Hence $y \geq B$ and $y = x$.

($\Leftarrow$) Let $x \geq A \subseteq_{fin} D$. By property m there exists $a \in MUB(A)$ such that $a \leq x$. Let $a' \leq x$ be such that $a' \in MUB(A)$. By applying m again, there exists $b \in MUB(a, a')$ such that $b \leq x$. Since $U^\infty(A) = U(A)$, we have $b \in U(A)$, i.e., $b \in MUB(A')$ for some $A' \subseteq A$. Since $a, a' \in UB(A)$, we get $a = b = a'$. This proves the uniqueness of a. □

So far, we have made use of the dcpo structure only. The following proposition involves step functions, which are defined with the help of $\perp$.

Proposition 5.2.16 (L-CCC) *The category of L-domains and continuous functions is cartesian closed. The full subcategory* **L** *of algebraic L-domains is cartesian closed.*

Proof. Suppose that $f, g \leq h$ are in $D \to E$. Then $f(x), g(x) \leq h(x)$. Define $k(x)$ as the minimum upper bound of $f(x), g(x)$ under $h(x)$. This function k is the minimum upper bound of f, g under h (to check the continuity of k, given Δ, one works in $\downarrow h(\bigvee\Delta)$). If D and E are algebraic, then we already know that any h is the lub of the set of compact functions below it. We have to check that this set is directed. This follows from the bounded completeness of $\downarrow h$ (cf. theorem 1.4.12). $\Box$

As a last result in this section we show that the terminal object, products, and exponents in a full subcategory of **Dcpo**, if any, must be those of **Dcpo.**

Proposition 5.2.17 *Let* **C** *be a full subcategory of* **Dcpo**. *We denote by* $\times$, $\rightarrow$ *the product and the exponent in* **Dcpo**. *We write* $D \cong E$ *when* D *and* E *are isomorphic in* **Dcpo**, *which amounts to* D *and* E *being isomorphic in the category of partial orders. Then the following properties hold:*

1. If **C** *has a terminal object* $\hat{1}$, *then* $\hat{1}$ *is a one point cpo.*

2. If **C** *has a terminal object* $\hat{1}$ *and products* $D\hat{\times}E$, *then* $D\hat{\times}E \cong D \times E$.

3. If **C** *has terminal object* $\hat{1}$, *binary products, and exponents* $D\hat{\rightarrow}E$, *then* $D\hat{\rightarrow}E \cong D \rightarrow E$.

PROOF. (1) If $\hat{1}$ is terminal and has distinct elements x, y, then the constant functions $z \mapsto x, z \mapsto y : \hat{1} \rightarrow \hat{1}$ are continuous and distinct: contradiction. In the sequel we freely confuse $x \in d$ and $x : \hat{1} \rightarrow D$.

(2) Let $D, E \in$ **C**. Consider the products:

$$\begin{array}{ll} (D\hat{\times}E, \widehat{\pi_1}, \widehat{\pi_2}) \text{ in } \mathbf{C} & \text{with pairing denoted by } \widehat{\langle,\rangle} \\ (D \times E, \pi_1, \pi_2) \text{ in } \mathbf{Cpo} & \text{with pairing } \langle,\rangle\ . \end{array}$$

We show that $\langle\widehat{\pi_1}, \widehat{\pi_2}\rangle : D\hat{\times}E \rightarrow D \times E$ is an isomorphism in **Cpo**:

- $\langle\widehat{\pi_1}, \widehat{\pi_2}\rangle$ is injective: We have, for any $x, x' \in \hat{1} \rightarrow D\hat{\times}E$:

$$\begin{array}{rcl} \langle\widehat{\pi_1}, \widehat{\pi_2}\rangle \circ x = \langle\widehat{\pi_1}, \widehat{\pi_2}\rangle \circ x' & \Leftrightarrow & \widehat{\pi_1} \circ x = \widehat{\pi_1} \circ x' \text{ and } \widehat{\pi_2} \circ x = \widehat{\pi_2} \circ x' \\ & \Leftrightarrow & \widehat{\langle\widehat{\pi_1}, \widehat{\pi_2}\rangle} \circ x = \widehat{\langle\widehat{\pi_1}, \widehat{\pi_2}\rangle} \circ x' \\ & \Leftrightarrow & x = x'\ . \end{array}$$

- $\langle\widehat{\pi_1}, \widehat{\pi_2}\rangle$ is surjective: Let $(y, z) \in D \times E$. We have: $(y, z) = \langle\widehat{\pi_1}, \widehat{\pi_2}\rangle(\widehat{\langle y, z\rangle})$, since $\widehat{\pi_1}(\widehat{\langle y, z\rangle}) = y$ and $\widehat{\pi_2}(\widehat{\langle y, z\rangle}) = z$.

- If $(y, z) \leq (y', z')$, then $\widehat{\langle y, z\rangle} \leq \widehat{\langle y', z'\rangle}$: We can assume the existence of an object $C \in$ **C** containing at least two elements c, c', such that $c < c'$: indeed, if **C** only admits objects of cardinality 1 then the proposition is trivially true, and if **C** contains only discretely ordered sets, then in particular D, E are discretely ordered, and so are $D\hat{\times}E$ (as an object of **C**) and $D \times E$ (by the definition of product in **Dcpo**). With this fixed C, for any D and $x, x' \in D$ such that $x \leq x'$, we can build a continuous map $f_{x,x'} : C \to D$ as follows:

$$f_{x,x'}(y) = \begin{cases} x & \text{if } y \leq c \\ x' & \text{otherwise} \end{cases} .$$

With the help of these functions, we have

$$\widehat{\langle y, z\rangle} = \widehat{\langle f_{y,y'}, f_{z,z'}\rangle} \circ c \quad \widehat{\langle y', z'\rangle} = \widehat{\langle f_{y,y'}, f_{z,z'}\rangle} \circ c'.$$

Thus, by monotonicity of $\widehat{\langle f_{y,y'}, f_{z,z'}\rangle}$, we get $\widehat{\langle y, z\rangle} \leq \widehat{\langle y', z'\rangle}$.

(3) Given (1) and (2), we may work directly with the standard product $\times$. Consider the exponents:

$$\begin{array}{ll} (D \hat{\rightarrow} E, \hat{ev}) & \text{in } \mathbf{C}, \text{ with currying denoted by } \hat{\Lambda} \\ (D \to E, ev) & \text{in } \mathbf{Cpo}, \text{ with currying denoted by } \Lambda\ . \end{array}$$

We show that $\Lambda(\hat{ev}) : (D \hat{\rightarrow} E) \to (D \to E)$ is an iso.

