The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming

C.-H. Luke Ong

Submitted for the degree of
Doctor of Philosophy

Imperial College of Science and Technology
University of London

May 31, 1988

Abstract

The commonly accepted basis for functional programming is the λ-calculus; and it is folklore that λ-calculus is the prototypical functional language in purified form. There is, nonetheless, a fundamental mismatch between theory and practice:

- Much of what is known about the model theory and proof theory of the λ-calculus is *sensible* in nature, *i.e.* all unsolvables are identified. Crucially, $\lambda x.\bot = \bot$ where $\bot$ represents any divergent term (or program).
- In practice, however, most implementations of functional languages are *lazy*, *i.e.* programs are reduced in *normal order* to *weak head normal forms* (whnf), corresponding to a *call-by-name* semantics. Consequently, $\lambda x.\bot \neq \bot$, because all abstractions, being in whnf, are deemed to be legitimate and meaningful programs.

This thesis seeks to develop a theory of *lazy* functional programming that corresponds to practice in the framework of the classical λ-calculus. The main topics studied in this thesis are as follows:

- The fundamental notions of solvability, λ-definability (of numeric functions), λ-theories and tree semantics in the classical sensible λ-calculus are reviewed and revised in the light of the lazy regime.
- Different formulations of *lazy λ-models* are presented and shown equivalent. We prove a *Local Structure Theorem* for the class of *free lazy PSE-models.*
- The full abstraction problem recast in the lazy λ-calculus, á la Abramsky, is studied. We focus on the *lazy λ-calculus with convergence testing* and study the problem of call-by-value simulation. A general method for constructing fully abstract models which are retracts of D — the initial solution of the domain equation $D \cong [D \to D]_\bot$ — with respect to a class of sufficiently expressive variants of Abramsky's $\lambda\ell$ is developed. The full abstraction problem of $\lambda\ell$ is reduced by this method to an open question of the *conservativity* of a labelled version of $\lambda\ell$ over itself.
- A proof system for the lazy λ-calculus (with convergence testing) based on Scott's logic of existence which is *correct* with respect to $\lambda\ell$, is introduced and given a *sound* and *complete* interpretation in partial categories. *Lazy reflexive objects* with enough points in *partial Cartesian closed dominical categories* give rise to lazy λ-models in which convergence testing is definable, thereby yielding a partial categories semantics.

Acknowledgements

I am immensely indebted to my two supervisors, Dr. Samson Abramsky and Professor John Darlington, for giving me freedom to pursue my own research interests and their careful efforts as academic mentors to help me cultivate confidence in my work. This thesis would scarcely have been written without Samson Abramsky who suggested the research topic to me in April 1986, and provided directions, even specific problems, at various crucial points in the course of my PhD programme. I would like to pay a special tribute to, what I term, his pedagogical perspicacity. For one thing, the past two years of comparatively smooth and thoroughly joyful course of research owes much to his judicious suggestion of Lazy Lambda Calculus as my research topic. His choice, and particularly, the timing — clearly in evidence of his acute discernment of my interests and masterly appreciation of what (little) I knew and was capable of learning as a somewhat perfervid and clumsy first year PhD student — set me off to a research area which he had evidently perceived to be fecund. For another thing, throughout our regular meetings over the past two and a half years, his insightful supervision and unflagging enthusiasm in my work were the most valuable help to me. His approach to Computer Science — always with rigour, much elegance, refreshingly cultured and philosophically informed, and finesse as a lecturer and teacher shall remain as a lasting influence on me.

I owe a special debt to Eugenio Moggi whom I first met at the British Theoretical Computer Science Colloquium at Leeds in March 1986. It was his talk at the Colloquium that fired my enthusiasm in Lambda Calculus. Since then, he has remained the most helpful of collaborators and generous of friends. Those who are familiar with his work on Lambda Calculus and Partial Categories will surely notice the significance of its influence on the work reported in Chapter 5 of this thesis.

I am very grateful to my colleagues from both the Functional Programming Research Group and the Computing Theory and Systems Technology Group in the Department of Computing for making scarce resources available for my use and also for providing a most lively and conducive environment for research. A special word of thanks goes to Axel Poigné (who left Imperial College in late 1987 to join GMD, West Germany) for his interest in my work and constant encouragement; and to Tom Maibaum for making special provisions to accommodate me in room 206A (for nearly a year) whose relative spaciousness and proximity to the Sun workstations I very much appreciate. I thank also Colin Atkinson, Mark Dawson, Yves Lafont and Paul Taylor who share the office with me for making it an enjoyable place in which to work. In particular, Mark's expertise in the nitty-gritties of systems programming and Paul's convenient package of

TEX macros for generating category-theoretic diagrams have made the task of document preparation using LATEX so much less painful.

Various colleagues outside of London have contributed much to my research programme. For a start, I ought to thank Alan Mycroft, Arthur Norman and Glynn Winskel from the Computer Laboratory in Cambridge for initiating me into Computer Science. I must especially acknowledge the advice given by Glynn Winskel in August 1985 (which I subsequently took up) regarding the institution in which to do my PhD and possible supervisors. I thank him for his advice, encouragement and interest in my work even after I have left Cambridge. Discussions with Henk Barendregt, Thierry Coquand, Carl Gunter, Furio Honsell, Giuseppe Longo, Mogens Nielson, Gordon Plotkin, Pino Rosolini and Allen Stoughton have helped me develop my ideas and improved the presentation of my work. In particular, it is with great pleasure and appreciation that I acknowledge the influence and inspiration of the work of Gordon Plotkin, Giuseppe Longo and Henk Barendregt. The problems they posed and techniques pioneered are central to my thesis.

This thesis would definitely not have been written without the financial aids arranged by Prof. Darlington: first, in the form of an Imperial College Bursary, and then research assistantship on the Alvey Project Flagship. I am deeply indebted to his commitment to academic freedom; his tolerance of and kindness to this independently spirited student has often been a striking surprise to me. It is my sincere hope that the fruit of my research will contribute somewhat to the theory of functional programming.

I am grateful to my parents and my friends at the SWDCG for their support and love which means a great deal to me.

Last, but not least, I thank Soo May for enriching the last year of my research beyond measure. I appreciate especially her sympathetic understanding throughout the time of my thesis preparation and patience in bearing the frustrating "nebulousness" of my companionship — for when I was writing I was not really with her even if I was; and when I was not writing, I was helplessly stricken with gloom for not doing what I had set out to do.

Contents

Chapter 1

Introduction

1.1 Preamble

Functional languages have had a small group of predominantly British advocates and cognoscenti for many years, until, it would seem, the fateful Turing Award speech delivered by J. W. Backus in 1978 [Bac78]. Since then, many more functional programming languages have sprung up, notably: LML [Aug84], MIRANDA [Tur85], PONDER [Fai85], LISPKIT [Hen80], TALE [BvL86] and ORWELL [Wad85]. A growing interest in the theory and practice of functional programming and the allocation of sizable resources to research in the area over the past few years are clear gestures of recognition and appreciation by the computing community at large. Functional languages are now beginning to attract a much wider interest; and several developments including the advent of highly parallel VLSI architectures are promising to translate the theoretical and programming advantages into practical reality. See *e.g.* [DR81].

Functional languages trace their origins to the lambda calculus developed by Church in the 1930's [Chu36] and recursion equations notation developed by Kleene, also in the 1930's [Kle36]. Lambda Calculus arose from research work in the theory of computability and in particular, in an attempt by Church to pin down mathematically the notion of numerical functions which are computable in a mechanical or algorithmic fashion. Church's proposal, or Church's thesis, as it became known later, was to identify the intuitively apprehensible class of effectively computable numerical functions with those that are definable in the lambda calculus given a suitable coding of the natural numbers in the calculus. There were other proposals to capture this class of computable functions, notably: μ-recursive and partial recursive functions by Gödel and Kleene in 1936; Turing Machine computable functions by Alan Turing [Tur36] and Universal Register Machine computable functions by Shepherdson and Sturgis [SS63]. Although there is great diversity among these various approaches and each has its own

rationale for being considered a plausible characterization of computability, the remarkable result is that the various approaches give rise to the same class of numerical functions. Because of this result, and that of their own experience, most mathematicians are led to accept Church's thesis. The fact that lambda calculus possesses such delightful computability results lends itself admirably to a mathematical foundation for functional programming.

Lambda calculus may be regarded as a quintessential paradigm for programming languages in general. This is not to say that lambda calculus by itself qualifies as a programming language and that programs can be written in it. What is meant is that one finds incarnations of many programming problems in the lambda calculus, and in particularly pure and generic forms. The syntax of the calculus is simple and classical: variables, application and abstraction in the pure calculus with applied calculi obtained by adding constants. Because of the similarities between the lambda calculus and programming languages, theoretical work on the semantics of the former could be applied to the latter. Theoretical work on the semantics of lambda calculus is particularly relevant to the denotational semantics of programming languages as well as to the design of languages, *e.g.* the language GEDANKEN [Rey70] is explicitly founded on the lambda calculus; see also [Gor73] and [Plo77] on how lambda calculus impinges on programming languages. Functional programming aficionados believe that the functional approach is a significant advance on conventional programming languages not only because of the benefits afforded them by the underlying mathematically clean and relatively well-understood basis and the rich body of mathematical knowledge closely related to it. The benefits also include the amenability of functional programs to formal derivation and manipulation and the comparative ease with which such programs lend themselves to parallel evaluation and systematic and methodical program development, owing to the property of referential transparency and the absence of side-effects. See *e.g.* [Dar84] for a comprehensive introduction to functional programming.

1.1.1 Lazy Evaluation

Lazy evaluation was introduced independently by [HM76] and [FW76]. Motivated more by the enhanced expressive power owing to lazy evaluation than by any pragmatic expediencies, lazy evaluation has since been regarded by many practitioners of functional programming as an indispensable feature of functional languages (see *e.g.* [Tur81]).

Consider a simple functional language consisting of λ-terms augmented with data constructors and first order primitive functions. Given a program

$$(\cdots(MN_1)\cdots N_n)$$

to be computed and an evaluator in the form of a reduction machine say, there are two crucial decisions to be made regarding the reduction process which will have significant semantic and pragmatic consequences, namely the *strategy* and the *extent* of the reduction scheme.

The first concerns the order in which redexes are chosen to be contracted at each stage. The issues at stake here include both *safety* as well as *efficiency*, and there is often a trade-off between the two. An applicative order reduction which contracts N_n first, then N_{n-1} and so on and finally M is efficient in the sense that the arguments needed by M are fully reduced when passed on to M, hence they are never reduced more than once. Unfortunately, such a reduction strategy may fail to terminate on a normalizable term which ought to be considered meaningful, *e.g.* $(\lambda x.y)\Omega$ where $\Omega \stackrel{\text{def}}{=} (\lambda x.xx)(\lambda x.xx)$. In this sense, it is unsafe. On the contrary, the leftmost[1] reduction strategy is safe, *i.e.* normalising. However, an argument may be needlessly reduced more than once, *e.g.*

$$(\lambda x.xyx)(\mathbf{I}z) \rightarrow ((\mathbf{I}z)y)(\mathbf{I}z) \rightarrow (zy)(\mathbf{I}z) \rightarrow (zy)z;$$

where **I** is the identity. The inefficiency of certain normalising reduction strategies, particularly the leftmost strategy, has been extenuated by the introduction of the graph model of reduction in which shared representation of terms is allowed. See [Tur79] and [Tur81] or [PJ87] for further details.

The *extent* of the reduction process answers the question: at what stage should the reduction of a program halt? It is concerned with the identification of a class of terms usually referred to as the "canonical normal forms" such that whenever the reduction of a program reaches a canonical normal form, the reduction process should be deemed to have yielded a meaningful answer; hence, the reduction does not need to proceed any further. Consider a program with circular structure, say

```
infl := (CONS 1 infl)
```

which is a plausible representation of a potentially "infinite" data object, namely, an infinite list of 1. A reduction scheme that halts only when all subprograms of the program to be reduced have been reduced to data constructors will preclude representations of pseudo infinite objects like `infl`. On the contrary, a leftmost strategy that halts whenever the program reaches an abstraction or the syntactic shape where the outermost subterm is a constructor or a primitive function (known collectively as *weak head normal forms*) will include infinite objects like `infl` among its class of canonical normal forms. This reduction strategy is called *lazy reduction.*

[1]Leftmost reduction is also known as leftmost-outermost or normal order reduction

1.1.2 The Lambda Calculus: An Introduction and Some Notations

In this subsection, we will set out the notational convention for the thesis. Most of the notations from elementary set theory and logic which we will use is standard and should cause no problem to the reader.

(i) **Sets and Functions:**
We use "$\stackrel{\text{def}}{=}$" for *definitional equality*, typically employed in laying down the definition of an entity, as in, say

$$\Lambda^o \stackrel{\text{def}}{=} \text{all closed } \lambda\text{-terms};$$

or in the case of $M \stackrel{\text{def}}{=} N$ meaning that "M is by definition equal to N".

Given a set X, we write $\wp X$ for the powerset of X and $\wp_{\mathsf{f}} X$ for the finite subsets of X. We write $X \subseteq_{\mathsf{f}} Y$ to mean "X is a finite subset of Y"; and $X \subset Y$ to mean "X is a proper subset of Y". We denote the function that maps e to $f(e)$ as $(e \mapsto f(e))$.

For any sets X and Y, write $[X \rightharpoonup Y]$ as the set of all *partial functions* form X to Y. We denote:

$$\phi(\sigma)\downarrow \stackrel{\text{def}}{=} \sigma \in \mathsf{dom}(\phi),$$

$$\phi(\sigma)\uparrow \stackrel{\text{def}}{=} \neg[\phi(\sigma)\downarrow].$$

(ii) **Numbers and Sequences:**
We use ω to denote the set of non-negative integers $\{0, 1, 2, \cdots\}$; and denote the set of *natural numbers*, *i.e.* $\{1, 2, 3, \cdots\}$, by $\mathbf{N}$. Denote the set of all finite sequences of ω by Seq, *i.e.*

$$\mathsf{Seq} \stackrel{\text{def}}{=} \{\langle n_1, \ldots, n_k\rangle : k, n_i \in \omega\}.$$

Let $\alpha = \langle n_1, \ldots, n_p\rangle, \beta = \langle m_1, \ldots, m_q\rangle \in \mathsf{Seq}$. The *length* of α, denoted $\mathsf{lh}(\alpha)$ is p. By convention, $\mathsf{lh}\langle\rangle = 0$. Also,

$$\begin{aligned}
\alpha * \beta &\stackrel{\text{def}}{=} \langle n_1, \cdots, n_p, m_1, \cdots, m_q\rangle \\
\alpha \leqslant \beta &\stackrel{\text{def}}{=} p \leqslant q \;\&\; \forall 1 \leqslant i \leqslant p.n_i = m_i, \\
\alpha < \beta &\stackrel{\text{def}}{=} \alpha \leqslant \beta \;\&\; \alpha \neq \beta.
\end{aligned}$$

(iii) **Naked and Labelled Trees:**
A *tree* is a set $A \subseteq \mathsf{Seq}$ such that

(1) $\forall \beta \leqslant \alpha \in A \Rightarrow \beta \in A$.

(2) $\alpha * \langle n+1 \rangle \in A \Rightarrow \alpha * \langle n \rangle \in A$.

The *subtree* of A at node α, denoted A_α, is the tree

$$\{\, \beta : \alpha * \beta \in A \,\}.$$

Let Σ be a set of symbols. A Σ-*labelled tree* is a tree with an element of Σ written at each node. More formally, it is a partial map $\phi : \mathsf{Seq} \rightharpoonup \Sigma$ such that

$$T_\phi \stackrel{\text{def}}{=} \{\, \alpha \in \mathsf{Seq} : \phi(\alpha)\downarrow \,\} \text{ is a tree,}$$

called the *underlying tree.* The *label* at node $\alpha \in T_\phi$ is $\phi(\alpha)$.

The Formal $\lambda\beta$-Theory

We introduce the formal theory $\lambda\beta$. Some by now standard results in lambda calculus needed in the sequel will be stated and the reader will be referred to the classic treatise on lambda calculus [Bar84] for their respective proofs and further ramifications. Notations and terminology introduced in this work will conform to those in [Bar84] as far as possible, unless otherwise stated. We introduce lambda calculus in the Hilbert style as follows:

DEFINITION **1.1.2.1** *The Formal Theory* $\lambda\beta$

- λ-*terms*: $M ::= x | (\lambda x.M) | (MN)$ where x ranges over Var $\stackrel{\text{def}}{=}$ a denumerable set of variables.
- *Formulae*: equations $M = N$ for all λ-terms M, N.
- *Axioms*: We introduce three axiom schemata where M, N, P range over λ-terms and x, y over variables.

(α) $\qquad \lambda x.M = \lambda y.M[x := y] \qquad (y \text{ not free in } M)$

(β) $\qquad (\lambda x.M)N = M[x := N]$

- *Rules*:
  "=" is an equivalence relation

$$(\mu) \qquad \frac{M = N}{PM = PN}$$

$$(\nu) \qquad \frac{M = N}{MP = NP}$$

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

The Formal Theory $\lambda\beta\eta$
$\lambda\beta\eta$ is the theory obtained by adding the following axiom scheme to $\lambda\beta$.

$$(\eta) \qquad \lambda x.Mx = M \qquad (x \text{ not free in } M)$$

NOTATION **1.1.2.2** (i) The set of pure, untyped λ-terms is denoted Λ. Let C be a set of constant symbols. We define $\Lambda(C)$, the collection of λC-terms, as follows:

$$M \in \Lambda(C) ::= c \mid x \mid (MN) \mid (\lambda x.M),$$

where c ranges over C. The symbol "$\equiv$" denotes syntactic equality. Outermost parentheses are omitted.
We will usually use upper-case letters M, N, P, Q, R, L with or without their respective subscripts as meta-variables for λ-terms and lower-case letters x, y, z, a, b, c with or without their respective subscripts for variables. $\vec{M}$ is a shorthand for $\langle M_1, M_2, \cdots, M_n\rangle$ where $|\vec{M}| \stackrel{\text{def}}{=} n \geqslant 0$, *i.e.* a possibly empty finite sequence of λ-terms (since M ranges over λ-term). Thus, similarly, $\vec{x}$ stands for a possibly empty finite sequence of variables. A ***substitution*** is a map σ : Var $\rightarrow \Lambda^o$ and for $M \in \Lambda$, we write M_σ to mean the λ-term obtained from M by substituting simultaneously $\sigma(x)$ for x, for all $x \in$ Var.