• $\Lambda(\hat{ev})$ is injective: If $\Lambda(\hat{ev})(h) = \Lambda(\hat{ev})(h')$, then $\hat{ev} \circ (h \times id) = \hat{ev} \circ (h' \times id)$ by the bijectivity of Λ. This entails:

$$h = \hat{\Lambda}(\hat{ev} \circ (h \times id)) = \hat{\Lambda}(\hat{ev} \circ (h' \times id)) = h'.$$

• $\Lambda(\hat{ev})$ is surjective: Let $f : D \to E$. We have $f = \Lambda(\hat{ev})(\hat{\Lambda}(ev)(f))$ since

$$\Lambda(\hat{ev}) \circ \hat{\Lambda}(ev) = \Lambda(\hat{ev} \circ (\hat{\Lambda}(ev) \times id)) = \Lambda(ev) = id.$$

• $g \leq g' \Rightarrow \hat{\Lambda}(ev)(g) \leq \hat{\Lambda}(ev)(g')$: Consider $f_{g,g'} : C \to (D \to E)$. We have

$$\begin{array}{rcl} \hat{\Lambda}(ev)(g) & = & \hat{\Lambda}(ev \circ (g \times id)) \\ & = & \hat{\Lambda}(ev \circ ((f_{g,g'} \circ c) \times id)) \\ & = & \hat{\Lambda}(ev \circ (f_{g,g'} \times id) \circ (c \times id)) \\ & = & \hat{\Lambda}(ev \circ (f_{g,g'} \times id))(c)\ . \end{array}$$

Let $k = \hat{\Lambda}(ev \circ (f_{g,g'} \times id))$. Then $\hat{\Lambda}(ev)(g) = k(c)$ and $\hat{\Lambda}(ev)(g') = k(c')$; The conclusion follows. $\square$

5.3 Full Sub-CCC's of Acpo *

This section is devoted to Jung's classification theorem for algebraic dcpo's. Both L-domains and bifinite domains satisfy property m. We shall first prove that this property is necessary. Actually we prove that bicompleteness is necessary (which is stronger).

Definition 5.3.1 (bicomplete) *A partial order $(D, \leq)$ is called bicomplete if both D and $D^{op} = (D, \leq)$ are directed complete.*

Proposition 5.3.2 *If $(D, \leq)$ is bicomplete, then it satifies property m.*

Proof. By Zorn's lemma. Consider $A \subseteq_{fin} D$ and $x \in UB(A)$. Let $B = \downarrow x \cap UB(A)$. In B, every chain is a fortiori codirected in D, and its glb is clearly in B. Hence B has a minimal element, which is clearly a minimal upper bound of A. □

Jung's theorem relies on three propositions: 5.3.3, 5.3.6, and 5.3.7. We shall also need a theorem due to Markowsky, whose proof is given in section 5.5: a partial order is a dcpo if and only if any non-empty well-ordered subset of D has a lub.

Proposition 5.3.3 *A continuous dcpo D with continuous function space $D \to D$ is bicomplete.*

Proof. The proof is by contradiction. By proposition 5.5.1, we may assume that there exists a non-empty op-well-ordered subset B of D which has no glb. Let A be the (possibly empty) set of lower bounds of B. Notice that by the assumption on B we have $A \cap B = \emptyset$. We define the following function r on D by

$$r(x) = \begin{cases} x & \text{if } x \in A \\ \bigwedge\{b \in B \mid b \geq x\} & \text{if } x \notin A \quad \text{(where the glb is meant in } B) \ . \end{cases}$$

• r is well defined: We first prove that the set C of lower bounds in B of $\{b \in B \mid b \geq x\}$ is not empty. Since $x \notin A$, we have $x \not\leq b'$ for some $b' \in B$. A fortiori, if $b \geq x$, then $b \not\leq b'$. But B is a total order, hence $b' < b$, which proves $b' \in C$. Thus, since we assumed that B is op-well-ordered, the maximum of C exists and is $\bigwedge\{b \in B \mid b \geq x\}$. In particular we have $x \notin A \Rightarrow r(x) \in B$.

• $r \circ r = r$: $r(D) \in A \cup B$, and r is the identity on $A \cup B$.

• r is continuous: If $\bigvee \Delta \in A$, then $\Delta \subseteq A$, hence $r(\bigvee \Delta) = \bigvee \Delta = \bigvee r(\Delta)$. If $\bigvee \Delta \notin A$, then $\delta \not\leq b'$ for some $b' \in B$, $\delta \in \Delta$ (i.e., $\delta \notin A$). Hence $\Delta' \cap A = \emptyset$, where $\Delta' = \Delta \cap \uparrow \delta$, and $r(\Delta') \subseteq B$. Clearly $\bigvee \Delta' = \bigvee \Delta$ and $\bigvee r(\Delta') = \bigvee r(\Delta)$. Hence it is enough to prove $\bigvee r(\Delta') \geq r(\bigvee \Delta')$. We proceed by contradiction. Let $b' = r(\bigvee \Delta')$. If $\bigvee r(\Delta') \not\geq b'$, then a fortiori $r(\delta) \not\geq b'$ for any $\delta \in \Delta'$. But we have

$$r(\delta) \not\geq b' \Leftrightarrow b' \not\leq \{b \in B \mid b \geq \delta\} \Leftrightarrow \exists\, b_\delta \in B \;\; b' > b_\delta \geq \delta.$$

(For the last equivalence, notice that $\bigvee \Delta \notin A$ implies $b' \in B$, and recall that B is a chain.) Since B is op-well-ordered, the non-empty set $\{b_\delta \mid \delta \in \Delta\}$ has a maximum $b_{\delta''}$ for some $\delta'' \in \Delta$. But then we have:

$$\begin{array}{l} b' > b_{\delta''} \geq \bigvee \Delta \text{ by construction} \\ b' = r(\bigvee \Delta') \leq \{b \in B \mid b \geq \bigvee \Delta'\} \text{ implies } b' \leq b_{\delta''} \ . \end{array}$$

Contradiction. Hence r is continuous. We know from exercise 4.6.9 that $D' \to D'$ is a retract of $D \to D$, where $D' = r(D) = A \cup B$. It is continuous, by proposition 5.1.10. The rest of the proof consists in obtaining a contradiction to the continuity of $D' \to D'$. It goes via successive claims:

Claim 1. If $A \neq \emptyset$, then there exist $x' \ll x \in A$ and $y' \ll y \in A$ such that x' and y' have no upper bound in A.

We first show that $A' = \Downarrow A$ is not directed. By continuity of D, the lub of A' would be larger than any element of A, and still belong to A; but since A is not empty, we know that it has no maximum by the assumption on B: contradiction. Hence there exists $x'' \ll x \in A$ and $y'' \ll y \in A$ such that x'' and y'' have no upper bound in A'. Let x' and y' be obtained by interpolation: $x'' \ll x' \ll x$ and $y'' \ll y' \ll y$. Suppose that x' and y' have an upper bound z in A: then, by directedness of $\Downarrow z$, x'' and y'' would have an upper bound in A'. This completes the proof of claim 1.

Claim 2. $\exists f \in D' \to D' \;\; f \ll id$ and $f(B) \subseteq B$.

Claim 2 is obvious if A is empty, since then $B = D'$. If $A \neq \emptyset$, let x', y' be as in claim 1. Since $x' \ll x = \bigvee\{f(x) \mid f \ll id\}$, we have $x' \leq g(x)$ for some $g \ll id$. Similarly $y' \leq h(y)$. Let f be an upper bound of g, h in $\Downarrow id$. Then $x' \leq f(x)$ and $y' \leq f(y)$. Let b be an element of B. Then b is an upper bound of x and y, since $x, y \in A$. Hence $f(b) \geq f(x) \geq x'$. Similarly $f(b) \geq y'$. Thus, by claim 1, $f(b) \in D' \setminus A = B$. This completes the proof of claim 2.

Since B is op-well-ordered, we can define a predecessor function: $pred(b)$ is the maximum b' such that $b' < b$ (there is at least one such b', otherwise b would be a minimum of B, contradicting our assumption on B). Define, for each $b \in B$, a function $g_b : D' \to D'$ by

$$g_b(x) = \begin{cases} pred(f(x)) & \text{if } x \in B \text{ and } x \leq b \\ x & \text{otherwise .} \end{cases}$$

where f is given by claim 2.