(ii) Application associates to the left and λ-abstractions to the right. Hence,

$$MN_1 \cdots N_n \equiv (\cdots((MN_1)N_2)\cdots N_n) \equiv M\vec{N}$$

and

$$\lambda x_1 \cdots x_n.M \equiv (\lambda x_1.(\lambda x_2.\cdots(\lambda x_n.M)\cdots)) \equiv \lambda\vec{x}.M.$$

Note that λ-terms can be divided into three mutually exclusive syntactic classes: namely applications, abstractions and variables.

(iii) We shall assume the standard notions of free and bound variables as explained *e.g.* in [Bar84]. A λ-term (or λC-term) is *closed* if it has no free variable; otherwise it is *open*. Denote the set of free variables of a λ-term M as $\mathsf{FV}(M)$. We write $\Lambda(C)^o$ for the collection of all closed λC-terms. To all intents and purposes, all expressions will be considered modulo α-convertibility. The (simultaneous) substitution of $\vec{M}$ for $\vec{x}$ respectively in N (which is, of course, meaningful provided $|\vec{x}| = |\vec{M}|$) is denoted $N[\vec{x} := \vec{M}]$; variable capture is avoided by suitable renaming of bound variables. In *op. cit.* the symbol $\sqsubseteq$ is used exclusively for the so-called "Böhm tree preorder". We prefer to use $\sqsubseteq$ as a generic notation for *operational preorders* and use appropriate subscript or superscript to specify the particular preorder we have in mind. For example, Böhm tree preorder, Longo tree preorder and (Abramsky) bisimulation preorder (to be introduced in Chapter 4) are denoted as $\sqsubseteq_B$, $\sqsubseteq_L$ and $\sqsubseteq^B$ respectively.

(iv) Provability in the theories $\lambda\beta$ and $\lambda\beta\eta$ will be denoted by $\lambda\beta \vdash$ and $\lambda\beta\eta \vdash$ respectively. We shall say that M and N are *β-convertible* or simply *convertible* if $\lambda\beta \vdash M = N$; and denote it as $M =_\beta N$ or simply $M = N$ when no confusion is likely to occur. Similarly, for $\beta\eta$-convertibility.

(v) We assume the notion of a context. A closed context is one without any free variable, which we abbreviate as $C[\,] \in \Lambda^o$.

(vi) We shall frequently use the following combinators:

$$\mathbf{I} \equiv \lambda x.x, \qquad \mathbf{K} \equiv \mathbf{T} \equiv \lambda xy.x,$$
$$\mathbf{F} \equiv \lambda xy.y, \qquad \mathbf{S} \equiv \lambda xyz.xz(yz),$$
$$\Delta_n \equiv (\lambda x.\underbrace{x\cdots x}_{n}), \qquad \mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx),$$
$$\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)).$$

1.2 Mismatch between Theory and Practice

1.2.1 Solvability: Meaningfulness in the Strict Regime

If lambda calculus is to be regarded as a prototypical functional programming language, then the following are pertinent questions to ask.

1. What are the meanings of λ-terms (seen as programs to be computed)?
2. What are the meaningless λ-terms? Can they be characterized?

These questions were obviously of interest not just to the computer scientists whose interests in lambda calculus may be subservient to and motivated by semantic and pragmatic concerns in the realms of real-life programming languages. Logicians and mathematicians alike have thought about the above questions.

Meaning of λ-terms: a first attempt

- The meaning of a λ-term is its β-normal form (nf), if it exists.
- The meaningless λ-terms are precisely those without normal forms and are therefore to be identified.

This view incorporates such a natural and simple interpretation of λ-terms that if it worked there would surely be no doubt that it was the right one. However, the λ-theory which is obtained by equating all λ-terms with no nf is inconsistent, *i.e.* every (closed) equation is provable in this λ-theory. [2]

A Second Attempt

- The meaning of a λ-term is its *head normal form* (hnf), if it exists; or if a more sophisticated discrimination is needed, its *Böhm tree.*
- All terms *without hnfs* (the *unsolvable* terms) are identified.

The proposal was first made by Henk Barendregt in his thesis [Bar71] which later formed the central thesis of his book, giving rise to a beautiful and successful theory (henceforth referred to as the *sensible* theory, *i.e.* one which identifies all unsolvable terms), as the book shows.

Basic Notions of the Sensible Theory

We will give a brief survey of the basic notions of the sensible theory.

DEFINITION **1.2.1.1** A closed λ-term is *solvable* if $\exists \vec{N} \subseteq \Lambda . M\vec{N} = \mathbf{I}$. The definition is extended to arbitrary λ-terms by considering their closures, *i.e.* M is solvable if $\lambda\vec{x}.M$ is solvable where $\mathsf{FV}(M) \equiv \{\vec{x}\}$.

That unsolvable terms can and should be regarded as computationally irrelevant and carrying no substantive information can be seen from the following result:

[2] We say that two λ-terms M and N are *incompatible* if any λ-theory that equates M and N is inconsistent. Let $M \equiv \lambda x.x\mathbf{K}\Omega$, $N \equiv \lambda x.x\mathbf{S}\Omega$. Since both are without nf, in any λ-theory that equates such terms, we have $\mathbf{K} = M\mathbf{K} = N\mathbf{K} = \mathbf{S}$, equating $\mathbf{K}$ and $\mathbf{S}$. It is a fact that $\mathbf{K}$ and $\mathbf{S}$ are incompatible. See [Bar84] for further details.

LEMMA **1.2.1.2** (**Genericity**) *Let $M, N \in \Lambda$ such that M is unsolvable and N has a nf. Then, $\forall C[\,] \in \Lambda$,*

$$C[M] = N \Rightarrow \forall L \in \Lambda . C[L] = N.$$

PROOF The proof uses the continuity of the context operation. See the proof of Proposition 14.3.24 in [Bar84, page 374]. □

We may read the above result as follows: A context $C[\,]$ may be regarded as an algorithm that accepts as input that which is instantiated in the "hole". If the output of a computation with meaningless input is meaningful, then the algorithm may be regarded as a constant function, *i.e.* it does not *need* the input for any computation.

EXAMPLE **1.2.1.3** $\Omega, \lambda x.\Omega$ and $\mathbf{YK}$ are unsolvable. In fact, we can say more:

$$\forall X \in \Lambda . X \text{ unsolvable} \Rightarrow \lambda x.X \text{ unsolvable}.$$

Hence, if all unsolvables have the same denotation $\bot$, say, then we have

$$\lambda x.\bot = \bot.$$

Wadsworth [Wad71] proved a significant Characterization Theorem for the solvable terms.

Seen from a computational point of view, his result says that there is a precise correspondence between solvability and a termination property in a certain reduction strategy. Note that the latter is a purely operational notion. We formalize some syntactic notions before stating Wadsworth's result.

FACT **1.2.1.4** *Each λ-term has exactly one of the following forms:*

(1) $\lambda \vec{x}.y\vec{M}$,

(2) $\lambda \vec{x}.(\lambda x.M_0)M_1 \cdots M_m, m \geqslant 1$. □

λ-terms of the shape (1) are in *head normal form* (hnf).

A corollary of the preceding observation is the following:

FACT **1.2.1.5** *The following is a mutually exclusive classification of λ-terms:*

(1) abstraction	$\lambda x.M$,
(2) λ-free hnf	$x\vec{M}$,
(3) leftmost β-reducible form ($l\beta rf$)	$(\lambda x.P)Q\vec{M}$.

□

We call (1) and (2) above collectively as *weak head normal forms* (whnf) and denote the collection of all (closed) whnf's as Λ_{whnf} ($\Lambda_{\mathsf{whnf}}{}^{o}$).

DEFINITION **1.2.1.6** (i) The redex $(\lambda x.P)Q$ in Fact 1.2.1.4(2) is called the *head redex.*

(ii) The *leftmost redex* of a λ-term is the redex whose "λ" occurs leftmost in the term. The leftmost redex is sometimes referred to as the *leftmost-outermost* or *normal order* redex.

(iii) A λ-term M has a *lazy redex* if M is a $l\beta$rf and the lazy redex is then precisely the leftmost redex.

EXAMPLE **1.2.1.7** In $(\lambda x.x((\lambda y.P)Q)), (\lambda y.P)Q$ is the leftmost redex but not the lazy or head redex. Generally:

- head redex $\overset{\Rightarrow}{\not\Leftarrow}$ leftmost redex.
- lazy redex $\overset{\Rightarrow}{\not\Leftarrow}$ head redex.

In what follows we define two reduction strategies which will turn out to be crucial in capturing the class of well-behaved terms in two computationally different regimes, the contrasts between which we will emphasize by referring to one as "strict" and the other as "lazy". We will spell out the precise difference in the following section.

DEFINITION **1.2.1.8** (i) Let $P_0 \equiv \lambda\vec{x}.(\lambda x.M_0)M_1\vec{N}$. The *head reduction path* of P_0 is the uniquely determined sequence

$$P_0 \to_{\mathrm{h}} P_1 \to_{\mathrm{h}} P_2 \cdots \to_{\mathrm{h}} P_n \to_{\mathrm{h}} \cdots \qquad (\dagger)$$

where $\to_{\mathrm{h}}$ is the 1-step head reduction.
If $(\dagger)$ terminates, say at P_n, then P_0 is said to *have a hnf* and P_n is the *principal hnf (phnf).* If $(\dagger)$ diverges, then P_0 has no hnf.

(ii) The *lazy reduction path* of a $l\beta$rf $P_0 \equiv (\lambda x.M_0)M_1\vec{N}$ is the uniquely determined sequence

$$P_0 \to_{\mathrm{l}} P_1 \to_{\mathrm{l}} P_2 \cdots \to_{\mathrm{l}} P_n \to_{\mathrm{l}} \cdots \qquad (*)$$

where $\to_{\mathrm{l}}$ is the 1-step lazy reduction.
If $(*)$ terminates, say at P_n, then P_n is said to *have a whnf*, and P_n is the *principal whnf.* Note that in this case, $\forall i < n$, P_i is in $l\beta$rf. If $(*)$ diverges, then P_0 has no whnf.

We state Wadsworth's result and refer the reader to either [Wad71] or [Bar84, page 300] for a proof.

FACT **1.2.1.9 (Characterization of Solvability)** *A λ-term is solvable iff it has a hnf.* □

Sensibleness = "Strictness"

Observe that the chief difference between head reduction and lazy reduction is that the former may "compute under a λ", and halts at hnfs; whereas the latter may not compute under a λ, and halts at whnfs. At each step, both schemes contract the leftmost redex, but the latter has a shorter *extent.* **It is precisely in the sense of unrestricted computation under a λ that we dub the head reduction, or more generally, the sensible theory, strict.** This is not inconsistent with the usual domain-theoretic understanding of strictness. For we claim that head reduction or the sensible approach gives rise to an interpretation of λ-terms which entails a "strict" λ-abstraction operator. According to the sensible approach, the undefined terms are just the unsolvables. Now, observe that if M is unsolvable, so is $\lambda x.M$. Semantically, this corresponds to a strict "graph" function, in λ-model theoretic jargon. See Chapter 3 for further discussions.

We may rephrase Wadsworth's result as follows:

> **Solvable terms or equivalently, terms which have hnfs, are precisely the meaningful terms in the "strict" regime.**

For more than a decade now, the consensus has been to regard solvable terms as the class of meaningful terms. Referring to model-theoretic investigations of lambda calculus, Coppo *et al.* [CDV81] maintain that they

> agree on the fact that terms with a hnf are all and only those to which a "meaning" different from "undefined" can be assigned.

See also [Lei86] for a more complete computational and semantic characterization of the functional properties of λ-terms, which arrives at essentially the same conclusion.

1.2.2 Meaningfulness in the Lazy Regime

The sensible theory, then, has been the commonly accepted foundation for functional programming languages; more precisely, for the *lazy* functional languages, which represents the mainstream of current functional programming practice. But do current functional languages as defined and implemented actually evaluate to head normal form? To the best of my knowledge, most do not — they do not evaluate under λ-abstractions. They evaluate to *weak head normal forms.*

We therefore have a stark mismatch between theory and practice. We illustrate this with a simple example. Consider an unsolvable term, say, Ω. Note that according to the sensible theory, or equivalently, in the "strict"

regime where there is no restriction on reduction under an λ-abstraction, we have $\lambda x.\Omega = \Omega$, since $\lambda x.\Omega$ is also unsolvable. However, in the lazy regime, $\lambda x.\Omega$ being in weak head normal form is a computationally meaningful value and hence is to be distinguished from Ω.

Since current practice, especially lazy evaluation, is well-motivated by sound programming methodological principles and the underlying implementation is driven by efficiency considerations, it makes sense to see if a good theory can be developed for it.

1.3 Overview of the Thesis

This thesis is an investigation into the foundations of functional programming in the framework of the pure untyped λ-calculus. Although λ-calculus has long been accepted as a basis for functional programming, and it is forklore that λ-calculus is the prototypical functional language in purified form; there is, nonetheless, a fundamental mismatch between theory and practice.

- Much of what is known about the model theory and proof theory of the λ-calculus is *sensible* in nature, *i.e.* all unsolvables are identified. Crucially, $\lambda x.\bot = \bot$ where $\bot$ represents any divergent term (or program).
- In practice, however, most implementations of functional languages are *lazy*, *i.e.* programs are reduced in *normal order* to *weak head normal forms* (whnf), corresponding to a *call-by-name* semantics. Consequently, $\lambda x.\bot \neq \bot$, because all abstractions, being in whnf, are deemed to be legitimate and meaningful programs.

This thesis seeks to develop a theory of *lazy* functional programming that corresponds to practice in the framework of the classical λ-calculus. We will regard (closed) terms of the calculus as *programs* and abstractions (= weak head normal forms) as *values*. The lazy evaluation mechanism is represented as a deterministic reduction relation on λ-terms which reduces programs in normal order to weak head normal forms.

The rest of the thesis is organized as follows:

Chapter 2 formulates *lazy evaluation* in the pure untyped λ-calculus. The fundamental notions of solvability, λ-definability (of numeric functions), tree semantics and λ-theories in the classical sensible λ-calculus are reviewed and revised in the light of the lazy regime. In the lazy regime, only a proper subclass of the unsolvables constitute the undefined programs — the *strongly unsolvables*,

which is characterized operationally as the divergent terms under the lazy reduction.

Weak Böhm trees, Böhm trees and *Longo trees* are studied and their inter-relationships, including a non full abstraction result, investigated. We show that Longo tree preorder is a *precongruence* (or it is *substitutive*). A theory of *lazy λ-theories* is developed.

Chapter 3 is a study of the *model theory* of the lazy λ-calculus. It may be divided into two parts. The first part begins with an introduction to the model theory of the classical λ-calculus. Different formulations of the notion of *lazy λ-model* based on Abramsky's *quasi-applicative structure with divergence* are then presented and shown equivalent. A general *computational adequacy* result for a class of continuous lazy λ-models is proved. The second part of this Chapter focuses on a class of λ-models called the *Plotkin-Scott-Engeler (PSE) models.* The basic properties of the subclass of free lazy PSE-models are surveyed. The main result of this Chapter is a *Local Structure Theorem* for the class of *free lazy PSE-models.*

The precursor of the work reported in **Chapter 4** is Abramsky's application of the *Stone duality* between domains and their logics of observable properties [Abr87,Abr88] to tackle the problem of full abstraction [Plo77,Mil77] reformulated in the lazy λ-calculus. The study is based on a paradigmatic functional programming language $\lambda\ell$ called the pure lazy language whose terms are the closed λ-terms. The lazy evaluation mechanism is governed by a reduction relation. A program in this language *converges* if it reduces to an abstraction; otherwise, it *diverges.* An operational preorder $\sqsubseteq^B$ called *applicative bisimulation* is then defined which compares two terms precisely according to the criterion of whether they are distinguishable by observing their respective applicative behaviour. Taking the cue from $\lambda\ell$, a general class of structures called *lambda transition systems* (lts) is defined which includes *both* the *language* $\lambda\ell$ and the canonical denotational *model* D — the initial solution of the domain equation $D \cong [D \to D]_\perp$. Reformulated in the lazy regime, the full abstraction problem is the following: Is it true that

$$\forall M, N \in \Lambda^o . M \sqsubseteq^B N \iff D \models M \sqsubseteq N?$$

The answer is no and a counter-example is provided.

Full abstraction fails because the language $\lambda\ell$ is not expressive enough — more precisely, not all finite information in the model D may be internally represented in the language. Abramsky shows that full abstraction for D may be achieved by enriching the language $\lambda\ell$ with a parallel convergence constant. We continue the study of full abstraction by considering *retracts* of D as candidates for fully abstract models with respect to various enriched variants of the language $\lambda\ell$.

PhD Thesis May 31, 1988

One such variant is $\lambda\ell_C$, lazy λ-calculus with *convergence testing.* Convergence testing (which is not definable in $\lambda\ell$) corresponds to a discriminatory function between convergent and divergent elements. Its introduction enables the projection functions $\psi_n : D \to D_n$ (the n-th approximant of D) to be internally definable in the language. We study the formal system $\lambda\beta C$, which is obtained by extending $\lambda\beta$ with convergence testing C and show that it is *Church-Rosser* and prove a *Standardization Theorem* for the associated reduction. Plotkin's problem of *simulating* call-by-value evaluation in the call-by-name regime is revisited. We show that there is a translation of terms from $\langle \Lambda^o, \Downarrow_v \rangle$ (Plotkin's call-by-value language) to $\lambda\ell_C$ which preserves call-by-value convergence exactly.

$\lambda\ell_C$ is not fully abstract with respect to D. A main result of this Chapter is a general method which uses *bisimulation logical relations* to construct retracts of D which are fully abstract models for a class of suitably expressive languages (lts's) which includes $\lambda\ell_C$. The problem of constructing a retract of D which is fully abstract for $\lambda\ell$ is reduced by this method to an open question of *conservativity* of $\lambda\ell_\omega$ (a labelled version of $\lambda\ell$) over $\lambda\ell$. This Chapter ends with an investigation into the relationships between the various lazy operational preorders introduced in this thesis.

Chapter 5 divides naturally into three sections. Section 1 introduces the various category-theoretic approaches to partiality, surveying notions like Rosolini's *p-category*, diPaola and Hellers' *dominical category*, their inter-relationships and various concreteness criteria. These lead up to the notions of *partial Cartesian closed category* pCCC and *partial Cartesian closed dominical category* pCCDC which may be regarded as *partial* counterparts of Cartesian closed category. Two main applications of partial categories then follow. A formal proof system for proving equivalences between program phrases in the lazy regime using Scott's logic of existence called the λ_L-calculus is the subject matter of section 2. λ_L is shown to be *correct* with respect to $\lambda\ell$. λ_L augmented with *convergence testing* has a *sound* and *complete* interpretation in categories of partial morphisms. The second application of partial categories, contained in section 3, is a category-theoretic presentation of *lazy* λC*-models*, which are lazy λ-models in which *convergence testing* is definable. We prove that *lazy reflexive objects* in a pCCDC give rise to lazy λC-models.

Finally, in **Chapter 6**, we consider the limitations of this thesis and mention the corresponding possibilities for further research.

Chapter 2

Sensible Theory Revised: Tree Semantics and Lazy Lambda Theories

Synopsis of the Chapter

This Chapter formulates *lazy evaluation* in the pure untyped λ-calculus. The fundamental notions of solvability, λ-definability (of numeric functions), tree semantics and λ-theories in the classical *sensible* λ-calculus are reviewed and revised in the light of the lazy regime. In the lazy regime, only a proper subclass of the unsolvables constitute the undefined programs — the *strongly unsolvables*, which is characterized operationally as the divergent terms under the lazy reduction. With this new notion of undefinedness, the lazy λ-calculus is shown to retain *universal computing power*. *Weak Böhm trees*, Böhm trees and *Longo trees* are studied and their inter-relationships, including a *non full abstraction* result, investigated. We show that Longo tree preorder is a *precongruence* (or *substitutive* preorder). Finally, a theory of *lazy λ-theories* is developed.

2.1 Laziness and its Properties

2.1.1 Formulation of Laziness

We formalize lazy reduction in the context of pure type-free λ-calculus in two slightly different ways:

DEFINITION **2.1.1.1** (i) Define the following relation scheme:

$$(\lambda x.P)Q\vec{M} \to_1 P[x := Q]\vec{M}$$

as the *one-step lazy reduction.* We then define:

$$\begin{array}{lll} \twoheadrightarrow_l & \stackrel{\text{def}}{=} & \text{the reflexive, transitive closure of } \rightarrow_l, \\ M\uparrow_l & \stackrel{\text{def}}{=} & \exists\{ M_n : n \in \omega \} \,\&\, \forall n \in \omega . M_n \rightarrow_l M_{n+1}, \\ M\not\rightarrow_l & \stackrel{\text{def}}{=} & M \notin \mathsf{dom}(\rightarrow_l), \\ M \downarrow_l N & \stackrel{\text{def}}{=} & M \twoheadrightarrow_l N \,\&\, N\not\rightarrow_l. \end{array}$$

We read $M\uparrow_l$ as "M diverges lazily" and $M \downarrow_l N$ as "M converges lazily to N". When no confusion is likely, we will omit the subscript l.

(ii) Gordon Plotkin [Plo75] and Samson Abramsky [Abr87, Chapter 6] introduced the following binary reduction relation. $M \Downarrow N$ ("M converges to *principal weak head normal form* N") is defined inductively over Λ^o as follows:

(abs-$\Downarrow$)
$$\frac{}{\lambda x.M \Downarrow \lambda x.M}$$

(β_N)
$$\frac{M \Downarrow \lambda x.P \quad P[x := Q] \Downarrow N}{MQ \Downarrow N}$$

NOTATION:

$M\Downarrow \stackrel{\text{def}}{=} \exists N.M \Downarrow N$ ("M converges")

$M\Uparrow \stackrel{\text{def}}{=} \neg(M\Downarrow)$ ("M diverges")

It is clear that $\downarrow_l$ and $\Downarrow$ are partial functions from Λ to Λ_{whnf} and Λ^o to $\Lambda_{\mathsf{whnf}}{}^o$ respectively. In fact, restricted to closed λ-terms, $\downarrow_l$ and $\Downarrow$ describe the same partial function, which we will show.

The two convergence predicates "$M \Downarrow N$" and "$M \downarrow_l N$" may be defined with a measure of the number of steps or the amount of time required for convergence.

Define $\langle M \Downarrow N, t\rangle$ ("M converges to N at time t") inductively as follows:

(abs-$\Downarrow$)
$$\frac{}{\langle \lambda x.M \Downarrow \lambda x.M, 0\rangle}$$

(β_N)
$$\frac{\langle M \Downarrow \lambda x.P, t\rangle \quad \langle P[x := Q] \Downarrow N, t'\rangle}{\langle MQ \Downarrow N, t + t' + 1\rangle}$$

Similarly, we define $\langle M \downarrow_l N, t\rangle$("$M$ converges lazily to N at time t") as:

$$\langle M \downarrow_l N, t\rangle \stackrel{\text{def}}{=} M \equiv M_0 \to_l M_1 \to_l \cdots \to_l M_t \equiv N \in \Lambda_{\text{whnf}}.$$

It should be clear that $M \Downarrow N \iff \exists t.\langle M \Downarrow N, t\rangle$; further, the "time" t is unique for each M. Similarly, for $M \downarrow_l N$.

LEMMA **2.1.1.2** *Let $M \in \Lambda^o$. Then, $M \Downarrow N \iff M \downarrow_l N$.*

PROOF We remark that: $M \twoheadrightarrow_l N \Rightarrow \forall \vec{Q} \subseteq \Lambda^o.M\vec{Q} \twoheadrightarrow_l N\vec{Q}$. "$\Rightarrow$": We prove: $\langle M \Downarrow N, t\rangle \Rightarrow \langle M \downarrow_l N, t\rangle$ by induction on t. This is clearly valid for $t = 0$. Suppose true for some $t \geqslant 0$. Let $\langle MQ \Downarrow N, t+1\rangle$. Then, by definition, we have $\langle M \Downarrow \lambda x.P, t_1\rangle$ and $\langle P[x := Q] \Downarrow N, t_2\rangle$ for some Q, $t_1, t_2 \leqslant t$ and that $t = t_1 + t_2$. By induction hypothesis, we have $\langle M \downarrow_l \lambda x.P, t_1\rangle$ and $\langle P[x := Q] \downarrow_l N, t_2\rangle$ respectively. Finally, by the remark at the beginning of the proof,

$$MQ \twoheadrightarrow_l (\lambda x.P)Q \twoheadrightarrow_l N \in \Lambda_{\text{whnf}}{}^o.$$

Whence, $\langle MQ \downarrow_l N, t_1 + t_2 + 1\rangle$.
"$\Leftarrow$": We prove: $\langle M \downarrow_l N, t\rangle \Rightarrow \langle M \Downarrow N, t\rangle$. Clearly, the base case of $t = 0$ is trivial. Suppose true for some $t \geqslant 0$. Let $\langle M \downarrow_l N, t+1\rangle$, i.e.

$$M \equiv AB \equiv M_o \to_l M_1 \to_l \cdots \to_l M_t \to_l M_{t+1} \equiv N.$$

Now, for some $0 \leqslant t' < t+1$, $\langle A \downarrow_l \lambda x.P, t'\rangle$ and $\langle P[x := B] \downarrow_l N, t''\rangle$ with $t = t' + t''$. By induction hypothesis, $\langle A \Downarrow \lambda x.P, t'\rangle$ and $\langle P[x := B] \Downarrow N, t''\rangle$, and so, $\langle M \Downarrow N, t+1\rangle$. □

Having pinned down a notion of lazy reduction, we shall show that it is inappropriate to regard the whole of the unsolvable terms as undefined or meaningless in the lazy regime. In fact, only a proper subclass of the unsolvable terms, which we will call the *strongly unsolvable terms*, corresponds to meaninglessness and warrants the denotation $\bot$.

A term which converges lazily may be construed as having resulted in an increase in the relevant computational information. In the light of this computational perspective, terms devoid of any potentially relevant information are precisely those which fail to terminate under the lazy reduction, *i.e.* $\to_l$.

Functionality Order and Strong Unsolvability

We introduce the notion of the *functionality order* of a λ-term. Informally, the functionality order of a term M expresses how "higher order" M is, which corresponds roughly to the number of nested λ-abstractions M (or its β-convertible form) has. A term of functionality order 0, say, is not β-convertible to an abstraction; or equivalently, when applied to another term, does not "act upon it".

DEFINITION **2.1.1.3** (i) A term M has *functionality order* (or simply *order*) 0, denoted $M \in \mathbf{O}_0$ iff $\neg[\exists N.\lambda\beta \vdash M = \lambda x.N]$.

(ii) M has *proper order* 0, denoted $M \in \mathbf{PO}_0$ if

$$M \in \mathbf{O}_0 \,\&\, \neg[\exists \vec{N}.\lambda\beta \vdash M = x\vec{N}].$$

We say that M is *strongly unsolvable* if M is of proper order 0.

(iii) M has *order* n, denoted $M \in \mathbf{O}_n$, iff n is the largest i such that

$$\exists N.\lambda\beta \vdash M = \lambda x_1 \cdots x_i.N.$$

(iv) M has *order* ∞, denoted $M \in \mathbf{O}_\infty$ iff $\forall n \in \omega.M \notin \mathbf{O}_n$.

(v) M is a *functional term*, denoted $M \in \mathcal{F}$, if $\exists N.\lambda\beta \vdash M = \lambda x.N$; further,

$$M \in \mathcal{F}_n \stackrel{\text{def}}{=} \exists N.\lambda\beta \vdash M = \lambda x_1 \cdots x_n.N.$$

REMARK **2.1.1.4** It should not be difficult to see that $\{\mathbf{O}_i : i \in \omega + 1\}$ is a partition of $\mathbf{\Lambda}$. Also,

$$\begin{array}{ccccccc} \mathcal{F} & \equiv & \mathcal{F}_1 & \supset & \mathcal{F}_2 & \supset & \mathcal{F}_3 \quad \cdots \\ & & \cup & & \cup & & \cup \\ & & \mathbf{O}_1 & & \mathbf{O}_2 & & \mathbf{O}_3 \quad \cdots \end{array}$$

EXAMPLE **2.1.1.5** (i) $x\vec{M} \in \mathbf{O}_0$, $\Omega\vec{M} \in \mathbf{PO}_0$. $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ is syntactically the minimal strongly unsolvable term and will be important as a prime example of the class of $\mathbf{PO}_0$-terms in the sequel. In fact,

$$\forall m, n \geqslant 2.\forall \vec{Q} \subseteq \mathbf{\Lambda}.\Delta_m\Delta_n\vec{Q} \in \mathbf{PO}_0.$$

(ii) Let F be any fixed-point operator in $\mathbf{\Lambda}$, *i.e.* $\forall M \in \mathbf{\Lambda}.M(FM) = FM$; for example, the "paradoxical fixed-point combinator" $\mathbf{Y}$ or

$$\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy)).$$

Since $\lambda\beta \vdash F\mathbf{K} = \mathbf{K}(F\mathbf{K}) = \cdots = \underbrace{\mathbf{K}(\cdots(\mathbf{K}}_{n}(F\mathbf{K}))) = \lambda x_1 \cdots x_n.F\mathbf{K}$, we have $F\mathbf{K} \in \mathbf{O}_\infty$. Note that $F\mathbf{I} \in \mathbf{PO}_0$.

We prove a classification result for the unsolvable terms which shows that the strongly unsolvable terms are a proper subclass of the unsolvables.

PROPOSITION **2.1.1.6 (Classification of the Unsolvables)** *M is unsolvable iff*

(i) $M \in \mathbf{O}_\infty$ *or*

(ii) $\exists n \geqslant 0. \exists N \in \mathbf{PO}_0. \lambda\beta \vdash \lambda x_1 \cdots x_n.N = M$

Then, n is called the order of unsolvability *of M. In the case of (i), we say that M has order of unsolvability* ∞.

PROOF "$\Leftarrow$": By Wadsworth's Characterization of Unsolvability.
"$\Rightarrow$": Suppose $M \notin \mathbf{O}_\infty$. Assume n is the largest such that $M = \lambda x_1 \cdots x_n.N$. Then, $\lambda\beta \not\vdash N = x\vec{P}$ since M is unsolvable. By maximality of n, $N \in \mathbf{O}_0$. Hence, $N \in \mathbf{PO}_0$. □

$\mathbf{PO_n}$: Unsolvables of Order n

By abuse of language and for consistency with the case $n = 0$, we will call an unsolvable term of order n, M, a $\mathbf{PO_n}$-term, denoted $M \in \mathbf{PO_n}$; also we decree that $\mathbf{PO}_\infty \stackrel{\text{def}}{=} \mathbf{O}_\infty$. Hence, we have

$$\{\text{ unsolvable terms }\} = \bigcup_{n \in \omega+1} \mathbf{PO_n}.$$

Note that $\forall n \in \omega. \mathbf{PO_n} \subset \mathbf{O}_n$ and $\mathbf{\Lambda} = \bigcup_{n \in \omega+1} \mathbf{O}_n$. In view of the above results, we will call a term that is not strongly unsolvable, in other words, a functional term, a *weakly solvable term.*

In the next section, we will return to the $\mathbf{PO_n}$-terms and study a class of λ-theories which we shall call *lazy λ-theories.*

2.1.2 Operational Characterization of Strong Unsolvability

Next, we establish an *operational characterization* of strong unsolvability. First, a definition.

DEFINITION **2.1.2.1** An infinite reduction starting from M

$$M \equiv M_0 \stackrel{\Delta_0}{\to} M_1 \stackrel{\Delta_1}{\to} M_2 \cdots$$

is *quasi-lazy (leftmost)* if $\exists\{ n_i : \forall i \in \omega. n_i < n_{i+1} \}. \forall i \in \omega. \Delta_{n_i}$ is the lazy (leftmost) redex of M_i.

We will prove the following result:

THEOREM **2.1.2.2** *Let $M \in \mathbf{\Lambda}$. Then, M is strongly unsolvable iff M has an infinite quasi-lazy reduction.* □

The proof will be established by a series of lemmas.

NOTATION **2.1.2.3** (i) $M \xrightarrow{\Delta} N \stackrel{\text{def}}{=} M$ reduces to N by a one-step β-reduction and Δ is the redex in M that is contracted.

(ii) $M \to_{\text{lf}} N \stackrel{\text{def}}{=} M$ reduces to N by a one-step leftmost reduction.

(iii) $M \to_{\neq \text{lf}} N \stackrel{\text{def}}{=} M \xrightarrow{\Delta} N$ and Δ is not the leftmost redex of M.

(iv) For $\to \equiv \to_{\text{l}}, \to_{\text{lf}}, \to_{\neq \text{lf}}$ respectively, $\twoheadrightarrow$is the reflexive, transitive closure of $\to$; $\xrightarrow{+}$ is the transitive closure of $\to$.

LEMMA **2.1.2.4 (Advancement of Leftmost Reduction)** *If M has an infinite quasi-leftmost reduction,*

$$M \equiv M_0 \to M_1 \to M_2 \to \cdots,$$

then M has an infinite leftmost reduction

$$M \equiv N_1 \to_{\text{lf}} N_1 \to_{\text{lf}} \cdots$$

PROOF See [Bar84, page 328]. □

Recall the syntactic classification of Λ in Fact 1.2.1.5. We summarize the way leftmost (lazy) reduction alters or preserves different syntactic shapes as follows:

LEMMA **2.1.2.5** *(i) Suppose $(\lambda x.P)Q\vec{M} \xrightarrow{\Delta} N$ where $\to$ is a one-step β-reduction. Then,*

(1) $N \equiv \lambda x.N' \Rightarrow \Delta$ is the lazy (leftmost) redex and $|\vec{M}| = 0$.

(2) $N \equiv x\vec{L} \Rightarrow \Delta$ is the lazy (leftmost) redex.

(3) Δ is not the lazy (leftmost) redex $\Rightarrow N$ is in $l\beta rf$.

(ii) Suppose $(\lambda x.P)\vec{M} \xrightarrow{\Delta} \lambda x.N$ where Δ is not *the left-most redex. Then, $|\vec{M}| =$ 0.*

(iii) Suppose $M \twoheadrightarrow N$. If M is in whnf i.e. an abstraction or λ-free hnf, then so is N; i.e. $\twoheadrightarrow$ preserves the syntactic shape of whnfs.

PROOF Obvious. □

An immediate corollary is that if M has an infinite quasi-lazy reduction

$$M \equiv M_0 \to M_1 \to M_2 \cdots$$

then $\forall n \in \omega . M_n$ is in 1βrf.

LEMMA **2.1.2.6** *Let $M \in \Lambda$. Then,*

(i) $M \in \mathcal{F} \iff M \downarrow_1 \lambda x.L$.

(ii) $\exists \lambda\beta \vdash \vec{N}.M = x\vec{N} \iff M \downarrow_1 x\vec{L}$.

(iii) $M \in \mathbf{PO}_0 \iff M\uparrow_1$.

PROOF (i) Suppose $M \in \mathcal{F}$. Then, by the Church-Rosser property of β, $\exists Z.M \twoheadrightarrow Z \;\&\; \lambda x.N \twoheadrightarrow Z$. By (iii) of Lemma 2.1.2.5, Z is an abstraction. By the Advancement of Leftmost Reduction Lemma,

$$\exists M'.M \twoheadrightarrow_{\mathrm{lf}} M' \twoheadrightarrow_{\neq \mathrm{lf}} Z \equiv \lambda x.N'.$$

By Lemma 2.1.2.5(i)(3), M' is an abstraction. Since $\twoheadrightarrow_1$ is just $\twoheadrightarrow_{\mathrm{lf}}$ with a shorter extent, we conclude that $M \downarrow_1 \lambda x.L$ for some L.
The other direction is immediate. (ii) is proved in the same way as (i).
(iii) $M \in \mathbf{PO}_0 \iff$ the lhs of both (i) and (ii) are not satisfied. Result then follows from the observation that for any $M \in \Lambda$ precisely one of the following holds:

(1) $M \downarrow_1 \lambda x.L$;

(2) $M \downarrow_1 x\vec{L}$;

(3) $M\uparrow_1$.