Claim 3. 1. g_b is continuous, for all $b \in B$.

2. $\{g_b \mid b \in B\}$ is directed and has id as lub.

3. There is no g_b such that $f \leq g_b$.

Claim 3 contradicts $f \ll id$. Thus we are left with the proof of claim 3.

(3) If $f \leq g_b$, then $f(b) \leq g_b(b) = pred(f(b))$, a contradiction to the definition of *pred*.

(2) We prove that $\{g_b \mid b \in B\}$ is actually a chain by proving $b' \leq b \Rightarrow g_b \leq g_{b'}$. The only interesting case is when $x \in B$ and $b' < x \leq b$. Then $g_b(x) = pred(f(x)) \leq f(x) \leq x = g_{b'}(x)$. The equality $id = \bigvee\{g_b \mid b \in B\}$ follows from the remark that $g_{pred(b)}(b) = b$ for all $b \in B$.

(1) It is easily checked that g_b is monotonic. Let Δ be directed in D'. The interesting case is $\bigvee \Delta \in B$. Then $\delta \in B$ for some $\delta \in \Delta$, as otherwise we would have $\Delta \subseteq A$ (and hence $\bigvee \Delta \in A$). We can choose δ to be the maximum of $B \cap \Delta$, since B is well-ordered. Then $\bigvee \Delta = \delta \in \Delta$, and the continuity of g_b follows by monotonicity. □

Remark 5.3.4 *This proof generalises the situation presented in exercise 1.4.15.*

The hypotheses of the previous proposition are actually redundant.

Exercise 5.3.5 ($\rightarrow$-cont $\Rightarrow$ cont) *Show that a dcpo with continuous function space is continuous. Hint: use the claim proved in proposition 5.3.7.*

Proposition 5.3.6 (L/M) *Let D and E be algebraic cpo's satisfying property m. If $D \rightarrow E$ is continuous, then E is an L-domain or $\mathcal{K}(D) \models M$.*

Proof. By contradiction. Suppose that E is not an L-domain and that $\mathcal{K}(D) \not\models M$. Then (cf. exercise 5.2.14) there exists c in E, two compacts $a_1, a_2 \leq c$, and two distinct mub's b_1, b_2 of $\{a_1, a_2\}$ below c. Since $D \models m$, also $\mathcal{K}(D) \models m$ by lemma 5.1.6. Since $\mathcal{K}(D) \not\models M$, by lemma 5.2.8 there exist x_1 and x_2 in $\mathcal{K}(D)$ such that $MUB(x_1, x_2)$ is infinite. Assume moreover that $D \rightarrow E$ is continuous. Then we define $g : D \rightarrow E$ by

$$g(d) = \begin{cases} \bot & \text{if } d \not\geq x_1 \text{ and } d \not\geq x_2 \\ a_1 & \text{if } d \geq x_1 \text{ and } d \not\geq x_2 \\ a_2 & \text{if } d \not\geq x_1 \text{ and } d \geq x_2 \\ b_1 & \text{if } d \geq x_1 \text{ and } d \geq x_2 \ . \end{cases}$$

We leave the reader check that g is continuous and is a mub of the step functions $x_1 \rightarrow a_1$ and $x_2 \rightarrow a_2$. In particular g is compact. We shall contradict the compactness of g. We define f by replacing b_1 by c in the last line of the definition of g. Clearly $g \leq f$. We shall exhibit a directed set of functions which has f as lub, but none of which dominates g. For each finite subset A of $MUB(x_1, x_2)$, define a function $f_A : D \rightarrow E$ by

$$f_A(d) = \begin{cases} \bot & \text{if } d \not\geq x_1 \text{ and } d \not\geq x_2 \\ a_1 & \text{if } d \geq x_1 \text{ and } d \not\geq x_2 \\ a_2 & \text{if } d \not\geq x_1 \text{ and } d \geq x_2 \\ b_2 & \text{if } d \in MUB(x_1, x_2)\backslash A \\ c & \text{otherwise} \ . \end{cases}$$

We have to check that the f_A's are continuous, and form a directed set with lub f. We leave this to the reader, with the following hint: to prove the continuity, observe that

$$x_1, x_2 \text{ compact}, \bigvee \Delta \in MUB(x_1, x_2) \Rightarrow \Delta \text{ has a maximum.}$$

Suppose $g \leq f_A$ for some A, and pick $d \in MUB(x_1, x_2)\backslash A$. We should have $b_1 = g(d) \leq f_A(d) = b_2$. Since we assumed $b_1 \neq b_2$, this contradicts the minimality of b_2. □

Proposition 5.3.7 *Let D be a dcpo with algebraic function space and such that $\mathcal{K}(D) \models M$. Then D is bifinite.*

Proof. Suppose that U^∞ is infinite, for some finite $A \subseteq \mathcal{K}(D)$. We set $B^0 = A, B^{n+1} = U^{n+1}(A)\backslash U^n(A)$. By our assumption, for each n, $B^{n+1} \neq \emptyset$. We construct a tree in the following way. The nodes are finites sequences $b_n \ldots b_0$ where $b_i \in B^i$ for all i, and where, for each $i < n$, b_i belongs to a subset of $U^i(A)$ of which b_{i+1} is a mub. The root is the empty sequence, the predecessor of $b_n \ldots b_0$ is $b_{n-1} \ldots b_0$. By construction, and by property M, this is a finitely branching tree. We show that for any $b \in U^\infty(A)$ there exists a node $b_n \ldots b_0$ such that $b = b_n$, which entails that the tree is infinite. Let n be minimum such that $b \in U^n(A)$; we have a fortiori $b \in B^n$. By definition of $U^n(A)$

we can find a subset B of $U^{n-1}(A)$ of which b is a mub. If B were also a subset of $U^{n-2}(A)$, then we would not have $b \in B^n$. Hence we can build $b_{n-1} \dots b_0$ as desired. Since the tree is infinite and finitely branching, by König's lemma it has an infinite branch $\dots b_n \dots b_0$, which in particular forms an infinite strictly increasing sequence in $U^\infty(A)$. Now we use the algebraicity of $D \to D$. We have

$$id = \bigvee\{f \mid f \text{ is compact and } f \leq id\}.$$

In particular $a = \bigvee\{f(a) \mid f \text{ is compact and } f \leq id\}$, for a compact, implies $a = f(a)$ for some f. By directedness we can find a compact $f \leq id$ for which $\forall\, a \in A\ (a = f(a))$. We claim:

$$\forall\, a \in U^\infty(A)\ \ (a = f(a)).$$

Suppose that we know $\forall\, a \in U^n(A)\ \ (a = f(a))$. Let a be a mub of $A' \subseteq U^n(A)$. Then $a \geq f(a) \geq f(A') = A'$ implies $f(a) = a$.

By the claim we have $f(b_n) = b_n$ for all n, and $f(c) = c$ follows by continuity for $c = \bigvee b_n$. We shall get a contradiction by proving the following claim:

Claim. If D is a dcpo which has a continuous function space, and if $f \ll id$, then $f(d) \ll d$ for all d.

If the claim is true, then $c = f(c) \ll c$, hence c is compact. But a lub of a strictly increasing sequence is not compact: contradiction.

We prove the claim by appealing again to a "retract" trick. Let Δ be such that $d \leq \bigvee \Delta$. Set $z = \bigvee \Delta$. Since $\downarrow z = D'$ is a retract of D (cf. lemma 5.1.9), $D' \to D'$ is continuous, as a retract of $D \to D$. We show that $f \ll id$ also holds in $D' \to D'$ (notice that since $f \leq id$, f maps D' into D'). For this, it is enough by lemma 5.1.3 to consider a directed $\Delta' \subseteq D' \to D'$ such that $id = \bigvee \Delta'$ in $D' \to D'$. Each g in Δ' can be extended to D by setting

$$g(x) = x \text{ whenever } x \not\leq z.$$

Hence Δ' can be viewed as a directed subset in $D \to D$, and has clearly id as lub there too. It follows that $f \leq g$ for some $g \in \Delta'$, and the inequality holds a fortiori in $D' \to D'$. We have proved $f \ll id$ in $D' \to D'$. Consider the family of constant functions $x \mapsto \delta$ for each $\delta \in \Delta$. It forms a directed set with lub the constant $x \mapsto z$. We have $(x \mapsto z) \geq id$ (in $D' \to D'$). Hence $f \leq (x \mapsto \delta)$ for some $\delta \in \Delta$. In particular $f(d) \leq \delta$. This ends the proof of the claim and of the proposition. □

Theorem 5.3.8 *The categories* **Bif** *and* **L** *are the two maximal cartesian closed full subcategories of* **Acpo**.

Proof. We have already proved that **Bif** and **L** are cartesian closed. By proposition 5.2.17, if **C** is a cartesian closed full subcategory of **Acpo**, we know that the exponents of **C** are the exponents of **Cpo**. Let $D \in \mathbf{C}$. Since both D and $D \to D$ are algebraic, D is bicomplete by proposition 5.3.3, hence, by proposition 5.3.2, $D \models m$. Thus we can apply proposition 5.3.6 to any $D, E \in \mathbf{C}$. Combining with proposition 5.3.7 applied to D, we get, for any $D, E \in \mathbf{C}$:

$$\mathcal{K}(D) \text{ is bifinite or } E \text{ is an algebraic L-domain.}$$

Suppose now that **C** is neither a subcategory of **L** nor a subcategory of **Bif**. Then there is an object D of **C** which is not bifinite and an object E of **C** which is not an L-domain: contradiction. □

The analysis is simplified in the case of ω-algebraic cpo's, thanks to the following proposition.

Proposition 5.3.9 *If D is an (algebraic) cpo and $D \to D$ is ω-algebraic, then $\mathcal{K}(D) \models M$.*

PROOF. We already know from proposition 5.3.3 that D is bicomplete, hence that $\mathcal{K}(D) \models m$. Assume that $MUB(a_1, a_2)$ is infinite for some compacts a_1, a_2. We build uncountably many mub's of $a_1 \to a_1$ and $a_2 \to a_2$. Since they are all compact, this contradicts the ω-algebraicity of D. We pick two distinct mub's b_1, b_2 of a_1, a_2. For any $S \subseteq MUB(a_1, a_2)$, we define $f_S : D \to D$ by

$$f_S(d) = \begin{cases} \bot & \text{if } d \not\geq a_1 \text{ and } d \not\geq a_2 \\ a_1 & \text{if } d \geq a_1 \text{ and } d \not\geq a_2 \\ a_2 & \text{if } d \not\geq a_1 \text{ and } d \geq a_2 \\ b_1 & \text{if } \exists s \in S \ \ d \geq s \\ b_2 & \text{if } \exists s \in MUB(a_1, a_2) \backslash S \ \ d \geq s\,. \end{cases}$$

To see that f_S is well-defined, we use the fact that D is an L-domain by proposition 5.3.6: if $d \in MUB(a_1, a_2)$, there is exactly one mub of a_1, a_2 below d. We omit the rest of the proof. □

Exercise 5.3.10 (Smyth) *Show that the category ω**Bif** of ω-bifinite cpo's and continuous functions is the largest cartesian closed full subcategory of ω**Acpo**.*

5.4 Full Sub-CCC's of Adcpo *

In this section, we present a brief account of Jung's results in the case of algebraic dcpo's, that is, we relax the assumption that the domains have a $\bot$. There are four maximal cartesian closed full subcategories of **Adcpo**. The duplication with respect to to the previous section comes from the following discriminating proposition, which is "orthogonal" to the discriminating proposition 5.3.6.

Proposition 5.4.1 (F/U) *Let D and E be continuous dcpo's satisfying property m. If $D \to E$ is continuous, then D has finitely many minimal elements or E is a disjoint union of cpo's.*

PROOF. By contradiction. We thus assume that D has infinitely many minimal elements and that E is not a disjoint union of cpo's. First notice that the collection of minimal elements of E can be alternatively described as $MUB(\emptyset)$. Hence by property m, E can be described as $E = \bigcup\{\uparrow e \mid e \text{ is a minimal element of } E\}$. Our assumption implies that there exists an upper bounded pair $\{e_1, e_2\}$ of distinct minimal elements of E. By property m one can find a mub e of e_1, e_2. The constant function $x \mapsto e_1$ is minimal, hence compact in $D \to E$ (minimal implies compact in a continuous dcpo). For any finite set A of minimal elements of D, we define a function f_A by

$$f_A(x) = \begin{cases} e & \text{if } x \in \uparrow A \\ e_2 & \text{otherwise .} \end{cases}$$

This defines a directed family of continuous functions which has $x \mapsto e$ as lub (the monotonicity of f_A follows from $e_2 \leq e$). Hence one must have $(x \mapsto e_1) \leq f_A$ for some A, which entails $e_1 \leq e_2$. But e_2 is minimal and $e_1 \neq e_2$: contradiction. □

The property of finiteness of the set of minimal elements is not strong enough to be closed under function spaces. But a strengthening will do.

Definition 5.4.2 (root) *Given a dcpo D , the set $U^\infty(\emptyset)$ is called the root of D.*

Proposition 5.4.3 *Let D be a (continuous) dcpo such that $D \to D$ satisfies property m and has finitely many minimal elements. Then D has a finite root.*

PROOF. With each element d of the root we associate the canonical retraction r_d onto $\downarrow d$ defined in lemma 5.1.9. We show that if $d \neq d'$, then r_d and $r_{d'}$ have no common lower bound. We can assume say $d \not\leq d'$. Then if $f \leq r_d, r_{d'}$, we have:

$$\begin{array}{lll} f \leq r_d & \Rightarrow & f(d) \leq d \\ f \leq r_{d'} & \Rightarrow & f(d) \leq r_{d'}(d) = d' \ . \end{array}$$

In fact, because d is in the root of D, $f(d) \leq d$ implies $f(d) = d$. This is obvious if d is a minimal element of D, and the property propagates to all elements of the root (cf. the proof of proposition 5.3.7). Hence $d = f(d) \leq d'$, contradicting the assumption. Since $D \to D \models m$, there exists a minimal function m_d below each r_d. The m_d's are all distinct, since $m_d = m_{d'}$ would entail that $\{r_d, r_{d'}\}$ has a lower bound. Hence if the root of D is infinite, then $D \to D$ has infinitely many minimal elements: contradiction.
□

Quite orthogonally, the results of the previous section can be exploited.