□

In words, the above Lemma says that the syntactic class of a λ-term M can be determined by performing lazy reduction $\twoheadrightarrow_1$ on it as follows. If the reduction fails to terminate, then M is strongly unsolvable. Otherwise, it converges at either of the two subclasses of the whnf's: abstractions or λ-free hnf's.

COROLLARY **2.1.2.7** *If M is strongly unsolvable, then M has an infinite quasi-lazy reduction, i.e. $M\uparrow_1$.* □

COROLLARY **2.1.2.8 (Substitutivity of Strong Unsolvability)**

$$M \in \mathbf{PO}_0 \Rightarrow \forall L \in \Lambda.M[x := L] \in \mathbf{PO}_0.$$

PROOF Let $M \in \mathbf{PO}_0$. Then, by the Lemma(iii), $M\uparrow_l$, *i.e.* we have:

$$M \equiv M_0 \to_l M_1 \to_l M_2 \to_l \cdots.$$

By the substitutivity of β and that substitution preserves lazy redex, we have

$$M_0[x := L] \to_l M_1[x := L] \to_l M_2[x := L] \to_l \cdots.$$

That is $M[x := L]\uparrow_l$; hence by Lemma(iii), $M[x := L] \in \mathbf{PO}_0$. □

COROLLARY **2.1.2.9** *Let $M \in \Lambda$. Then,*

$$M \in \mathbf{PO}_0 \iff \forall \sigma : \mathrm{Var} \to \Lambda^o.M_\sigma\Uparrow.$$

PROOF The "⇒"-direction follows form the Substitutivity of Strong Unsolvability and Lemma 2.1.1.2. For "⇐", suppose for a contradiction, M is a functional term. Then, by Lemma(i), $M_\sigma\downarrow_l$ and hence $M_\sigma\Downarrow$, leading to a contradiction. Suppose, for another contradiction, $M \downarrow_l x\vec{N}$. Then, for any closed σ which maps x to $\mathbf{YK}$ say, we have $M_\sigma\Downarrow$, contradicting the premise. We are therefore left with the only alternative and that is $M \in \mathbf{PO}_0$. □

COROLLARY **2.1.2.10** $M \in \mathbf{PO}_0 \Rightarrow \forall \vec{N} \in \Lambda.M\vec{N} \in \mathbf{PO}_0$. □

PROPOSITION **2.1.2.11** *If M has an infinite quasi-lazy reduction, then M is strongly unsolvable.*

PROOF Let the infinite reduction be

$$M \equiv M_0 \stackrel{\Delta_0}{\to} M_1 \stackrel{\Delta_1}{\to} M_2 \stackrel{\Delta_2}{\to} \cdots$$

By Lemma 2.1.2.5(iii) and the infinite nature of the reduction, $\forall n \in \omega$, M_n is in $l\beta$rf; and so, lazy and leftmost redex coincide. By the Advancement of Leftmost Reduction Lemma, $\exists\langle N_i : i \in \omega\rangle$ such that

$$M \equiv N_0 \to_{lf} N_1 \to_{lf} N_2 \to_{lf} \cdots$$

which is lazy reduction, since $\forall i \in \omega$, N_i is in $l\beta$rf; whence $M\uparrow_l$. The result then follows by an appeal to Lemma 2.1.2.6(iii). □

We have thus completed the proof of the Theorem.

2.1.3 Partial Recursive Functions and λ-Definability Revisited

A celebrated computability result of long standing is that, of the partial numeric functions, exactly the partial recursive functions are λ-definable.

A *partial numeric function* is a partial mapping $\phi : \omega^p \rightharpoonup \omega$ for some $p \geqslant 1$. Let ϕ, ψ be two partial functions. We say that $\phi(\vec{n})$ and $\psi(\vec{n})$ are *Kleene equal*, denoted $\phi(\vec{n}) \simeq \psi(\vec{n})$ if

$$\phi(\vec{n}) = a \iff \psi(\vec{n}) = a.$$

That is, if the lhs is defined then so is the rhs and they have identical values and vice versa.

The original definition of λ-definability was as follows:

> A partial numeric function ϕ with p argument(s) is *λ-definable* if there exists an $F \in \Lambda$ such that $\forall \vec{n} \subseteq \omega^p$,
>
> $$F^{\ulcorner}\vec{n}^{\urcorner} =_\beta \begin{cases} {}^{\ulcorner}\phi(\vec{n})^{\urcorner} & \text{if } \phi(\vec{n}) \text{ is defined,} \\ \text{has no nf} & \text{else;} \end{cases}$$
>
> where ${}^{\ulcorner}(-)^{\urcorner}$ is an encoding of the numerals in Λ.

As mentioned earlier, the first breakthrough in ascribing meanings to λ-terms came with the Barendregt-Wadsworth perspective of characterizing undefinedness in terms of unsolvability. In keeping with the spirit of that proposal, the notion of λ-definability was modified by replacing the "else-clause" of the above definition by

> "$F^{\ulcorner}\vec{n}^{\urcorner}$ is unsolvable if $\phi(\vec{n})$ is undefined."

In line with the *leitmotif* of this work — *strong unsolvability characterizes undefinedness in the lazy regime*, we shall modify the definition of λ-definability accordingly.

DEFINITION **2.1.3.1** A partial numeric function $\phi : \omega^p \rightharpoonup \omega$ is *weakly lazy λ-definable* if $\exists F \in \Lambda$ such that $\forall \vec{n} \in \omega^p$,

$$F^{\ulcorner}\vec{n}^{\urcorner} =_\beta {}^{\ulcorner}\phi(\vec{n})^{\urcorner} \quad \text{if } \phi(\vec{n}) \text{ is defined;}$$

$$F^{\ulcorner}\vec{n}^{\urcorner}\Uparrow \quad \text{else.}$$

We say that ϕ is weakly lazily λ-definable by F.

The main result in this section is to re-establish Kleene's pre-eminent result in the lazy setting as follows:

THEOREM **2.1.3.2** *A partial numeric function is partial recursive iff it is weakly lazily λ-definable.* □

In the proof of this Theorem, which will occupy the rest of this section, references will be made to existing computability results in lambda calculus, especially to §6.3 and §8.4 in [Bar84]. Recall the definitions of the initial functions, namely, the projection functions, successor, "test-for-zero", primitive recursion and minimalization. We will recapitulate the combinators that simulate the various "data structures" and their respective operations.

- *Truth values*: $\mathbf{T} \equiv \lambda xy.x, \mathbf{F} \equiv \lambda xy.y$. Then,

  $$\mathbf{T}MN \twoheadrightarrow M, \qquad \mathbf{F}MN \twoheadrightarrow N.$$

  Note that $\mathbf{T} \equiv \mathbf{K}$ and $\mathbf{KI} \twoheadrightarrow \mathbf{F}$.

- *Conditionals*: Let B (Boolean) be a term. Define the construct

  $$\textbf{if } B \textbf{ then } M \textbf{ else } N \stackrel{\text{def}}{=} BMN.$$

  Then,

  $$\frac{B \twoheadrightarrow \mathbf{T}}{\textbf{if } B \textbf{ then } M \textbf{ else } N \twoheadrightarrow M}$$

  $$\frac{B \twoheadrightarrow \mathbf{F}}{\textbf{if } B \textbf{ then } M \textbf{ else } N \twoheadrightarrow N}$$

  If M or N are in whnf, then the above proof rules are still sound if $\twoheadrightarrow$ is replaced by $\Downarrow$. Of course, if B converges to neither $\mathbf{T}$ nor $\mathbf{F}$, then we have no way of characterizing BMN.

- *Composition*: $M \circ N \stackrel{\text{def}}{=} \lambda x.M(Nx)$.

- *Pairing*: $[M,N] \stackrel{\text{def}}{=} \lambda z.zMN$ and

  $$(M)_0 \stackrel{\text{def}}{=} M\mathbf{T}, \qquad (N)_1 \stackrel{\text{def}}{=} M\mathbf{F}.$$

  We have $([M_0,M_1])_i \twoheadrightarrow M_i$, $i = 0,1$. Note that this pairing does not satisfy the conventional restraint of being surjective; in fact, no such surjective pairing is λ-definable.

- *finite sequence*: $\langle M_0,\cdots,M_n\rangle \stackrel{\text{def}}{=} \lambda z.zM_0\cdots M_n$. Define

$$\mathbf{U}_i^n \stackrel{\text{def}}{=} \lambda x_0 \cdots x_n.x_i, \qquad \mathbf{P}_i^n \stackrel{\text{def}}{=} \lambda z.z\mathbf{U}_i^n.$$

Then, we have $\mathbf{P}_i^n\langle M_0, \cdots, M_n\rangle \twoheadrightarrow M_i$.

- *Numerals*: For each $n \in \omega$, a term $\ulcorner n \urcorner$ is defined as follows:

$$\ulcorner 0 \urcorner \stackrel{\text{def}}{=} \mathbf{I}, \qquad \ulcorner n+1 \urcorner \stackrel{\text{def}}{=} [\mathbf{F}, \ulcorner n \urcorner].$$

Note that the numerals are in whnfs and are distinct nf's.

- The *successor* $\mathbf{S}^+$ and *test-for-zero* **Zero** are defined as:

$$\mathbf{S}^+ \stackrel{\text{def}}{=} \lambda x.[\mathbf{F}, x], \qquad \mathbf{Zero} \stackrel{\text{def}}{=} \lambda x.x\mathbf{T}.$$

It should be clear that:

$$\mathbf{S}^+\ulcorner n \urcorner \twoheadrightarrow \ulcorner n+1 \urcorner,$$

$$\mathbf{Zero}\ulcorner 0 \urcorner \twoheadrightarrow \mathbf{T}, \quad \mathbf{Zero}\ulcorner n+1 \urcorner \twoheadrightarrow \mathbf{F}.$$

Observe that Definition 2.1.3.1 coincides with the definition of λ-definability of total numeric functions in [Bar84, Definition 6.3.1] in the case of total functions. Hence, the following is immediate:

LEMMA **2.1.3.3** *The total recursive functions are weakly lazily λ-definable as partial functions.* □

It is probably helpful to recall the following definition:

DEFINITION **2.1.3.4** Let $\mathcal{P}$ be a class of partial numeric functions.

(i) $\mathcal{P}$ is *closed under composition* if $\chi, \psi_i \in \mathcal{P}$ implies that ϕ defined as follows

$$\phi(\vec{n}) \simeq \chi(\psi_1(\vec{n}), \cdots, \psi_m(\vec{n}))$$

belongs to $\mathcal{P}$.
Note that $\chi(\psi_1(\vec{n}), \cdots, \psi_m(\vec{n}))$ denotes the partial numeric function ϕ such that $\phi(\vec{n}) = a$ iff

$$\exists a_1, \cdots, a_m.[\psi_i(\vec{n}) = a_i \text{ for } 1 \leqslant i \leqslant m \ \& \chi(\vec{a}) = a].$$

(ii) $\mathcal{P}$ is *closed under minimalization* if $\mathcal{P}$ contains all functions ϕ defined by

$$\phi(\vec{n}) = \mu m.[\chi(\vec{n}, m) = 0]$$

where $\chi \in \mathcal{P}$ and is total.

(iii) The class of *partial recursive function*, denoted $\mathcal{PR}$, is the least class of partial numeric functions containing the total recursive functions and is closed under composition and minimalization.

LEMMA **2.1.3.5** *Let $F \in \Lambda$ weakly lazily λ-define a partial numeric function ϕ. Then, $\forall \vec{n} \subseteq \omega$,*

$$F^{\ulcorner}\vec{n}^{\urcorner}\mathbf{KII} = \mathbf{I} \quad \textit{if } \phi(\vec{n}) \textit{ is defined;}$$

$$F^{\ulcorner}\vec{n}^{\urcorner}\mathbf{KII}\Uparrow \quad \textit{else.}$$

PROOF It should be clear that $\phi(\vec{n})$ is defined implies that $F^{\ulcorner}\vec{n}^{\urcorner}\mathbf{KII} = \mathbf{I}$. Also, $\phi(\vec{n})$ is not defined implies that $F^{\ulcorner}\vec{n}^{\urcorner}$ is strongly unsolvable which implies that $F^{\ulcorner}\vec{n}^{\urcorner}\mathbf{KII}$ is also strongly unsolvable. □

The following Lemma uses a trick first proposed by Lercher as documented in [Bar84].

LEMMA **2.1.3.6** *The weakly lazily λ-definable partial numeric functions are closed under composition.*

PROOF Let $\phi(\vec{n}) \simeq \chi(\psi_1(\vec{n}), \cdots, \psi_m(\vec{n}))$ where $\chi, \psi_1, \cdots, \psi_m$ are lazily λ-defined by $G, H_1, \cdots, H_m$ say. Define

$$F \equiv \lambda\vec{x}.(H_1\vec{x}\mathbf{KII}) \cdots (H_m\vec{x}\mathbf{KII})G(H_1\vec{x}) \cdots (H_m\vec{x}).$$

We claim that F lazily λ-defines ϕ. Indeed, if one of the ψ_i's is undefined at $\vec{n}$, say ψ_j, then the j-th "jamming-factor" $H_j{}^{\ulcorner}\vec{n}^{\urcorner}\mathbf{KII}$ is strongly unsolvable. Consequently, $F^{\ulcorner}\vec{n}^{\urcorner}$ is strongly unsolvable. Conversely, if all the $\phi_i(\vec{n})$ are defined, then by the previous Lemma, all the "jamming-factors" converge lazily to $\mathbf{I}$. Hence,

$$F^{\ulcorner}\vec{n}^{\urcorner} \twoheadrightarrow_l G(H_1{}^{\ulcorner}\vec{n}^{\urcorner}) \cdots (H_m{}^{\ulcorner}\vec{n}^{\urcorner}),$$

which is as it should be. □

PROPOSITION **2.1.3.7** *Let P be such that $\forall n \in \omega.P^{\ulcorner}n^{\urcorner} = \mathbf{F}$. Then, μP is strongly unsolvable.*

PROOF Denote $\Theta \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy))$. Recall that

$$\forall M \in \Lambda.\Theta M \twoheadrightarrow_l M(\Theta M).$$

Define

$$\mathbf{H}_P \stackrel{\text{def}}{=} \Theta(\lambda hz.\text{if } Pz \text{ then } z \text{ else } h(\mathbf{S}^+z))$$

and

$$\mu P \stackrel{\text{def}}{=} \mathbf{H}_P{}^{\ulcorner}0^{\urcorner}.$$

Then,

$$\mathbf{H}_P{}^{\ulcorner}n^{\urcorner} \twoheadrightarrow_l \textbf{if } P^{\ulcorner}n^{\urcorner} \textbf{ then } {}^{\ulcorner}n^{\urcorner} \textbf{ else } \mathbf{H}_P(\mathbf{S}^{+\ulcorner}n^{\urcorner}).$$

Therefore, μP has the following reduction path:

$$\begin{aligned}
& \mu P \equiv \mathbf{H}_P{}^{\ulcorner}0^{\urcorner} \\
\twoheadrightarrow_l\ & \textbf{if } P^{\ulcorner}0^{\urcorner} \textbf{ then } {}^{\ulcorner}0^{\urcorner} \textbf{ else } \mathbf{H}_P(\mathbf{S}^{+\ulcorner}0^{\urcorner}) \\
\twoheadrightarrow_l\ & \mathbf{H}_P(\mathbf{S}^{+\ulcorner}0^{\urcorner}) \\
\twoheadrightarrow_l\ & \mathbf{H}_P(\mathbf{S}^{+}(\mathbf{S}^{+\ulcorner}0^{\urcorner})) \\
\twoheadrightarrow_l\ & \cdots
\end{aligned}$$

If follows that μP has an infinite lazy reduction and is therefore strongly unsolvable by the Operational Characterization Theorem. □

LEMMA **2.1.3.8** *The weakly lazily λ-definable partial numeric functions are closed under minimalization.*

PROOF Let $\phi(\vec{n}) \simeq \mu m.[\chi(\vec{n},m) = 0]$ where χ is total and lazily λ-definable by, say, G. Define $F \equiv \lambda\vec{x}.\mu y.[\mathrm{Zero}G(\vec{x}y)]$. If $\phi(\vec{n})$ is defined, then $\exists m.\chi(\vec{n},m) = 0$. Hence, $F^{\ulcorner}n^{\urcorner} = {}^{\ulcorner}\phi(\vec{n})^{\urcorner}$ by Proposition 6.3.9(ii) in *op. cit.* and the observation that lazy reduction alone suffices. If $\phi(\vec{n})$ is undefined, then $\forall m.\chi(\vec{n},m) \neq 0$ and so $\forall m.\mathbf{Zero}(G^{\ulcorner}\vec{n}^{\urcorner}m^{\urcorner}) = \mathbf{F}$. Hence, by the preceding Proposition, $F^{\ulcorner}\vec{n}^{\urcorner} = \mu m.[\mathbf{Zero}(G^{\ulcorner}\vec{n}^{\urcorner}m^{\urcorner})]$ is strongly unsolvable. Thus, F lazily λ-defines ϕ. □

LEMMA **2.1.3.9** *If F weakly lazily λ-defines a partial numeric function ϕ. Then,*

$$\phi(\vec{n}) = m \iff F^{\ulcorner}\vec{n}^{\urcorner} = {}^{\ulcorner}m^{\urcorner}.$$

PROOF "$\Rightarrow$": By definition.
"$\Leftarrow$": If $F^{\ulcorner}\vec{n}^{\urcorner} = {}^{\ulcorner}m^{\urcorner}$, *i.e.* not strongly unsolvable, $\phi(\vec{n})$ is defined and $\phi(\vec{n}) = m'$. But then, ${}^{\ulcorner}m^{\urcorner} = {}^{\ulcorner}m'^{\urcorner}$, hence $m = m'$. □

Now, we are in a position to prove the Theorem.
"$\Rightarrow$": By Lemmas 2.1.3.3, 2.1.3.6 and 2.1.3.8.
"$\Leftarrow$": Let ϕ be λ-definable by F. Then, by the preceding Lemma,

$$\phi(\vec{n}) = m \iff F^{\ulcorner}n^{\urcorner} = {}^{\ulcorner}m^{\urcorner}.$$

and we are done. □

2.2 Tree Semantics

In this section, we introduce Böhm trees, weak Böhm trees and Longo trees and investigate their inter-relationships. The classical notion of Böhm trees as structures capturing the operational meanings of λ-terms is incorrect in the lazy regime. We propose two revised tree-structures: weak Böhm trees and Levy-Longo trees. The former capture the operational essence of the lazy evaluation and the latter provide denotations for λ-terms enabling fine distinctions to be made between their operational behaviours.