Lemma 5.4.4 (∀L/∀M) *Let D and E be algebraic dcpo's satisfying property m. If $\uparrow e$ is not an L-domain and if $\mathcal{K}(\uparrow d) \not\models M$, for some compacts e, d of E, D, respectively, then $D \to E$ is not continuous.*

PROOF. Obvious consequence of proposition 5.3.6 and exercise 4.6.9. □

Corollary 5.4.5 (∀L/∀B) *If D and E are algebraic dcpo's satisfying property m and if $D \to E$ is an algebraic dcpo, then either all basic Scott opens $\uparrow d$ ($d \in \mathcal{K}(D)$) are bifinite or all $\uparrow e$'s ($e \in \mathcal{K}(E)$) are L-domains.*

PROOF. Lemma 5.4.4 is applicable by proposition 5.3.3. Suppose that a Scott-open $\uparrow e$ is not an L-domain: then, by the lemma, $\forall\, d\ \mathcal{K}(D) \models M$. We conclude by noticing that proposition 5.3.7 is applicable to $\uparrow d$ thanks to the following claim.

Claim. If $D \to D'$ is algebraic and d, d' are compact, then $\uparrow d \to \uparrow d'$ is algebraic.

The claim follows from the observation that $\uparrow d \to \uparrow d'$ is order-isomorphic to $\uparrow (d \to d')$. □

Theorem 5.4.6 *There are exactly four maximal cartesian closed full subcategories of* **Adcpo***, with the following respective classes of objects:*

(**UL**)	*the disjoint unions of algebraic L-domains,*
(**UB**)	*the disjoint unions of bifinite cpo's,*
(**FL**)	*the dcpo's with a finite root whose basic Scott opens $\uparrow d$ are algebraic L-domains,*
(**FB**)	*the profinite dcpo's.*

Proof. We omit the verifications that these four categories are cartesian closed. We also leave it to the reader to verify that the profinite dcpo's are the dcpo's with a finite root such that all $\uparrow d$'s are bifinite. The proof proceeds like the proof of theorem 5.3.8, exploiting not only the discrimination L/M (in its variant ∀L/∀B), but also the discrimination F/U. We use **B**, **L**, **F**, **U** as abbreviations for bifinite, L-domain, finite root, disjoint union of cpo's, respectively. Let **C** be a cartesian closed full subcategory of **Adcpo**. By corollary 5.4.5 on one hand, and by combining proposition 5.4.1 (applied to $D \to D$ and E) and 5.4.3 on the other hand, we get, as in the proof of theorem 5.3.8:

$$(\mathbf{C} \subseteq \mathbf{B} \text{ or } \mathbf{C} \subseteq \mathbf{L}) \text{ and } (\mathbf{C} \subseteq \mathbf{F} \text{ or } \mathbf{C} \subseteq \mathbf{U}).$$

Assume now that **C** is not included in any of **UL**, **UB** and **FL**. Let $D_1, D_2, D_3 \in \mathbf{C}$ witness these non-inclusions, and let D be an arbitrary object of **C**. Then we proceed by cases:

- $D_1 \notin \mathbf{U}$: Then $D, D_3 \in \mathbf{F}$ since $\mathbf{C} \subseteq \mathbf{F}$ or $\mathbf{C} \subseteq \mathbf{U}$. By non-inclusion, we have $D_3 \notin \mathbf{L}$, which implies $D \in \mathbf{B}$ since $\mathbf{C} \subseteq \mathbf{B}$ or $\mathbf{C} \subseteq \mathbf{L}$.

- $D_1 \notin \mathbf{L}$: Similarly we deduce that $D \in \mathbf{B}$ and $D \in \mathbf{F}$, using witness D_2. □

Exercise 5.4.7 *Show that the category* ω**Prof** *of ω-profinite dcpo's and continuous functions is the largest cartesian closed full subcategory of* ω**Adcpo**.

5.5 Completeness of Well-Ordered Lub's *

We prove the theorem of [Mar76] which we used in the proof of proposition 5.3.3. The proof assumes some familiarity with ordinals and cardinals. We simply recall that every set can be well-ordered, that ordinals are canonical representatives of isomorphisms classes of well-orderings, and that cardinals are the least ordinals of a given cardinality. We write $\sharp\Delta$ for the cardinal of Δ

Proposition 5.5.1 (Markowsky) *1. A partial order D is a dcpo iff any non-empty chain of D has a lub.*

2. Let D be a partial order. D is a dcpo iff any non-empty well-ordered subset of D has a lub.

PROOF. (2) clearly implies (1), but (1) serves as a stepping stone to (2).

(1) We first prove the following claim.

Claim 1. Let Δ be an infinite directed set. There is an ordinal and a family $\{\Delta_\alpha\}_{\alpha<\gamma}$ of directed subsets of Δ indexed over the cardinal γ of Δ, such that:

(A) $\alpha < \beta \Rightarrow \Delta_\alpha \subset \Delta_\beta$,
(B) Δ_α is finite if α is finite, $\sharp\Delta_\alpha = \sharp\alpha$ if α is infinite, and
(C) $\Delta = \bigcup_{\alpha<\gamma} \Delta_\alpha$.

In order to prove the claim, we first fix a choice u_F of an upper bound of F for any $F \subseteq_{fin} \Delta$. Let $\{x_\alpha\}_{\alpha<\gamma}$ be a bijective indexing of Δ. We construct Δ_α and we prove properties (A), (B), and (C) together by transfinite induction:

- $\alpha = 0$: $\Delta_0 = \{x_0\}$.
- $\alpha = \beta + 1$: We set:

$$\begin{array}{rcl} \Delta_{\alpha,0} & = & \Delta_\beta \cup \{x_\delta\} \text{ where } \delta \text{ is the least index such that } x_\delta \in \Delta\backslash\Delta_\beta \\ \Delta_{\alpha,i+1} & = & \Delta_{\alpha,i} \cup \{u_F \mid F \subseteq_{fin} \Delta_{\alpha,i}\} \\ \Delta_\alpha & = & \bigcup_{i\in\omega} \Delta_{\alpha,i} \,. \end{array}$$

(It will be part of the proof to show that indeed $\Delta\backslash\Delta_\beta$ is non-empty.)

- α is a limit ordinal: We set $\Delta_\alpha = \bigcup_{\beta<\alpha} \Delta_\beta$.

By construction, the Δ_α's are directed and property (A) holds. Property (C) can be rephrased as $\Delta = \Delta_\gamma$, and thus follows from property (B) by minimality of γ. Property (B) clearly holds for $\alpha = 0$. For finite ordinals $\alpha = \beta + 1$, the definition of Δ_α boils down to $\Delta_\alpha = \Delta_{\alpha,0} \cup \{u_{\Delta_{\alpha,0}}\}$, which is therefore finite. For infinite ordinals, the limit ordinal case is obvious. If $\alpha = \beta + 1$, then $\sharp\Delta_{\alpha,i} = \sharp\Delta_\beta$ for any i. Hence

$$\sharp\Delta_\alpha \leq \sharp(\Delta_\beta \times \omega) = \sharp\Delta_\beta = \sharp\beta = \sharp\alpha.$$

This completes the proof of (B). We next prove that property (B) at β ensures the well-definedness of $\Delta_{\beta+1}$. This is clear for finite β. Suppose thus that β is infinite, that $\beta + 1 = \alpha \leq \gamma$, and that $\Delta = \Delta_\beta$. Then $\gamma = \sharp\Delta = \sharp\Delta_\beta = \sharp\beta$, which contradicts the minimality of γ since $\beta < \gamma$.

We prove (1) by contradiction. Let γ be the least ordinal for which there exists a directed Δ of cardinal γ such that $\bigvee \Delta$ does not exist, and let $\{\Delta_\alpha\}$ be as in the claim. Then $\bigvee \Delta_\alpha$ exists for each $\alpha < \gamma$, by the minimality of γ. The collection $\{\bigvee \Delta_\alpha \mid \alpha < \gamma\}$ forms a chain by (A), and its lub is the lub of Δ by (C). Contradiction.

(2) It is enough, by (1), to show that for any chain $X \subseteq D$ there exists a subset Y of X which is well-ordered by the restriction of the order of D and is such that $\bigvee Y = \bigvee X$. Let $\{x_\alpha\}_{\alpha<\delta}$ be a bijective indexing of X by an ordinal δ. We assign to each $\alpha < \delta$ an element $y_\alpha \in X$ and a subset X_α of X, or "stop" as follows:

- $\alpha = 0$: $y_0 = x_0$ and $X_0 = X \backslash \{x \in X \mid x \leq y_0\}$.
- $\alpha = \beta + 1$: If $X_\beta = \emptyset$, then "stop"; otherwise, set:

$$\begin{array}{l} y_\alpha = x_{\delta_0}, \text{ where } \delta_0 \text{ is the least element of } \{\delta \mid x_\delta \in X_\beta\} \\ X_\alpha = X_\beta \backslash \{x \in X_\beta \mid x \leq y_\alpha\} \ . \end{array}$$

- α is a limit ordinal: If $\bigcap_{\beta<\alpha} X_\beta = \emptyset$, then "stop"; otherwise, set:

$$\begin{array}{l} y_\alpha = x_{\delta_0}, \text{ where } \delta_0 \text{ is the least element of } \{\delta \mid x_\delta \in \bigcap_{\beta<\alpha} X_\beta\} \\ X_\alpha = \bigcap_{\beta<\alpha} X_\beta \backslash \{x \in \bigcap_{\beta<\alpha} X_\beta \mid x \leq y_\alpha\} \ . \end{array}$$

Claim 2. The following properties hold for every α at which y_α, X_α are defined:

$$\begin{array}{ll} \text{(D)} & x \in X \backslash X_\alpha \Leftrightarrow \exists \beta \leq \alpha \ \ x \leq y_\beta, \\ \text{(E)} & \gamma < \alpha \Rightarrow y_\gamma < y_\alpha, \\ \text{(F)} & x_\alpha \leq y_\alpha. \end{array}$$

The three properties are proved by induction on α.

(D) If $\alpha = 0$, we have $x \notin X_0 \Leftrightarrow x \leq x_0$. If $\alpha = \beta + 1$, then (A) follows by induction from ($x \notin X_\alpha \Leftrightarrow (x \notin X_\beta$ or $x \leq y_\alpha$)). If α is a limit ordinal, then (A) similarly follows from

$$x \notin X_\alpha \Leftrightarrow (\exists \beta < \alpha \ \ x \notin X_\beta) \text{ or } x \leq y_\alpha.$$

(E) If $\alpha = \beta + 1$, then by induction it is enough to check $y_\beta < y_\alpha$. Suppose $y_\alpha \leq y_\beta$. Then $y_\alpha \notin X_\beta$ by (D), contradicting the definition of α. If α is a limit ordinal, if $\gamma < \alpha$, and if $y_\alpha \leq y_\gamma$, we get a similar contradiction from $y_\alpha \in \bigcap_{\gamma<\alpha} X_\beta$.

(F) If $\alpha = 0$, then a fortiori $y_0 = x_0$. If $\alpha = \beta + 1$, then for any $\gamma \leq \beta$ we have by induction $x_\gamma \leq y_\gamma$, hence $x_\gamma \notin X_\alpha$ by (D), which, by definition of δ_0 entails $\delta_0 \geq \alpha$. If $\delta_0 = \alpha$ then a fortiori $x_\alpha \leq y_\alpha$. If $\delta_0 > \alpha$, then $x_\alpha \notin X_\beta$ by minimality of δ_0, and

$$\begin{array}{ll} x_\alpha \leq y_\gamma & \text{for some } \gamma \leq \beta, \text{ by (D)} \\ y_\gamma \leq y_\alpha & \text{by (E)} \ . \end{array}$$

We use a similar reasoning if α is a limit ordinal. This completes the proof of claim 2.

The set $Y = \{y_\alpha \mid y_\alpha \text{ is defined}\}$ is well-ordered by (E). Since $Y \subseteq X$, we are left to show $X \leq \bigvee Y$. This follows from (F) if y_α is defined for any index. Otherwise, the construction of the y_β's has reached a "stop" at some α. There are two cases:

- $\alpha = \beta + 1$ and $X_\alpha = \emptyset$: Then $X \leq \bigvee Y$ follows from (D).
- α is a limit ordinal and $\bigcap_{\beta<\alpha} X_\beta = \emptyset$. Let $x \in X$, then $x \notin X_\beta$ for some $\beta < \alpha$, and the conclusion again follows from (D). □

Chapter 6

The Language PCF

We have provided semantics for both typed and untyped λ-calculus. In this chapter we extend the approach to typed λ-calculus with fixpoints (λY-calculus), we suggest formal ways of reasoning with fixpoints, and we introduce a core functional language called PCF, originally due to Scott [Sco93], and thoroughly studied by Plotkin [Plo77]. PCF has served as a basis for much of the theoretical work in semantics. We prove the adequacy of the interpretation with respect to the operational semantics and we discuss the full-abstraction problem, which has triggered a lot of research, both in syntax and semantics.

In section 6.1 we introduce the notion of *cpo-enriched* CCC, which serves to interpret the λY-calculus. In section 6.2, we introduce fixpoint induction and show an application of this reasoning principle. In section 6.3, we introduce the language PCF, we define its standard denotational semantics and its operational semantics, and we show a computational adequacy property: the meaning of a closed term of base type is defined if and only if its evaluation terminates. In section 6.4 we address a tighter correspondence between denotational and operational semantics, known as full abstraction property. We show how a fully abstract model of PCF can be obtained, by means of a suitable quotient of an (infinite) term model of PCF. In section 6.5, we introduce Vuillemin's sequential functions, which capture first-order PCF definability.

6.1 The λY-Calculus

The λY-calculus is the typed λ-calculus extended with a family of constants Y^σ of type $(\sigma \to \sigma) \to \sigma$ for each type σ (Y for short), with the following reduction rule:

$$(Y) \quad YM \to M(YM).$$

It is also convenient to consider a special constant Ω^σ at each type (to be interpreted by $\perp$).

Definition 6.1.1 (cpo-enriched-CCC) *A cartesian closed category* $\mathbf{C}$ *is called a cpo-enriched cartesian closed category if all homsets* $\mathbf{C}[a,b]$ *are cpo's, if composition is continuous, if pairing and currying are monotonic, and if the following strictness conditions hold (for all* f *of the appropriate type):*

$$\bot \circ f = \bot \quad ev \circ \langle \bot, f\rangle = \bot .$$

Remark 6.1.2 *Notice that our definition of a cpo-enriched CCC involves the cartesian closed structure of the category: thus in our terminology a cpo-enriched CCC is not just a cpo-enriched category which happens to be cartesian closed.*

Lemma 6.1.3 *In a cpo-enriched CCC pairing and currying are continuous.*

Proof. We consider the case of currying only (the argument is the same for pairing). In order to prove $\Lambda(\bigvee \Delta) = \bigvee\{\Lambda(f) \mid f \in \Delta\}$, it is enough to check that $\bigvee\{\Lambda(f) \mid f \in \Delta\}$ satisfies the characterizing equation:

$$ev \circ (\bigvee\{\Lambda(f) \mid f \in \Delta\} \times id) = \bigvee \Delta .$$

The monotonicity of Λ guarantees that $\{\Lambda(f) \mid f \in \Delta\}$ is directed. Hence by continuity of composition (and pairing) we have

$$ev \circ (\bigvee\{\Lambda(f) \mid f \in \Delta\} \times id) = \bigvee\{ev \circ (\Lambda(f) \times id) \mid f \in \Delta\} = \bigvee \Delta .$$

$\square$

The following definition was first given by Berry [Ber79].