2.2.1 Böhm Trees as Operational Semantics

Böhm trees were first introduced by C. Böhm in the proof of his celebrated result on the separability of λ-terms [Boh68]. For each $M \in \Lambda$, one can define its corresponding Böhm tree, $\mathsf{BT}(M)$, which conveys the operational meanings of M in an essentially "strict" regime.

> Terms and their BT's roughly relate to each other as a real number relates to its continued fraction expression. If M has a nf, then $\mathsf{BT}(M)$ is finite. In this respect, nf's correspond to rational numbers. [Bar84, page 215].

The set of all Böhm-like trees, considered as partial functions on Seq—the set of finite sequences of ω, ordered by inclusion forms a Scott domain, called the *Böhm tree domain* which is denoted $\mathcal{B}$. All unsolvable terms have the same Böhm tree which is the least element of the domain, namely the tree with singleton node $\perp$; and all normalizable terms are characterized as precisely those terms which have finite maximal Böhm trees.

Given a Böhm tree B, one can define a canonical lambda term $M(B)$, which has B as its Böhm tree. Since a Böhm tree B of depth $N > 1$ has obvious (finite) approximants, namely, B truncated at depth $n < N$, denoted $B^{(n)}$ and that $B = \bigsqcup_{n \leqslant N} B^{(n)}$, a notion of approximation can therefore be imported to the set of λ-terms. Formally, we say that M is *Böhm tree less than* N, denoted $M \sqsubseteq_B N$ iff $\mathsf{BT}(M) \subseteq_B \mathsf{BT}(N)$. More significantly, one can define a topology, usually called the *(Böhm) tree topology* as the smallest one which makes the map $\mathsf{BT}: \Lambda \to \mathcal{B}$ continuous where $\mathcal{B}$ is endowed with the Scott topology.

Considered as functions from $\Lambda \times \Lambda \to \Lambda$ and $\Lambda \to \Lambda$ respectively, the λ-calculus application and the context operator $C[\ ]$ can be proved to be continuous. Topological techniques turned out to be crucial in the proofs of various seminal results in λ calculus, notably the *sequentiality* and *stability* of λ-calculus due to Gerald Berry [Ber78], the Genericity Lemma (see Chapter 1) and also in the construction of the Böhm tree λ-model [Bar84, pages 486-491].

Giuseppe Longo in his paper [Lon84] highlights and advocates the use of Böhm tree-like domains in the study of the semantics of λ-calculus based functional languages. We refer the reader to [BCL85] for a comprehensive survey on the full-abstraction problem in which the authors employ Böhm tree domain in the study of the semantics of PCF which is a functional language based on the typed λ-calculus augmented with constants for recursion and elementary arithmetic operations.

Just as solvable lambda terms correspond to the computationally well-behaved terms only in the "strict" regime, so Böhm trees only correctly express the computational contents of λ-terms in the "strict" regime. Since this point is not very well-known, we emphasize the observation as follows:

Böhm trees capture strict semantics.

DEFINITION **2.2.1.1** (i) Let $\Sigma_B \stackrel{\text{def}}{=} \langle \{\bot\} \cup \{\lambda\vec{x}.y : \vec{x}, y \subseteq \mathsf{Var}\}, \leqslant_B \rangle$ where $\leqslant_B$ is the least partial order satisfying $\forall s \in \Sigma_B . \bot \leqslant_B s$. As usual, we write $s <_B t$ to mean $s \leqslant_B t \;\&\; s \neq t$. Note that $s \leqslant_B t \Rightarrow s = \bot$ or $s = t$ and $s <_B t \Rightarrow s = \bot$.

(ii) Let $M \in \Lambda$. The *Böhm tree* of M, $\mathsf{BT}(M)$, is a Σ-labelled tree defined informally as follows:

$$\mathsf{BT}(M) = \quad \bot \qquad \text{if } M \text{ is unsolvable;}$$

$$\mathsf{BT}(M) = \quad \lambda\vec{x}.y \qquad \text{if } M \text{ is solvable and has principal hnf}$$

$$\mathsf{BT}(M_1) \cdots \mathsf{BT}(M_m) \qquad \lambda\vec{x}.yM_1 \cdots M_m$$

(iii) Let $\mathcal{B}$ be the set of all Σ_B-labelled trees (called *Böhm-like trees*) and

$$\Lambda\mathcal{B} \stackrel{\text{def}}{=} \{\phi : \phi \text{ is a } \Sigma_B\text{-labelled tree } \;\&\; \exists M \in \Lambda . \mathsf{BT}(M) = \phi\}.$$

Define a binary relation $\subseteq_B$ on $\mathcal{B}$ as the least partial order such that for $B_1, B_2 \in \Lambda\mathcal{B} . B_1 \subseteq_B B_2$ iff B_2 is obtained from B_1 by replacing some occurrence(s) of $\bot$ in B_1 by the Böhm tree of some λ-term(s).
More formally, $B_1 \subseteq_B B_2$ iff

(1) $T_{B_1} \subseteq T_{B_2}$ *i.e.* B_1 is a subset of B_2 as underlying trees.

(2) $\forall \alpha \in \mathsf{Seq} . B_1(\alpha)\downarrow \Rightarrow B_1(\alpha) \leqslant_B B_2(\alpha)$.

(3) $\forall \alpha \in \mathsf{Seq} . B_2(\alpha)\downarrow \;\&\; B_1(\alpha)\uparrow \Rightarrow \exists \beta < \alpha . B_1(\beta) <_B B_2(\beta)$.

We will show in the later section on Longo trees, *a fortriori*, that the two formulations of $\subseteq_B$ above are equivalent. We define for $M, N \in \Lambda$,

$$M \trianglelefteq_B N \stackrel{\text{def}}{=} \mathsf{BT}(M) \subseteq_B \mathsf{BT}(N).$$

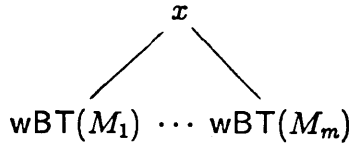
2.2.2 Weak Böhm Trees

Taking the cue form Böhm trees, we construct weak Böhm trees which induce an operational preorder on λ-terms in the lazy regime.

DEFINITION **2.2.2.1** (i) Let $\Sigma_w \stackrel{\text{def}}{=} \langle \{\bot\} \cup \{\Lambda\} \cup \mathrm{Var}, \leqslant_w \rangle$ where $\leqslant_w$ is the least partial order on Σ_w such that $\forall s \in \Sigma_w . \bot \leqslant_w s$. The *weak Böhm tree* of a λ-term M, $\mathsf{wBT}(M)$, is a Σ_w-labelled tree defined informally as follows:

(1) $\mathsf{wBT}(M) = \bot$ if M is strongly unsolvable;

(2) $\mathsf{wBT}(M) = \Lambda$ if M is a λ-abstraction;

(3) $\mathsf{wBT}(M) =$  if M has λ-free hnf $xM_1 \cdots M_m$

(ii) Let $\mathcal{WB}$ be the set of all Σ_w-labelled trees (called *weak Böhm-like trees*) and $\Lambda\mathcal{WB} = \{\phi : \phi \text{ is a } \Sigma_w\text{-labelled tree } \& \exists M \in \Lambda . \mathsf{wBT}(M) = \phi\}$. Define a binary relation $\subseteq_w$ on $\mathcal{WB}$ as the least partial order such that for $B_1, B_2 \in \mathcal{WB}$, $B_1 \subseteq_w B_2$ iff B_2 is obtained from B_1 by replacing some occurrence(s) of $\bot$ in B_1 by the weak Böhm tree of some λ-term. More formally, $B_1 \subseteq_w B_2$ iff

(1) $T_{B_1} \subseteq T_{B_2}$,

(2) $\forall \alpha \in \mathsf{Seq} . B_1(\alpha)\downarrow \Rightarrow B_1(\alpha) \leqslant B_2(\alpha)$,

(3) $\forall \alpha \in \mathsf{Seq} . B_1(\alpha)\uparrow \;\&\; B_2(\alpha)\downarrow \Rightarrow \exists \beta < \alpha . B_1(\beta) <_w B_2(\beta)$.

For $M, N \in \Lambda$, we define

$$M \sqsubseteq_w N \stackrel{\text{def}}{=} \mathsf{wBT}(M) \subseteq_w \mathsf{wBT}(N).$$

REMARK **2.2.2.2** There is no straightforward link between the operational preorder induced by Böhm trees and that induced by weak Böhm trees. Let $M, N \in \Lambda$, then

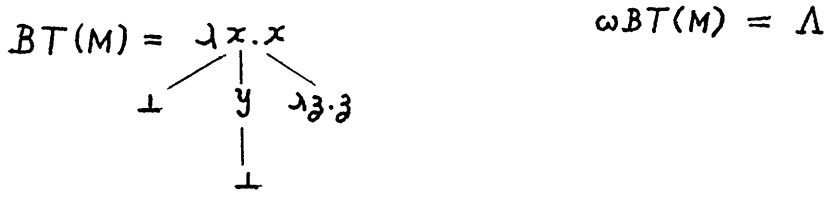
$$M \sqsubseteq_B N \not\Rightarrow\not\Leftarrow M \sqsubseteq_w N$$

Just consider:
"$\not\Rightarrow$" $M \equiv \lambda x.\Omega, N \equiv \Omega$
"$\not\Leftarrow$" $M \equiv \lambda x.x, N \equiv \lambda x.\Omega$

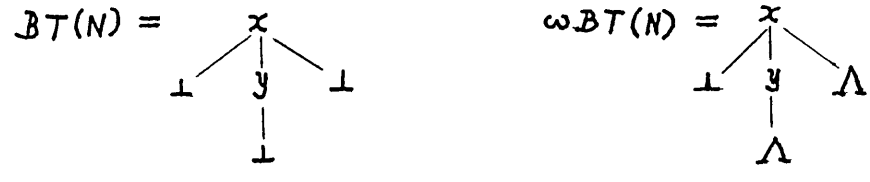
The incompatibility of the two operational preorders is due to the following:

- wBTs afford finer distinction between unsolvable terms than BTs; wBTs "filter out" only the $\mathbf{PO}_0$-terms.
- BTs afford (much!) finer distinction between abstractions than wBTs.

EXAMPLE **2.2.2.3** (i) $M \equiv \lambda x.x\Omega(y(\mathbf{YK}))\lambda z.z$



(ii) $N \equiv x\Omega(y(\mathbf{YK}))\lambda x.\Omega$



LEMMA **2.2.2.4** *$\langle \mathcal{WB}, \subseteq_W \rangle$ is a consistently complete algebraic cpo.*

PROOF By the proof of Lemma 2.2.3.8, *a fortriori.* □

Note that in $\Lambda\mathcal{WB}$, the least element corresponds to the strongly unsolvable terms, but unlike Böhm trees, the set of finite maximal weak Böhm trees do not correspond to the set of normalizable λ-terms. Nevertheless, wBTs induce a "correct" notion of operational preorder between terms in the lazy regime, except that the preorder induced is dismally coarse.

2.2.3 Levy-Longo Trees

We introduce Levy-Longo trees which are a natural extensions of the weak Böhm trees. Levy-Longo trees capture the operational content of λ-terms in the lazy regime and that they afford an appropriate discriminatory mechanism between terms. Levy knew of this operational preorder long ago and published it, albeit not in a tree-like fashion, in [Lev75]. Giuseppe Longo [Lon83] later investigated a class of λ-models called Plotkin-Scott-Engeler Algebras (see Chapter 3) which reflect the operational preorder induced by Levy-Longo trees (which Longo simply called "trees" in *op. cit.*) internally in their order structures. We will take up the same theme in the sequel. From now on, we will abbreviate Levy-Longo trees as Longo trees.

DEFINITION **2.2.3.1** Let

$$\Sigma_L \stackrel{\text{def}}{=} \langle \{\top\} \cup \{\lambda\vec{x}.\bot : \vec{x} \subseteq \mathsf{Var}\} \cup \{\lambda\vec{x}.y : \vec{x}, y \subseteq \mathsf{Var}\}, \leqslant_L \rangle$$

where $\leqslant_L$ is the least partial order on Σ_L satisfying:

(1) $\forall s \in \Sigma_L \backslash \{\top\}.\lambda\vec{x}.\bot \leqslant_L \lambda\vec{x}.s$,

(2) $\lambda\vec{x}.\bot \leqslant_L \top$.

Note that $s \leqslant_L t \Rightarrow s = t$ or $s = \lambda\vec{x}.\bot$. $\bot$ is the least element in $\langle \Sigma_L, \leqslant_L \rangle$ and that $\top$ and $\lambda\vec{x}.y$ are the maximal elements. In fact, we can say more:

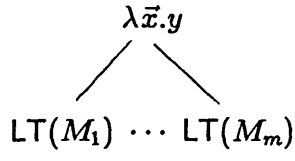
LEMMA **2.2.3.2** *$\langle \Sigma_L, \leqslant_L \rangle$ is a consistently complete algebraic cpo, i.e. a Scott domain.*

PROOF $\langle \Sigma_L, \leqslant_L \rangle$ is by definition a poset. Let $X \subseteq \Sigma_L$ be consistent or directed. If X contains a maximal element a, then clearly $a = \bigsqcup X$. If not, then elements of X must all be of the form $\lambda\vec{x}.\bot$. If $|X|$ is not bounded, then $\bigsqcup X = \top$. If $|X|$ is bounded, then $\bigsqcup X = \lambda x_1 \cdots x_N.\bot$ where $N = \max\{n : \lambda x_1 \cdots x_n.\bot \in X\}$. The finite elements of $\langle \Sigma_L, \leqslant_L \rangle$ are just $\Sigma_L \backslash \{\top\}$. □

DEFINITION **2.2.3.3** (i) The *Longo tree* of M, $\mathsf{LT}(M)$, is a Σ_L-labelled tree defined informally as follows:

(1) $\mathsf{LT}(M) = \top$ if $M \in \mathbf{PO}_\infty$,

(2) $\mathsf{LT}(M) = \lambda x_1 \cdots x_n.\bot$ if $M \in \mathbf{PO}_n$ such that $n \in \omega$,

(3) $\mathsf{LT}(M) =$  if M is solvable and has principal hnf $\lambda\vec{x}.yM_1 \cdots M_m, m \geqslant 0$

(ii) Let $\mathcal{L}$ be the set of all Σ_L-labelled trees (called *Longo-like trees)* and

$$\Lambda\mathcal{L} \stackrel{\text{def}}{=} \{\phi : \phi \text{ is a } \Sigma_L\text{-labelled tree } \& \exists M \in \Lambda.\mathsf{LT}(M) = \phi\}.$$

Define a partial order $\subseteq_L$ on $\mathcal{L}$ as follows: For $L_1, L_2 \in \mathcal{L}.L_1 \subseteq_L L_2$ iff L_2 is obtained from L_1 by

(a) replacing some occurrence(s) of $\bot$ in L_1 by the Longo tree of some λ-term and/or

(b) replacing some occurrence(s) of $\lambda\vec{x}.\bot$ in L_1 by $\top$.

(iii) More formally, $L_1 \subseteq_L L_2$ iff

(1) $T_{L_1} \subseteq T_{L_2}$,

(2) $\forall\alpha \in \mathsf{Seq}.L_1(\alpha)\downarrow \Rightarrow L_2(\alpha)\downarrow \;\&\; L_1(\alpha) \leqslant_L L_2(\alpha)$,

(3) $\forall \alpha \in \mathsf{Seq}.L_1(\alpha)\uparrow \,\&\, L_2(\alpha)\downarrow \Rightarrow \exists \beta < \alpha.L_1(\beta) <_L L_2(\beta)$.

Note that (2) subsumes (1).

$M \sqsubseteq_L N \stackrel{\text{def}}{=} \mathsf{LT}(M) \subseteq_L \mathsf{LT}(N)$.

LEMMA **2.2.3.4** *The formulations of $\subseteq_L$ in (ii) and (iii) of the previous definition are equivalent.*

PROOF "(ii) $\Rightarrow$ (iii)": Suppose $L_1 \subseteq_L L_2$ according to (ii). Clearly, we have (1) $T_{L_1} \subseteq T_{L_2}$. Suppose $L_1(\alpha)\downarrow$, then observe that either rules (a) and (b) do *not* apply at all $\beta \leqslant \alpha$ or they apply at α (in which case, $L_1(\alpha) = \lambda\vec{x}.\bot$), since rules (a) and (b) can only possibly apply at nodes with labels $\lambda\vec{x}.\bot$ and such nodes are by construction terminal nodes. If the former, then trivially $L_2(\alpha)\downarrow \& L_1(\alpha) = L_2(\alpha)$. If the latter is true, then since $\forall s \in \Sigma_L \setminus \{\top\}.\lambda\vec{x}.\bot \leqslant_L \lambda\vec{x}.s$ and $\lambda\vec{x}.\bot \leqslant_L \top$, we therefore also have $L_2(\alpha)\downarrow \,\&\, L_1(\alpha) \leqslant_L L_2(\alpha)$. Suppose $L_2(\alpha)\downarrow \,\&\, L_1(\alpha)\uparrow$, then this can only arise as a result of an application of rule (a) at some node $\beta < \alpha$. Clearly, $L_1(\beta) <_L L_2(\beta)$.
"(iii) $\Rightarrow$ (ii)": It suffices to prove the following:

> For $\alpha \in \mathsf{Seq}.L_2(\alpha)\downarrow$, *either*
>
> $$\forall \beta \leqslant \alpha.L_1(\beta) = L_2(\beta);$$
>
> That is to say, rules (a) or (b) do not apply at all $\beta \leqslant \alpha$ *or* rules (a) or (b) apply at some $\beta \leqslant \alpha$ and $L_2(\alpha)$ is part of the replaced Longo tree in rule (a) or (b) respectively.

Now suppose $L_2(\alpha)\downarrow$. We consider two cases:

- $L_1(\alpha)\downarrow$: By (2), $L_1(\alpha) \leqslant L_2(\alpha)$. Then, either $L_1(\alpha) = L_2(\alpha)$ or

  $$L_1(\alpha) = \lambda\vec{x}.\bot \;\&\; L_2(\alpha) = \lambda\vec{x}.s \text{ where } s \in \Sigma_L, \text{or} \bot.$$

  Suppose $L_1(\alpha) = L_2(\alpha)$, then $\forall \beta < \alpha.L_1(\beta) \neq \lambda\vec{x}.\bot$, and so, because of (2) i.e. $L_1(\beta) \leqslant_L L_2(\beta)$, we thus have $L_1(\beta) = L_2(\beta)$. Suppose $L_1(\alpha) = \lambda\vec{x}.\bot$, then clearly either rule (a) or (b) applies at α.

- $L_1(\alpha)\uparrow$: By (3), $\exists \beta < \alpha.L_1(\beta) <_L L_2(\beta)$. Again, by definition of $\leqslant_L$, $L_1(\beta) = \lambda\vec{x}.\bot$ and $L_2(\beta) = \lambda\vec{x}.s$ where $s \in \Sigma_L \setminus \{\top\}$. Note that $L_2(\beta) \neq \top$, for otherwise, $L_2(\beta)$ would be a terminal node of L_2, leading to a contradiction. Hence, we conclude that $L_2(\alpha)$ is part of the Longo tree replacing the "$\bot$" at $L_1(\beta)$, since any subtree of a Longo tree is the Longo tree of some λ-term.

□

EXAMPLE **2.2.3.5** (i) Let $l_1 \equiv \mathsf{LT}(x\Omega\Omega)$ and $l_2 \equiv \mathsf{LT}(x\Omega\Omega\Omega)$. Then, $l_1 \not\subseteq_L l_2$. This can be seen immediately from the informal definition of $\subseteq_L$. For the formal definition, note that $l_2\langle 2\rangle\downarrow$ & $l_1\langle 2\rangle\uparrow$ but $\neg[\exists\alpha < \langle 2\rangle.l_1(\alpha) <_L l_2(\alpha)]$.

(ii) Let l_1, l_2 be as before. We assert that $l_2 \not\subseteq_L l_1$. This should be clear, because $l_2\langle 2\rangle\downarrow$ but $l_1\langle 2\rangle\uparrow$.

(iii) Let $l_1 \equiv \mathsf{LT}(\lambda x.xxyz)$ and $l_2 \equiv \mathsf{LT}(\lambda x.xx(y\Omega)z)$. Then, $l_1 \not\subseteq_L l_2$. Note that $l_2\langle 1,0\rangle\downarrow$ & $l_1\langle 1,0\rangle\uparrow$ but $\neg[\exists\alpha < \langle 1,0\rangle.l_1(\alpha) <_L l_2(\alpha)]$.

OBSERVATION **2.2.3.6** (i) The maximal elements of $\langle\mathcal{L}, \subseteq_L\rangle$ are (finite or infinite) Longo trees such that all (finite) terminal nodes have labels which are maximal in $\langle\Sigma_L, \leqslant_L\rangle$ *i.e.* of the form $\lambda\vec{x}.y$ or $\top$.

(ii) For $l \in \Lambda\mathcal{L}.\forall\alpha \in \mathsf{Seq}.l(\alpha)\downarrow \Rightarrow \forall\beta < \alpha.l(\beta) = \lambda\vec{x}.y$ which is maximal in $\langle\Sigma_L, \leqslant_L\rangle$. Note that by construction, $l(\alpha) = \lambda\vec{x}.\bot$ or $\top$ implies that α is a terminal node.

REMARK **2.2.3.7** $\phi \in \mathcal{L}$ if ϕ: Seq $\rightharpoonup \Sigma_L$ satisfies the following:

(i) T_ϕ is a tree,

(ii) $\forall\alpha \in \mathsf{Seq}.[\exists\beta > \alpha.\phi(\beta)\downarrow \Rightarrow \phi(\alpha) = \lambda\vec{x}.y]$.

This is clear by definition.

LEMMA **2.2.3.8** *$\langle\mathcal{L}, \subseteq_L\rangle$ is a consistently complete algebraic cpo.*

PROOF $\langle\mathcal{L}, \subseteq_{\mathcal{L}}\rangle$ is clearly a poset by definition of $\subseteq_L$. We show that $\langle\mathcal{L}, \subseteq_{\mathcal{L}}\rangle$ is directed complete and consistently complete simultaneously to save writing. Suppose $\chi \subseteq \mathcal{L}$ is directed (consistent). Then $\forall\alpha \in$ Seq:

$$\mathrm{lab}_\chi(\alpha) \stackrel{\mathrm{def}}{=} \{\, a \in \Sigma_L : \exists l \in \chi.l(\alpha) = a \,\}$$

is directed (consistent) in $\langle\Sigma_L, \leqslant_L\rangle$ if $\mathrm{lab}_\chi(\alpha) \neq \emptyset$. Define l_χ : Seq $\rightharpoonup \Sigma_L$ by

$$l_\chi(\alpha) \stackrel{\mathrm{def}}{=} \begin{cases} \bigsqcup \mathrm{lab}_\chi(\alpha) & \text{if } \mathrm{lab}_\chi(\alpha) \neq \emptyset \\ \uparrow & \text{else.} \end{cases}$$

The above definition is sound by Lemma 2.2.3.2. CLAIM: $l_\chi \in \mathcal{L}$. This is proved by showing that the two conditions in Remark 2.2.3.7 are satisfied. Suppose for $\alpha \in$ Seq, $l_\chi(\alpha)\downarrow$, then $\exists x \in \chi.x(\alpha)\downarrow$. Since T_x is a tree, so $\forall\beta \leqslant \alpha.x(\beta)\downarrow$ which implies $l_\chi(\beta)\downarrow$. Similarly for $l_\chi(\alpha * \langle n+1\rangle)\downarrow \Rightarrow l_\chi(\alpha * \langle n\rangle)\downarrow$. Whence, T_{l_χ} is a tree. Suppose $l_\chi(\alpha)\downarrow$ and α is not a terminal node of l_χ, *i.e.* $\exists\beta > \alpha.l_\chi(\beta)\downarrow$. By definition of l_χ, $\exists x \in \chi.x(\beta)\downarrow$. Since T_x is a tree, $x(\alpha) = \lambda\vec{x}.y$ which is maximal in $\langle\Sigma_L, \leqslant_L\rangle$, and so, $l_\chi(\alpha) = x(\alpha) = \lambda\vec{x}.y$. Hence, $l_\chi \in \mathcal{L}$.

Next, we show that $\chi \subseteq_L l_\chi$. For any $x \in \chi$, it remains to show that conditions (2) and (3) in Definition 2.2.3.3(iii) are satisfied. Observe that for $\alpha \in$ Seq,

$$x(\alpha)\downarrow \Rightarrow [x(\alpha) \in \mathrm{lab}_\chi(\alpha)] \Rightarrow [x(\alpha) \leqslant_L l_\chi(\alpha)];$$

whence condition (2) is satisfied. For condition (3), note that if $l_\chi(\alpha)\downarrow$ and $x(\alpha)\uparrow$, then for χ directed, $\exists y \in \chi.y(\alpha)\downarrow$ by definition of lab_χ. For the consistent case, we take y to be the upper bound in the following (of course y may not necessarily belong to χ which is immaterial to the proof; but we clearly have $x \subseteq_L y$ and $y(\alpha)\downarrow$). In both cases, $\forall \beta < \alpha.y(\beta)$ is maximal in $\langle \Sigma_L, \leqslant_L \rangle$, hence $y(\beta) = \bigsqcup \mathrm{lab}_\chi(\beta) = l_\chi(\beta)$. Now, suppose for a contradiction, $\forall \beta < \alpha.x(\beta)\downarrow \Rightarrow [x(\beta) \not\leqslant_L y(\beta)]$, i.e. $x(\beta) = y(\beta)$ because by definition of l_χ, we know that $x(\beta) \leqslant_L y(\beta)$ necessarily. For the consistent case, $x \not\subseteq_L y$ which contradicts the upper bound supposition; for the directed case, notice that x and y are not $\subseteq_L$-compatible. For if there exists z such that $x \subseteq_L z \& y \subseteq_L z$. Then, applying condition (2) to $y \subseteq_L z$, we have $z(\alpha)\downarrow$. Now apply (3) to x and z at α, we arrive at a contradiction because $\forall \beta < \alpha.x(\beta)\downarrow \Rightarrow x(\beta)$ is $\leqslant_L$-maximal. Hence, $\exists \beta < \alpha.x(\beta) <_L y(\beta) = l_\chi(\beta)$. Therefore, $\chi \subseteq_L l_\chi$. Clearly, $l_\chi \subseteq_L \bigsqcup \chi$.

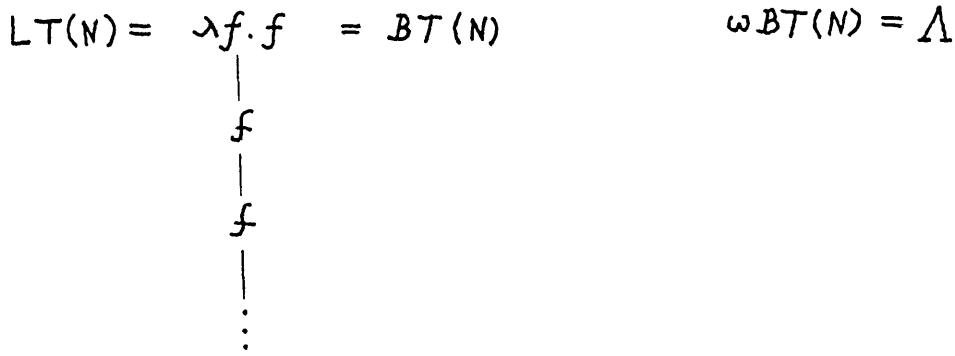
The compact elements of $\mathcal{L}$ are precisely those Longo trees with finite depth and which do not have $\top$ among its set of labels; and it is clear that each Longo tree is the lub of its compact approximants. □

REMARK **2.2.3.9** Note that $\langle \Lambda\mathcal{L}, \subseteq_L \rangle$ is *not* closed under ω-increasing chains. To see this we claim the following:
CLAIM: *Let $L \in \mathcal{L}$. Then,*

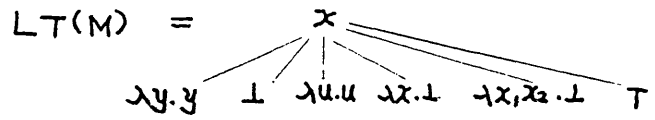
$$L \in \Lambda\mathcal{L} \iff \mathsf{FV}(L) \text{ is finite and } L \text{ is r.e.}$$

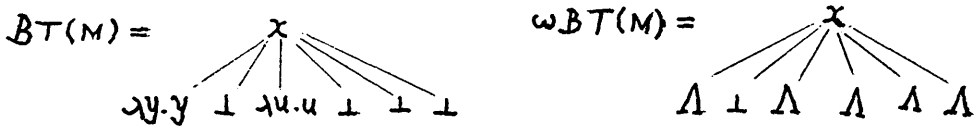
The proof of the claim is similar to that of [Bar84, Theorem 10.1.23]. Consider the following chain: $\langle l_n : n \in \omega \rangle$ where $l_n \stackrel{\mathrm{def}}{=} \mathsf{LT}(x_1(x_2(\cdots(x_n\Omega)\cdots)))$ where the x_i's are all distinct. Clearly, $\bigsqcup\{ l_n : n \in \omega \} \in \mathcal{L}$ but does not belong to $\Lambda\mathcal{L}$ according to the Claim.

EXAMPLE **2.2.3.10** (i) $N \equiv \mathrm{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.



(ii) $M \equiv x(\lambda y.y)\Omega\lambda u.u(\lambda x.\Omega)(\lambda x_1 x_2.\Omega)(\mathbf{YK})$.





(iii) For $n \in \omega$ define $\Omega_n \stackrel{\text{def}}{=} \lambda x_1 \cdots x_n.\Omega$ and $\Omega_\infty \stackrel{\text{def}}{=} \mathbf{YK}$. Then,
$\mathsf{wBT}(\Omega_0) = \bot$; $\forall n.0 < n \in \omega.\mathsf{wBT}(\Omega_n) = \Lambda$;
$\forall n \in \omega + 1.\mathsf{BT}(\Omega_n) = \bot$;
$\forall n \in \omega.\mathsf{LT}(\Omega_n) = \lambda x_1 \cdots x_n.\bot$; $\mathsf{LT}(\Omega_\infty) = \top$.

REMARK **2.2.3.11** In (i) and (ii) of the previous example, we see infinite behaviour exhibited in two different "dimensions" which we unimaginatively dub vertical and horizontal respectively, for want of more appropriate descriptions.

- *vertical dimension*: An infinite object in the vertical dimension is just a tree of vertical depth. In (i), the BT and LT of Y are the same tree which has infinite depth.
- *horizontal dimension*: The infinite object in the horizontal dimension bears the label "⊤". It is the class of unsolvable terms of infinite order. What we describe as the horizontal dimension is just the universe inhabited by the class of unsolvable terms, ordered according to their uniquely defined orders of unsolvability. Observe that only Longo trees have adequate discriminatory power to characterize unsolvable terms completely according to their orders.

There is a more mathematically attuned argument for differentiating between horizontal and vertical infinite behaviours as we have just done. Notice that non-compact elements in the Scott domain of Longo trees are precisely the union (not disjoint) of two subclasses of Longo trees: namely, those that have infinite depth; and those that have "⊤" among its set of labels.

TABLE **2.2.3.12** The following table summarizes and contrasts the respective discriminatory power of wBT, LT and BT.

Discriminatory Power of Trees				
Syntax Class	wBT	LT	BT	Solvability
$\mathbf{PO}_0$-terms	$\bot$			
Abstractions	Λ	$\lambda\vec{x}.\bot$ $\top$	$\bot$	Unsolvables
		$\lambda\vec{x}.y$ $/\cdot\cdot\backslash$		Solvables
λ-free hnfs	x $/\cdot\cdot\backslash$			

2.3 Some Properties of Trees

2.3.1 Longo Tree Preorder is a Precongruence

The main result in this subsection is that the preorder induced by Longo trees is a *precongruence* (or *substitutive* preorder). More precisely,

THEOREM **2.3.1.1** *Let* $M, N \in \Lambda$. *Then,* $M \sqsubseteq_L N \Rightarrow \forall C[\,] \in \Lambda.C[M] \sqsubseteq_L C[N]$.

This result is, as far as I know, new. The proof presented in this section is syntactic in nature and follows that of the same result for Böhm trees as presented in [Bar84, §14.3]. The main novel element in the proof is the introduction of "diagonal" approximants of (Longo) trees and terms, *i.e.* approximations parametrized not just *depth-wise* as is well-known in the case of Böhm trees, but also "breadth-wise" respecting the preorder between the unsolvables.

DEFINITION **2.3.1.2** Let $B \in \mathcal{WB}, \mathcal{B}$ or $\mathcal{L}$. For $k \geqslant 1$, define

$$B^k(\alpha) \stackrel{\text{def}}{=} \begin{cases} B(\alpha) & \text{if } \mathsf{lh}(\alpha) < k, \\ \bot & \text{if } B(\alpha)\downarrow \,\&\, \mathsf{lh}(\alpha) = k, \\ \uparrow & \text{if } B(\alpha)\uparrow \text{ or } \mathsf{lh}(\alpha) > k. \end{cases}$$

REMARK **2.3.1.3** By convention, we say that the root node of a tree is of depth 0. Then, B^k is just the tree B truncated at all depths greater than k; all nodes of B which are defined at depth k now have label $\bot$. Note that this definition differs slightly form the one in [Bar84]. We clearly have $\forall k \geqslant 1.\mathsf{LT}^k(M) \subseteq_L \mathsf{LT}(M)$. Similarly for wBTand BT.

PROPOSITION **2.3.1.4** *(i)* $\forall M \in \Lambda.\mathsf{LT}(M) = \bigsqcup_{k\in\omega} \mathsf{LT}^k(M)$.
(ii) $\mathsf{LT}(M) \subseteq_L \mathsf{LT}(N) \iff \forall k \in \omega.\mathsf{LT}^k(M) \subseteq_L \mathsf{LT}^k(N)$.
Similarly for wBT *and* BT.

PROOF Straightforward. □

DEFINITION **2.3.1.5** (i) $\Lambda\bot$, the set of $\lambda\bot$-terms, is defined by the following grammar:

$$\Lambda\bot \ni M ::= \bot \mid x \mid (MN) \mid (\lambda x.M).$$

(ii) Define the notion of reduction $(\beta\bot)$ on $\Lambda\bot$ as the union of (β) and $(\bot)$; the latter is simply $\bot P \to \bot$.

(iii) Let $A \in \Lambda\mathcal{L}$ be finite. By induction on $\mathsf{d}(A) \stackrel{\text{def}}{=} \max\{\mathsf{lh}(\alpha) : \alpha \in A\}$, a λ-term $L(A)$ is defined which has A as its Longo tree.

$$\begin{array}{lll} A = \lambda x_1 \cdots x_n.\bot, n \geqslant 0 & \Rightarrow & L(A) = \lambda x_1 \cdots x_n.\mathbf{\Omega}, \\ A = \top & \Rightarrow & L(A) = \mathbf{YK}, \\ A = \lambda\vec{x}.y & \Rightarrow & L(A) = \lambda\vec{x}.y, \\ A = \begin{array}{c} \lambda\vec{x}.y \\ / \ \backslash \\ A_1 \cdots A_n \end{array} & \Rightarrow & L(A) = \lambda\vec{x}.yL(A_1)\cdots L(A_n). \end{array}$$

(iii) Let A be a finite Longo tree. Then, $L[A] \in \Lambda\bot$ is obtained from $L(A)$ by replacing every occurrence of $\mathbf{\Omega}$ by $\bot$.