Definition 6.1.4 (least fixpoint model) *A least fixpoint model is a cpo-enriched cartesian closed category where* Ω *and* Y *are interpreted as follows:*

$$\begin{array}{rcl} [\![\Omega]\!] & = & \bot \\ [\![Y]\!] & = & \bigvee_{n<\omega} [\![\lambda f.f^n\Omega]\!] \end{array}$$

where $M^n\Omega = M(\cdots(M\Omega)\cdots)$, *n times.*

The fact that the sequence of the $[\![\lambda f.f^n\Omega]\!]$'s is increasing follows from the assumptions of monotonicity in the definition of cpo-enriched CCC.

Proposition 6.1.5 *In a least fixpoint model, the* (Y) *rule is valid.*

Proof. Exploiting the continuity of the composition and pairing, we have

$$\begin{array}{lll} [\![YM]\!] & = ev \circ \langle \bigvee_{n<\omega}[\![\lambda f.f^n\Omega]\!], [\![M]\!]\rangle & = \bigvee_{n<\omega}[\![M^n\Omega]\!] \\ [\![M(YM)]\!] & = ev \circ \langle [\![M]\!], \bigvee_{n<\omega}[\![M^n\Omega]\!]\rangle & = \bigvee_{n<\omega}[\![M^{n+1}\Omega]\!] . \end{array}$$

$\square$

Proposition 6.1.6 **Cpo** *is a cpo-enriched CCC. In particular, for any cpo D, $Fix : (D \to D) \to D$, defined by $Fix(f) = \bigvee_{n\in\omega} f^n(\bot)$ is continuous.*

Exercise 6.1.7 *Consider the extension of the simply typed λ-calculus with a collection of constants Y_n $(n \geq 0)$ and rules:*

$$(Y_n) \quad Y_{n+1}M \to M(Y_nM).$$

Prove that the system obtained by adding these rules to the β-rule is strongly normalizing. Hint: adapt the proof of theorem 2.2.9.

Exercise 6.1.8 *Let $\mathbf{C}$ be a cpo-enriched cartesian-closed category such that currying is strict, i.e. $\Lambda(\bot) = \bot$. Adapt the definition of Böhm tree given in chapter 2 to the λY-calculus by setting $\omega(\lambda\vec{x}.YM_1 \dots M_p) = \Omega$ $(p \geq 1)$. Show that the following holds:*

$$[\![M]\!] = \bigvee\{[\![\omega(N)]\!] \mid M \to^\star N\}.$$

Hints: (1) Extend the meaning function by setting: $[\![Y_n]\!] = [\![\lambda f.f^n\Omega]\!]$. (2) Show that $[\![M]\!] = \bigvee_{n<\omega}[\![M_n]\!]$, where M_n is obtained from M by replacing all its occurrences of Y by Y_n. (3) Consider the normal form N_0 of M_n. Show that it is the result of replacing all the occurrences of Y by Y_0 in a reduct N of M, and use the strictness assumptions to show $[\![N_0]\!] = [\![\omega(N)]\!]$.

Exercise 6.1.9 *A class of continuous functionals $F_D : (D \to D) \to D$, ranging over all cpo's D, is called a fixpoint operator if $F_D(f)$ is a fixpoint of f, for any D and $f : D \to D$. It is called moreover uniform if the following holds:*

$$\forall f : D \to D, g : E \to E, h : D \to E \quad (h \circ f = g \circ h \Rightarrow h(F_D(f)) = F_D(g))$$

where h is supposed strict. Show that Fix is the unique uniform fixpoint operator.

6.2 Fixpoint Induction

A key motivation for denotational semantics lies in its applications to the proof of properties of programs. An important tool is fixpoint induction. If we want to show that a property $\mathcal{P}$ holds of a term YM, then, knowing that the meaning of YM is the lub of the sequence $\bot, F(\bot), F(F(\bot)), \dots$, where F is the meaning of M, it is enough to check the following properties.

- The meaning of property $\mathcal{P}$, say $[\![\mathcal{P}]\!]$, is a sub-dcpo of the domain D associated to the type of YM: in full, $[\![\mathcal{P}]\!]$ is closed under limits of non-decreasing chains; such predicates are called *inclusive*.
- Both properties $\bot \in [\![\mathcal{P}]\!]$ and $\forall x(x \in [\![\mathcal{P}]\!] \Rightarrow F(x) \in [\![\mathcal{P}]\!])$ hold.

This is summarised by the following inference rule, known as fixpoint induction principle

$$\frac{\mathcal{P} \textit{ inclusive} \quad \bot \in \mathcal{P} \quad \forall x(x \in \mathcal{P} \Rightarrow F(x) \in \mathcal{P})}{Fix(F) \in \mathcal{P}}$$

where $\mathcal{P} \subseteq D$, for a given cpo D, and $F : D \to D$ is continuous. Such an inference rule is a step towards mechanizing proofs of programs. What is needed next is a formal theory for proving that some predicates are inclusive (see exercise 6.2.2).

Remark 6.2.1 *The sufficiency of the above conditions, hence the validity of fixpoint induction, follows immediately from the Peano induction principle on natural numbers. Thus, mathematically speaking, it is not strictly necessary to formulate the above principle explicitly. One can prove $\perp \in [\![\mathcal{P}]\!]$, $F(\perp) \in [\![\mathcal{P}]\!]$, $F(F(\perp)) \in [\![\mathcal{P}]\!], \ldots$ and use Peano induction to conclude (if $[\![\mathcal{P}]\!]$ is inclusive). The interest of stating an explicit induction principle is to enable one: (1) to write lighter proofs, as $F(x)$ is easier to write than $F(F(\ldots(\perp)\ldots))$; and (2) to insert it in a mechanical proof-checker like* LCF *[Pau87].*

Exercise 6.2.2 *(1) Let D be a cpo. Show that $\emptyset$ and D are inclusive predicates in D. Show that $x = x$ and $x \leq y$ are inclusive in $D \times D$. (2) Let D and E be cpo's and $f : D \to E$ be continuous. Let $\mathcal{R}$ be inclusive in E. Show that $f^{-1}(\mathcal{R})$ is inclusive. (3) Let D be a cpo and $\mathcal{P}, \mathcal{Q}$ be inclusive in D. Then show that $\mathcal{P} \cap \mathcal{Q}$ and $\mathcal{P} \cup \mathcal{Q}$ are inclusive. (4) Let D and E be cpo's and $\mathcal{R}$ be inclusive on $D \times E$ in its first argument. Show that the predicate $\forall y \; (x\mathcal{R}y)$ is inclusive on D. (5) Let D and E be dcpo's and $\mathcal{P}, \mathcal{Q}$ be inclusive in D, E respectively. Show that $\mathcal{P} \times \mathcal{Q}$ is inclusive in $D \times E$, and that $\mathcal{P} \to \mathcal{Q}$ is inclusive in $D \to E$, where $\mathcal{P} \to \mathcal{Q} = \{f : D \to E \mid \forall d \in \mathcal{P} \;\; f(d) \in \mathcal{Q}\}$.*

As an illustration, we carry in some detail the proof of the following proposition, due to Bekič, which shows that n-ary fixpoints can be computed using unary fixpoints.