(iv) Let $M \in \Lambda$. Define $M^{[n]} \stackrel{\text{def}}{=} L[\mathsf{LT}^n(M)] \in \Lambda\bot$.

FACT **2.3.1.6** *(i)* $\forall$ *finite* $A \in \Lambda\mathcal{L}.\mathsf{LT}(L(A)) = A$.
(ii) $\mathsf{LT}(M^{[n]}) = \mathsf{LT}^n(M)$. □

The following terminology originates from [Wad71] and is slightly different from that in §14.3 of [Bar84].

DEFINITION **2.3.1.7** Let $M \in \Lambda\bot$.

(i) $P \in \Lambda\bot$ is an *approximate normal form (anf)* of M iff $P \trianglelefteq_L M$ and P is a $\beta\bot$-nf; and we define $\mathcal{A}(M) \stackrel{\text{def}}{=} \{P \in \Lambda\bot : P \text{ is an anf of } M\}$.

(ii) $M \in \Lambda\bot$. Then, $\alpha(M) \in \Lambda\bot$ is obtained from M by replacing the outermost ($\subseteq$-maximal) redexes by $\bot$. Define $\alpha(M)$ inductively as follows:

$$\begin{aligned}
\alpha(\lambda\vec{x}.yM_1\cdots M_m) &= \lambda\vec{x}.y\alpha(M_1)\cdots\alpha(M_m) \quad m \geqslant 0\\
\alpha(\lambda\vec{x}.(\lambda y.P)QM_1\cdots M_m) &= \alpha(\lambda\vec{x}.\bot M_1\cdots M_m)\\
&= \lambda\vec{x}.\bot\alpha(M_1)\cdots\alpha(M_m).
\end{aligned}$$

(iii) $\omega(M)$ is the $\bot$-nf of $\alpha(M)$, defined inductively as follows:

$$\begin{aligned}
\omega(\lambda\vec{x}.yM_1\cdots M_m) &= \lambda\vec{x}.y\omega(M_1)\cdots\omega(M_m) \quad m \geqslant 0\\
\omega(\lambda\vec{x}.(\lambda y.P)QM_1\cdots M_m) &= \omega(\lambda\vec{x}.\bot M_1\cdots M_m)\\
&= \lambda\vec{x}.\bot.
\end{aligned}$$

(iv) $\mathcal{A}'(M) \stackrel{\text{def}}{=} \{\omega(N) \mid M \twoheadrightarrow_\beta N\}$.

LEMMA **2.3.1.8** *Let $M \in \Lambda\bot$. Then*

(i) $\omega(M) \sqsubseteq_L M$.

(ii) $M \twoheadrightarrow_\beta N \Rightarrow \omega(M) \sqsubseteq_L \omega(N)$.

PROOF (i) By induction on the length of M. Let $M = \lambda\vec{x}.(\lambda y.P)QM_1\cdots M_m$. Then $\omega(M) = \lambda\vec{x}.\bot$; clearly, $\omega(M) \sqsubseteq_L M$. Let $M = \lambda\vec{x}.yM_1\cdots M_m$, then $\omega(M) = \lambda\vec{x}.y\omega(M_1)\cdots\omega(M_m) \sqsubseteq_L M$, by the induction hypothesis.
(ii) By induction on the structure of M, we have $M \twoheadrightarrow_\beta N \Rightarrow \omega(M) \sqsubseteq_L \omega(N)$. Result then follows from the transitivity of $\sqsubseteq_L$. □

DEFINITION **2.3.1.9** Let $M \in \Lambda\bot$ and $\chi \subseteq \Lambda\bot$.

(i) $\bigsqcup\chi = M$ if $\bigsqcup\{\mathsf{LT}(P) : P \in \chi\} = \mathsf{LT}(M)$.

(ii) χ is *directed* if $\{\mathsf{LT}(P) : P \in \chi\}$ is directed in $\langle\Lambda\mathcal{L}, \subseteq_L\rangle$.

PROPOSITION **2.3.1.10** *(i) $\mathcal{A}(M)$ is directed.*

(ii) $\mathcal{A}'(M)$ is directed.

PROOF

(i) By definition, we want to show $\{\mathsf{LT}(N) : N \in \beta\bot\text{–nf} \ \&\ N \sqsubseteq_L M\}$ is directed in $\langle\mathcal{L}, \subseteq_L\rangle$. Let $P, P' \in \mathcal{A}(M)$ such that $\mathsf{LT}(P)$ and $\mathsf{LT}(P')$ are consistent. Then $l \stackrel{\text{def}}{=} \mathsf{LT}(P) \sqcup \mathsf{LT}(P')$ is well-defined and belongs to $\Lambda\mathcal{L}$ because of the Claim in Remark 2.2.3.9. It is not difficult to see that $L[l] \in \mathcal{A}(M)$.

(ii) By the Church-Rosser property of (β) and Lemma 2.3.1.8.

□

The following "diagonal approximation" is needed because in Longo trees, two "dimensions" of infinite behaviour are exhibited.

DEFINITION **2.3.1.11** Let $M \in \Lambda\bot, l \in \Lambda\mathcal{L}$

(i) The *n-th diagonal approximant* of l, $l^{\langle n\rangle} \in \Lambda\mathcal{L}$ is defined as follows:

$$l^{\langle 0\rangle} \stackrel{\text{def}}{=} \bot;\ \text{singleton tree}$$

$$l^{\langle n+1\rangle}(\sigma) \stackrel{\text{def}}{=} \begin{cases} \uparrow & \text{if} \quad \mathsf{lh}(\sigma) > n+1 \text{ or } l(\sigma)\uparrow; \\ \bot & \text{if} \quad \mathsf{lh}(\sigma) = n+1 \ \&\ l(\sigma)\downarrow \\ \lambda\vec{x}.y & \text{else if } l(\sigma) = \lambda\vec{x}.y \\ \lambda x_1 \cdots x_N.\bot & \text{if } l(\sigma) = \lambda x_1 \cdots x_m.\bot \\ \text{s.t. } N = \min(m, n+1) & \text{or } \top. \end{cases}$$

(ii) The *n-th diagonal approximant* of M, $M^{\langle n\rangle} \in \Lambda\bot$ is defined inductively as follows:

$$M^{\langle 0\rangle} \stackrel{\text{def}}{=} \bot;$$

$$M^{\langle n+1\rangle} \stackrel{\text{def}}{=} \begin{cases} \lambda\vec{x}.yM_0^{\langle n\rangle} \cdots M_{m-1}^{\langle n\rangle} & \text{if } M \text{ has phnf } \lambda\vec{x}.y\vec{M}, \\ \lambda x_1 \cdots x_N.\bot & \text{if } M \text{ is unsolvable of order} \\ & m \in \omega+1 \ \&\ N = \min(m, n+1). \end{cases}$$

LEMMA **2.3.1.12** *Let $M \in \Lambda\bot$. Then,*

(i) $\forall n \geqslant 0.\mathsf{LT}(M^{\langle n\rangle}) = \mathsf{LT}^{\langle n\rangle}(M)$.

(ii) $\forall M \in \Lambda.\mathsf{LT}(M) = \bigsqcup_{n\in\omega} \mathsf{LT}^{\langle n\rangle}(M)$.

PROOF Straightforward induction on $\mathsf{d}(\mathsf{LT}(M))$. □

REMARK **2.3.1.13** For each $n \in \omega$, $M^{[n]} \sqsubseteq_L M$. In fact, $M^{[n]}$ has Longo tree which is identical to the Longo tree of M truncated after depth n, *i.e.* $\mathsf{LT}(M^{[n]}) = \mathsf{LT}^n(M)$. However, $M^{[n]}$ does not necessarily belong to $\mathcal{A}(M)$. For example, take $M \equiv \mathbf{YK}$. Then $M^{[1]} = \mathbf{YK}$ which is not a $\beta\bot$-nf; hence the need to introduce $M^{\langle n\rangle}$.

FACT **2.3.1.14** (i) $\forall n \in \omega.M^{\langle n\rangle} \sqsubseteq_L M$ and $M^{\langle n\rangle}$ is in $\beta\bot$-nf; hence,

$$\forall n \in \omega. M^{\langle n \rangle} \in \mathcal{A}(M).$$

(2) From Lemma 2.3.1.12, we deduce that $M = \bigsqcup M^{\langle n \rangle}$. Hence, $M = \bigsqcup \mathcal{A}(M)$.

PROPOSITION **2.3.1.15** *(i)* $\mathcal{A}'(M) \subseteq \mathcal{A}(M)$.
(ii) $M = \bigsqcup \mathcal{A}(M) = \bigsqcup \mathcal{A}'(M)$.

PROOF (i) Let $\omega(N) \in \mathcal{A}'$ with $M \twoheadrightarrow_\beta N$. Then $\mathsf{LT}(M) = \mathsf{LT}(N)$, hence $\omega(N) \trianglelefteq_L M$ by Lemma 2.3.1.8. Clearly, $\omega(N)$ is a $\beta\bot$-nf, so $\omega(N) \in \mathcal{A}(M)$.
(ii) CLAIM: $\forall P \in \mathcal{A}(M). \exists P' \in \mathcal{A}'(M). P \trianglelefteq_L P'$. Then, $M = \bigsqcup \mathcal{A}(M) \trianglelefteq_L \bigsqcup \mathcal{A}'(M) \trianglelefteq M$ *i.e.* $M = \bigsqcup \mathcal{A}'(M)$.
PROOF OF CLAIM: Define

$$g(M,0) \stackrel{\text{def}}{=} M;$$

$$g(M,n+1) \stackrel{\text{def}}{=} \begin{cases} \lambda \vec{x}.yg(M_0,n)\cdots g(M_{m-1},n) & \text{if } M \text{ has phnf } \lambda\vec{x}.y\vec{M}, \\ \lambda x_1 \cdots x_N.M' & \text{if } M \text{ is unsolvable of} \\ & \text{order } p;\ N = \min(p, n+1) \\ & \& \ M \twoheadrightarrow_{\mathrm{h}} \lambda x_1 \cdots x_N.M'. \end{cases}$$

Recall that corresponding to the two cases in the preceding definition, $M \twoheadrightarrow_{\mathrm{h}} \lambda\vec{x}.y\vec{M}$ and $M \twoheadrightarrow_h \lambda\vec{x}.M'$ where $\twoheadrightarrow_{\mathrm{h}}$ is the *head* reduction.

Hence, we have $\forall n. M \twoheadrightarrow_\beta g(M,n)$. Also, $\forall n. M^{\langle n \rangle} \trianglelefteq_L \omega(g(M,n))$. Finally, observe that $\forall P \in \mathcal{A}(M). \exists n \in \omega. P \trianglelefteq_L M^{\langle n \rangle}$; then take $P' \equiv \omega(g(M,n))$. □

COROLLARY **2.3.1.16** *Let* $M, N \in \Lambda\bot$. *Then*

$$M \trianglelefteq_L N \iff \forall M' : M \twoheadrightarrow_\beta M'. \exists N' : N \twoheadrightarrow_\beta N'. \omega(M') \trianglelefteq_L \omega(N').$$

PROOF "$\Leftarrow$": Since $M = \bigsqcup\{\omega(M') : M \twoheadrightarrow_\beta M'\}$, result then follows from the Proposition.
"$\Rightarrow$": Let $M \twoheadrightarrow_\beta M'$. Then, by Lemma 2.3.1.8,

$$\omega(M') \trianglelefteq_L M' \approx_L M \trianglelefteq_L \bigsqcup\{\omega(N') : N \twoheadrightarrow_\beta N'\},$$

i.e. $\mathsf{LT}(\omega(M')) \subseteq_L \bigsqcup\{\mathsf{LT}(\omega(N')) : N \twoheadrightarrow_\beta N'\}_1$. Since $\mathsf{LT}(\omega(M'))$ is a finite element of $\langle \mathcal{L}, \subseteq_L \rangle$ and $\{\cdots\}_1$ is directed by Proposition 2.3.1.10,

$$\exists N'. N \twoheadrightarrow_\beta N'. \omega(M') \trianglelefteq_L \omega(N').$$

□

LEMMA **2.3.1.17** *Let $M \in \Lambda\bot$ and $C[\,] \in \Lambda\bot$. Then, $C[\bot] \sqsubseteq_L C[M]$.*

PROOF The proof uses Corollary 2.3.1.16. Let $C[\bot] \twoheadrightarrow_\beta L$. Then, $C[M] \twoheadrightarrow_\beta L[\bot := M]$. It suffices to prove $\omega(L) \sqsubseteq_L \omega(L[\bot := M])$. By induction on the length of the term L. Then, three cases are as follows:

(1) $L \equiv \lambda\vec{x}.y\vec{L}$. Suppose by hypothesis, $\omega(L_i) \sqsubseteq_L \omega(L_i[\bot := M])$. Then,

$$\begin{aligned}\omega(L) &= \lambda\vec{x}.y\omega(L_0)\cdots\omega(L_{m-1}) \\ &\sqsubseteq_L \lambda\vec{x}.y\omega(L_0[\bot := M])\cdots\omega(L_{m-1}[\bot := M]) \\ &= \omega(\lambda\vec{x}.y\vec{L}[\bot := M]).\end{aligned}$$

(2) $L \equiv \lambda\vec{x}.(\lambda y.P)Q\vec{L}$, $\omega(L) = \lambda\vec{x}.\bot \sqsubseteq_L \lambda\vec{x}.M'$. Hence, $\omega(L) \sqsubseteq_L \omega(L[\bot := M])$.

(3) $L \equiv \lambda\vec{x}.\bot\vec{L}$. Similar to above.

□

COROLLARY **2.3.1.18** $P \in \mathcal{A}(M) \Rightarrow C[P] \sqsubseteq_L C[M]$.

PROOF P is a $\beta\bot$-nf by assumption. Hence,

$$\exists D \in \Lambda, \exists 0 \leqslant n_i \in \omega.P \equiv D[\lambda x_1 \cdots x_{n_1}.\bot, \cdots, \lambda x_1 \cdots x_{n_m}.\bot] \quad \&$$

$$M \equiv D[M_1, \ldots, M_m];$$

for appropriate M_i. There are, in general, more than one choice for such $D[\,]$ and M_i's. We define a canonical choice for $D[\,]$ as follows. Choose $D[\,]$ and the corresponding M_i's satisfying the above such that:

(1) If M_i is solvable, then $n_i = 0$;

(2) If M_i is unsolvable of order $p \neq \infty$, then, $n_i = 0$;

Now, define a new context $D' \equiv D[\lambda x_1 \cdots x_{n_1}.[\,], \cdots, \lambda x_1 \cdots x_{n_m}.[\,]]$. We note that in the light of our choice of $D[\,]$ subject to stipulations mentioned in (1) and (2), $n_i > 0$ iff $M_i \in \mathbf{PO}_\infty$. Then, $P \equiv D'[\bot, \cdots, \bot]$ and $\lambda\beta \vdash M = D'[M'_1, \cdots, M'_m]$. Hence,

$$C[P] \equiv C[D'[\bot, \cdots, \bot]] \sqsubseteq_L C[D'[M'_1, \cdots, M'_m]] \approx_L C[M],$$

by repeated applications of Lemma 2.3.1.17. □

LEMMA **2.3.1.19** *(i)* $P \twoheadrightarrow_{\perp} Q \Rightarrow \omega(P) \equiv \omega(Q)$.
(ii) $P \rightarrow_{\perp} Q \Rightarrow P \approx_L Q$.

PROOF

(i) Suffices to show $P \rightarrow_{\perp} Q \Rightarrow \omega(P) \equiv \omega(Q)$ which is straightforward.

(ii) Since for $P \in \Lambda\perp.\mathsf{LT}(P) \stackrel{\text{def}}{=} \mathsf{LT}(P[\perp := \Omega])$. Observe that $\Omega\vec{M} \in \mathbf{PO}_0$. Hence, $P \approx_L Q$.

□

COROLLARY **2.3.1.20** *Let* $P, Q \in \Lambda\perp$. *Then,*

$$P =_{\beta\perp} Q \Rightarrow P \approx_L Q.$$

PROOF By virtue of $P =_{\beta} Q$ or $P =_{\perp} Q$ implying $P \approx_L Q$. □

The following notion is due to Welch and the two results that follow the definition are extracted from [Bar84] §14.3.17 and §14.3.8 respectively.

DEFINITION **2.3.1.21** (i) Let $M \in \Lambda$ and $F \subset M$ (meaning that F is a set of subterms of M). Then, M reduces to N *without touching* F, denoted $M \twoheadrightarrow_{\neg F} N$, if there exists a reduction $\sigma : M \twoheadrightarrow N$ such that a residual (see [Bar84] for definition) of an element of F is never contracted.

(ii) If $P \subset M$, write $M \twoheadrightarrow_{\neg P} N$ if $M \twoheadrightarrow_{\neg F_P} N$ where F_P is the set of all redexes $\subset P$.

LEMMA **2.3.1.22** *Let* $M \twoheadrightarrow_{\neg F} N$ *by a reduction* σ. *Then* $M[F := \perp] \twoheadrightarrow N[F' := \perp]$ *where* $[F := \perp]$ *denotes the replacement of the* [$\subset$ *-maximal*] *redexes in* F *by* $\perp$ *and* F' *is the set of residuals of* F *w.r.t.* σ.

PROOF See [Bar84, pp370]. □

PROPOSITION **2.3.1.23 (Welch)** *Let* $C[M], N \in \Lambda$. *Suppose* $C[M] \twoheadrightarrow N$. *Then,* $\exists M_1, N_1 : M \twoheadrightarrow M_1, N \twoheadrightarrow N_1 . C[M_1] \twoheadrightarrow_{\neg M_1} N_1$.