Proposition 6.2.3 *Let D, E be cpo's and $f : D \times E \to D$, $g : D \times E \to E$ be continuous. Let (x_0, y_0) be the least fixpoint of $\langle f, g\rangle$. Let x_1 be the least fixpoint of $f \circ \langle id, h\rangle$, where $h = Fix \circ \Lambda(g) : D \to E$ (hence $h(x_1)$ is such that $g(x_1, h(x_1)) = h(x_1)$). Then $x_0 = x_1$ and $y_0 = h(x_1)$.*

PROOF. $(x_0, y_0) \leq (x_1, h(x_1))$: Define the predicate $\mathcal{Q}(u, v)$ as $(u, v) \leq (x_1, h(x_1))$. This is an inclusive predicate (see exercise 6.2.2). Thus we may start the fixpoint induction engine. The base case is obvious. Suppose that $(u, v) \leq (x_1, h(x_1))$. We want to show that $f(u, v) \leq x_1$ and $g(u, v) \leq h(x_1)$. By monotonicity we have $f(u, v) \leq f(x_1, h(x_1))$ and $g(u, v) \leq g(x_1, h(x_1))$. But $f(x_1, h(x_1)) = x_1$ since x_1 is a fixpoint of $f \circ \langle id, h\rangle$. This settles the inequality $f(u, v) \leq x_1$. By definition of h, we have $h(x_1) = g(x_1, h(x_1))$, which settles the other inequality.

$(x_1, h(x_1)) \leq (x_0, y_0)$: We define a second predicate $\mathcal{R}(u)$ as $(u, h(u)) \leq (x_0, y_0)$. We leave the base case aside for the moment, and suppose that $\mathcal{R}(u)$ holds. We have to prove $\mathcal{R}(f(u, h(u)))$. We have $f(u, h(u))) \leq f(x_0, y_0) = x_0$ by monotonicity, and by definition of (x_0, y_0). We need a little more work to obtain $h(f(u, h(u))) \leq y_0$. It is enough to check $h(x_0) \leq y_0$. By definition of h, y_0 we

n	$: \iota$	$(n \in \omega)$
tt, ff	$: o$	
$succ, pred$	$: \iota \to \iota$	
$zero?$	$: \iota \to o$	
if then else	$: o \to \iota \to \iota \to \iota$	
if then else	$: o \to o \to o \to o$	
Ω	$: \sigma$	for all σ
Y	$: (\sigma \to \sigma) \to \sigma$	for all σ

Figure 6.1: The constants of PCF

have $h(x_0) = g(x_0, h(x_0))$, and $y_0 = g(x_0, y_0)$. We define a third inclusive predicate $\mathcal{S}(u)$ as $u \leq y_0$, remembering that $h(x_0)$ is the least fixpoint of $\Lambda(g)(x_0)$. The base case is obvious. Suppose that $u \leq y_0$. Then $g(x_0, u) \leq g(x_0, y_0) = y_0$. Hence fixpoint induction with respect to $\mathcal{S}$ allows us to conclude $h(x_0) \leq y_0$. We are left with the base case with respect to $\mathcal{R}$: $(\perp, h(\perp)) \leq (x_0, y_0)$ follows a fortiori from $h(x_0) \leq y_0$. $\square$

Let us shortly analyse this proof: we have focused in turn on each of the least fixpoint operators involved in the statement, exploiting just the fact that the other least fixpoints are fixpoints.

6.3 The Programming Language PCF

Scott [Sco93], and then Plotkin [Plo77], introduced a particular simply typed λY-calculus, PCF, which has become a quite popular language in studies of semantics. It has two basic types: the type ι of natural numbers, and the type o of booleans. Its set of constants is given in figure 6.1.The language PCF is interpreted in **Cpo** as specified in figure 6.2 (for the interpretation of Ω and Y, cf. definition 6.1.4). We use the same notation for the constants and for their interpretation, to simplify notation. This interpretation is called the *continuous model* of PCF. More generally, we define the following notion of standard model.

Definition 6.3.1 (standard) *Let* **C** *be a least fixpoint model. If we interpret* ι *and* o *by objects* D^ι *and* D^o *such that* $\mathbf{C}[1, D^\iota]$ *and* $\mathbf{C}[1, D^o]$ *are (order-isomorphic) to* $\omega_\perp$ *and* $\mathbf{B}_\perp$*, if the basic constants are interpreted as in figure 6.2, and if the*

$$\begin{array}{ll}
D^o = \mathbf{B}_\perp & \text{where } \mathbf{B}_\perp = \{\perp, tt, ff\} \\
D^\iota = \omega_\perp & \text{flat domain on natural numbers} \\
D^{\sigma\to\tau} = D^\sigma \to_{cont} D^\tau & \text{exponent in } \mathbf{Cpo}
\end{array}$$

$$succ(x) = \begin{cases} \perp & if\ x = \perp \\ x+1 & if\ x \neq \perp \end{cases} \qquad pred(x) = \begin{cases} \perp & \text{if } x = \perp \text{ or } x = 0 \\ x-1 & \text{otherwise} \end{cases}$$

$$zero?(x) = \begin{cases} \perp & \text{if } x = \perp \\ tt & \text{if } x = 0 \\ ff & \text{otherwise} \end{cases} \qquad if\ x\ then\ y\ else\ z = \begin{cases} \perp & \text{if } x = \perp \\ y & \text{if } x = tt \\ z & \text{if } x = ff \end{cases}$$

Figure 6.2: Interpretation of PCF in **Cpo**

first-order constants behave functionally as specified in figure 6.2 (replacing, say, $succ(x)$ by $ev \circ \langle succ, x\rangle$), then we say that we have a standard model of PCF.

Recall that if **C** has enough points, then the model is called extensional (cf. definition 4.5.4).

Definition 6.3.2 (order-extensional) *Let* **C**, *D^ι, and D^o be as in definition 6.3.1. Suppose moreover that* **C** *has enough points and that the order between the morphisms is the pointwise ordering, like in* **Cpo**. *Then the model is called order-extensional.*

Operational semantics of PCF. We equip PCF with an operational semantics which is adequately modelled by any standard model. It is described in figure 6.3 by means of a deterministic evaluation relation $\to_{op}$.

Exercise 6.3.3 *Let $add_l = Y(\lambda fxy.if\ zero?(x)\ then\ y\ else\ succ(f(pred(x))y))$. Compute $add_l\,4\,3$ using the rules in figure 6.2.*

Exercise 6.3.4 *Imitate the techniques of chapter 2 to establish that the rewriting system $\to$ specified by the eight axioms of figure 6.2 (applied in any context) is confluent, and that if $M \to^\star N$ and N is a normal form, then $M \to_{op}^\star N$. Hint: prove suitable versions of the standardisation and Church-Rosser theorems presented in chapter 2.*

Next we investigate the relationships between the denotational and the operational semantics of PCF.