PROOF See [Bar84, pp370]. □

The following result and its proof follow [Bar84, Proposition 14.3.19], though $\mathcal{A}(M)$,$\mathcal{A}'(M)$ and $\leq_L$ have been given somewhat different meanings.

PROPOSITION **2.3.1.24** *Let* $C[M] \in \Lambda$. *Then,*

$$\forall P \in \mathcal{A}'(C[M]).\exists Q \in \mathcal{A}'(M).P \leq_L C[Q].$$

PROOF Let $P \in \mathcal{A}'(C[M])$. Then, $P = \omega(N)$ such that $C[M] \twoheadrightarrow N$. By Proposition 2.3.1.23, for some $M_1, N_1 \in \Lambda$

(1) $M \twoheadrightarrow M_1, N \twoheadrightarrow N_1$,

(2) $C[M_1] \twoheadrightarrow_{\neg M_1} N_1$,

(3) Take $Q \equiv \omega(M_1)$. Then, $Q \in \mathcal{A}'(M)$. It remains to show $P \leq_L C[Q]$. By (2) and Lemma 2.3.1.22,

$$C[M_1][F := \bot] \twoheadrightarrow N_1[F' := \bot],$$

where F is the set of all redex occurrences in M_1.

Now,

$P \equiv \omega(N)$	$\leq_L$	$\omega(N_1)$	(1), Lemma 2.3.1.8
	$\equiv$	$\omega(N_1[F' := \bot])$	since all redexes have to be replaced by $\bot$ anyway
	$\leq_L$	$N_1[F' := \bot]$	Lemma 2.3.1.8
	$\approx_L$	$C[M_1][F := \bot]$	(3),Corollary 2.3.1.20
	$\equiv$	$C[\alpha(M_1)]$	
	$\approx_L$	$C[\omega(M_1)]$	Corollary 2.3.1.20
	$\equiv$	$C[Q]$.	

□

COROLLARY **2.3.1.25 (Proof of Theorem 2.3.1.1)** *Let* $C[\,] \in \Lambda$ *and* $M, N \in \Lambda$. *Then,*

(i) $C[M] = \bigsqcup\{ C[Q] : Q \in \mathcal{A}'(M) \} = \bigsqcup\{ C[Q] : Q \in \mathcal{A}(M) \}$.

(ii) $C[M] = \bigsqcup_n C[M^{(n)}]$.

(iii) $M \leq_L N \Rightarrow C[M] \leq_L C[N]$.

PROOF (i) By Corollary 2.3.1.18, $\forall Q \in \mathcal{A}(M).C[Q] \sqsubseteq_L C[M]$. Then,

$$\bigsqcup\{ C[Q] : Q \in \mathcal{A}'(M) \} \sqsubseteq_L \bigsqcup\{ C[Q] : Q \in \mathcal{A}(M) \} \sqsubseteq_L C[M]$$

because $\mathcal{A}'(M) \subseteq \mathcal{A}(M)$. Since $C[M] = \bigsqcup\{ P : P \in \mathcal{A}'(C[M]) \}$, by Proposition 2.3.1.15. The rest follows from the Proposition.
(ii) Since $M^{(n)} \in \mathcal{A}(M)$ and $\forall P \in \mathcal{A}(M).\exists p.P \sqsubseteq_L M^{(p)}$. Now use (i).
(iii) Assume $M \sqsubseteq_L N$. Then,

$$\begin{aligned} C[M] &= \bigsqcup\{ C[P] : P \in \mathcal{A}(M) \} \text{ by (i)} \\ &\sqsubseteq_L C[N]. \end{aligned}$$

since $P \sqsubseteq_L M \Rightarrow P \sqsubseteq_L N$ and Corollary 2.3.1.18. □

2.3.2 A Non Full Abstraction Result

Longo tree semantics compares terms according to their intrinsic syntactic structure in keeping with operational considerations in the lazy regime such as:

- regarding the strongly unsolvables as least terms,
- comparing abstraction terms according to the number of nested abstractions they can yield.

This furnishes a tool enabling fine distinctions to be made between λ-terms. Longo tree preorder is substitutive, and ordered by $\subseteq_L$, the Longo-like trees form a Scott domain. The Longo tree of a term may be regarded as its *denotation.*

Weak Böhm trees capture the *operational* essence of lazy evaluation — normal order reduction terminating at whnf (= abstractions). Assuming that convergence to abstractions or λ-free hnfs are the only computational "observables", it is reasonable not to distinguish between terms that are *operationally equivalent*, *i.e.* between terms M, N such that

$$\forall C[\,] \in \Lambda.C[M] \sqsubseteq_w C[N] \;\&\; C[N] \sqsubseteq_w C[M].$$

Is the weak Böhm tree semantics *fully abstract* with respect to the Longo tree semantics? — regarding the former as inducing a notion of operational equivalence and the latter as yielding denotations of terms. The answer is no.

PROPOSITION **2.3.2.1 (Non Full Abstraction)** *Let* $M, N \in \Lambda$,

$$M \sqsubseteq_L N \Rightarrow \forall C[\,] \in \Lambda^{\circ}.C[M] \sqsubseteq_w C[N].$$

The converse is false.

PROOF "$\Rightarrow$" follows from the precongruence of $\precsim_L$ and that $\precsim_L \subseteq \precsim_w$. For "$\not\Leftarrow$", consider closed terms $M \equiv \lambda x.x\Omega$ and $N \equiv \lambda x.xx$. Let $C[\,]$ range over closed contexts. Observe that the assertion

$$\forall C[\,] \in \Lambda^o.C[M] \precsim_w C[N]$$

is equivalent to

$$\forall C[\,] \in \Lambda^o.C[M]\Downarrow \Rightarrow C[N]\Downarrow.$$

By Proposition 4.1.3.5 (see Chapter 4), it follows that the preceding assertion is equivalent to $M \precsim^B N$ which is easy to verify. □

2.4 Lazy Lambda Theories

2.4.1 Preliminary Definitions

Let $\mathcal{T}$ be a set of closed equations of λ-terms. Define $\mathsf{Th}(\mathcal{T})$ to be the set of closed equations provable in $\lambda\beta \cup \mathcal{T}$. We say $\mathcal{T}$ is a *λ-theory* if $\mathcal{T}$ is *consistent* and $\mathcal{T} = \mathsf{Th}(\mathcal{T})$. A λ-theory is *inconsistent* if all λ-terms can be proved equal. $\mathcal{A}$, a set of closed equations, is an *axiomatization* of the λ-theory $\mathcal{T}$ if $\mathsf{Th}(\mathcal{A}) = \mathcal{T}$. A λ-theory $\mathcal{T}$ is *r.e.* if after coding $\mathcal{T}$ is a r.e. set of integers.

An equational theory $\mathcal{T}$ is *Hilbert-Post (HP) complete* if for every equation $M = N$ in the language of $\mathcal{T}$, either $\mathcal{T} \vdash M = N$ or $\mathsf{Th}(\mathcal{T} \cup \{M = N\})$ is inconsistent. In other words, a HP-complete theory is a *maximal consistent* theory.

DEFINITION **2.4.1.1** Let $\mathcal{C}$ be a class of λ-theories, *e.g.* semi-sensible λ-theories or fully lazy λ-theories which will be defined in the sequel, typically characterized by the non-provability of certain equations. A (consistent) λ-theory $\mathcal{T}$, belonging to the class $\mathcal{C}$ is *HP-complete w.r.t.* $\mathcal{C}$ if for every (closed) equation $M = N$ in the language of $\mathcal{T}$, either $\mathcal{T} \vdash M = N$ or $\mathsf{Th}(\mathcal{T} \cup \{M = N\}) \notin \mathcal{C}$.

DEFINITION **2.4.1.2** Let $\mathcal{K}_o \stackrel{\text{def}}{=} \{M = N : M, N \in \Lambda^o \ \&\ \text{unsolvable}\}$ and $\mathcal{K} \stackrel{\text{def}}{=} \mathsf{Th}(\mathcal{K}_o)$. A λ-theory $\mathcal{T}$ is *sensible* if $\mathcal{K} \subseteq \mathcal{T}$. $\mathcal{T}$ is *semi-sensible* if $\mathcal{T}$ does not equate any solvable term to an unsolvable one.

2.4.2 Lazy Lambda Theories

The theory of sensible λ-theory and sensible λ-model[1] is beautifully presented in Barendregt's book on λ-calculus [Bar84]. We have argued earlier on that sensible λ-calculus corresponds to the theory of "strict" λ-calculus. In this section, we shall develop a theory of *non-sensible* λ-theories (more suggestive and mnemonic names will be revealed later!) which is motivated by computational considerations in the lazy regime.

DEFINITION **2.4.2.1** Define, for $n \in \omega + 1$,

$$\mathcal{PO}_n \stackrel{\text{def}}{=} \{M = N : M, N \in \mathbf{PO}_n\},$$

$$\mathcal{L} \stackrel{\text{def}}{=} \mathsf{Th}(\bigcup_{n \in \omega+1} \mathcal{PO}_n).$$

REMARK **2.4.2.2** Strictly speaking, we should consider only $M \in \mathbf{PO}_m{}^o \stackrel{\text{def}}{=} \mathbf{PO}_m \cap \Lambda^o$ in the preceding definition as it is customary to consider equation between closed terms in defining λ-theories, but the above extension is natural in the light of the *substitutivity* of the $\mathbf{PO}_n$-terms, *i.e.*

$$\forall m \in \omega + 1. M \in \mathbf{PO}_m \Rightarrow \forall P \in \Lambda. M[x := P] \in \mathbf{PO}_m.$$

It is easy to see this by an appeal to the substitutivity of β-conversion and $\mathbf{PO}_0$-terms. For any $M \in \mathbf{PO}_{n+1}$ satisfies:

[1] *Sensible* λ-models are those that give the same denotation to all unsolvable terms, e.g. D_∞, $\mathcal{P}\omega$ and T^ω.

$M =_\beta \lambda x_1 \cdots x_{n+1}.N$ where $N \in \mathbf{PO}_0$.

Then, $M[x := P] =_\beta \lambda x_1 \cdots x_{n+1}.N[x := P]$ and $N[x := P] \in \mathbf{PO}_0$. Similarly, for $\mathbf{PO}_\infty$-terms.

Define the λ-theories induced by Longo trees, Böhm trees and weak Böhm trees by $\mathcal{LT}, \mathcal{BT}$ and $w\mathcal{BT}$ respectively. Of the three, only $\mathcal{LT}, \mathcal{BT}$ are semi-sensible and only $\mathcal{BT}$ is sensible.

We remark that $\mathcal{L}$ is consistent. Since, clearly $\bigcup_{n\in\omega+1} \mathcal{PO}_n \subset \mathcal{K}_o$ which is a corollary of the Classification Lemma of the unsolvables, we conclude that $\mathcal{L} \subseteq \mathcal{K}$. The consistency of $\mathcal{L}$ then follows from that of $\mathcal{K}$. See [Bar84] for proofs of the consistency of $\mathcal{K}$ and $\mathcal{K}\eta$.

DEFINITION **2.4.2.3** (i) Let $\mathcal{T}$ be a λ-theory. $\mathcal{T}$ is *zero sensible* if $\mathsf{Th}(\mathcal{PO}_0) \subseteq \mathcal{T}$ *i.e.* $\mathcal{T}$ equates all the strongly unsolvable terms.

(ii) $\mathcal{T}$ is *finitely sensible* if $\mathcal{T}$ equates all unsolvable λ-terms of *finite* orders.

(iii) $\mathcal{T}$ is *pre-lazy* if $\mathcal{L} \subseteq \mathcal{T}$.

(iv) An *fully lazy λ-theory* $\mathcal{T}$ is a pre-lazy λ-theory which equates any two unsolvable terms iff they have the same order, *i.e.*

$$\forall m, n \in \omega + 1. \forall M \in \mathbf{PO}_m. \forall N \in \mathbf{PO}_n. \mathcal{T} \vdash M = N \iff m = n.$$

LEMMA **2.4.2.4** *Let $\mathcal{T}$ be a zero sensible λ-theory which equates all $\mathbf{PO}_\infty$-terms, then $\mathcal{T}$ is pre-lazy.*

PROOF It remains to show that $\mathcal{T}$ equates any two terms of the same finite order of unsolvability greater than 0. Let $M, N \in \mathbf{PO}_{n+1}$. Then,

$$M =_\beta \lambda x_1 \cdots x_{n+1}.M', \quad N =_\beta \lambda x_1 \cdots x_{n+1}.N';$$

and $M', N' \in \mathbf{PO}_0$. But $M' =_\mathcal{T} N'$, then result follows from the fact that $\mathcal{T}$ is a congruence. □

PROPOSITION **2.4.2.5** *A pre-lazy λ-theory $\mathcal{T}$ is either fully lazy or finitely sensible. Clearly, a sensible λ-theory is trivially finitely sensible.*

PROOF Let Ω_n be the representative $\mathbf{PO}_n$ term for each $n \in \omega + 1$.
I. Suppose $\exists m, n \in \omega$ and $m \neq n$ such that $\Omega_m = \Omega_n$. Then, $\exists k > 0. \Omega_0 = \Omega_k$, since $\forall C \in \Lambda^o. \Omega_{n+1} C = \Omega_n$. From which we can deduce $\forall l. l = ak + b$ where $0 \leqslant b < k$, $\Omega_l = \Omega_b$, since $\lambda x. \Omega_n = \Omega_{n+1}$. Now, because $\Omega_0 = \Omega_0 C = \Omega_k C = \Omega_{k-1}$, we then have $\Omega_0 = \Omega_1 = \cdots = \Omega_k$. Hence, $\forall m, n \in \omega. \Omega_m = \Omega_n$, *i.e.* $\mathcal{T}$ is finitely

sensible.
II. Suppose $\neg[\exists m,n \in \omega, m \neq n.\mathcal{T} \vdash \Omega_m = \Omega_n]$. It remains to show: $\forall n \in \omega.\mathcal{T} \nvdash \Omega_n = \Omega_\infty$, then we are done. Suppose, for a contradiction, that for some $n \geqslant 0.\mathcal{T} \vdash \Omega_n = \Omega_\infty$. In particular, $\mathcal{T} \vdash \mathbf{YK} = \Omega_n$, since $\mathbf{YK} \in \mathbf{PO}_\infty$. But $\mathcal{T} \vdash \mathbf{YK} = \lambda x.\mathbf{YK} = \lambda x.\Omega_n = \Omega_{n+1}$, *i.e.* $\mathcal{T} \vdash \Omega_n = \Omega_{n+1}$, which contradicts the assumption. □

COROLLARY **2.4.2.6** *Let $\mathcal{T}$ be a pre-lazy λ-theory. If $\mathcal{T} \nvdash \Omega_0 = \Omega_1$ then $\mathcal{T}$ is fully lazy.*

PROOF In view of the reasoning in II of the previous proof, we only need to show that $\mathcal{T} \nvdash \Omega_0 = \Omega_1 \Rightarrow \forall n,m \in \omega.n \neq m.\mathcal{T} \nvdash \Omega_n = \Omega_m$ which is clear from I. □

OPEN QUESTION **2.4.2.7** Is every pre-lazy λ-theory which is finitely sensible necessarily sensible?

DISCUSSION **2.4.2.8** The development hitherto in this section is a way of imposing some structure in the universe of unsolvable λ-terms. It may be seen as an approach to "non-sensible" λ-calculus. In fact, we can turn the unsolvables into a preorder-ed set in the following obvious way:

$$\forall m,n \in \omega + 1.\forall M \in \mathbf{PO}_m.\forall N \in \mathbf{PO}_n.M \sqsubseteq N \iff m \leqslant n.$$

However, the associated poset is not a very interesting one, being isomorphic to $\omega + 1$.

Rick Statman [Sta86] has a more interesting way of turning the λ-terms into a poset. The *Statman preorder* $\leqslant$ is defined as follows: for $M,N \in \Lambda$,

$$M \leqslant N \stackrel{\text{def}}{=} \exists \vec{P}.M\vec{P} = N.$$

We read $M \leqslant N$ as "M is more solvable than N". The structure of the induced poset runs counter to the intuitions of classical sensible λ-calculus. For example, the poset has a bottom element consisting of all *solvables* and has all $\mathbf{PO}_\infty$ elements as maximal elements. Statman shows that every countable poset can be embedded into the poset.

OPEN QUESTION **2.4.2.9** Is there an ordered λ-model whose induced preorder on λ-terms coincides with (or extends) $\leqslant$?

Chapter 3

Lazy Lambda Models and the Free Lazy PSE-Model

Synopsis of the Chapter

This Chapter may be divided into two parts. The first part begins with an introduction to the model theory of λ-calculus. Different formulations of *lazy λ-models* based on Abramsky's *quasi-applicative structure with divergence* are then presented and shown equivalent. A general computational adequacy result for a class of continuous lazy λ-models is proved. The second part of this Chapter focuses on a class of λ-models called the *Plotkin-Scott-Engeler (PSE) models.* The basic properties of the subclass of lazy free PSE-models are surveyed. The main result of this Chapter, which confirms and strengthens a conjecture of Longo in [Lon83, Remark 3.9], is a *Local Structure Theorem* for the class of *free lazy PSE-models.*

3.1 Models of the Lambda Calculus

3.1.1 Introduction

Although the theory of the pure, untyped λ-calculus as a formal system was fairly well established in the late 1940's, it took the mathematical community about thirty years to produce the first models which are not of a syntactic nature, usually loosely referred to as "mathematical" models. Before 1970, the only models of the λ-calculus were "term models". They were proved to be *consistent* as a corollary of the well known *Church-Rosser property* of the β-reduction, similarly for $\beta\eta$-reduction, established as long ago as 1936 [CR36]. That it has taken the mathematical community so long to produce a breakthrough in the model theory

of the *pure, untyped* λ-calculus calls for some explanation. There are at least two factors which seemed, at various times, to be formidable obstacles in the search for a mathematical model of $\lambda\beta$:

- **Intrinsic Intensionality of λ-terms**
  It seems fair to suggest that the intended interpretation of λ-terms is that of *algorithms* which can themselves be regarded as data and given as input to other algorithms. Algorithms have significant *intensional* import; whereas set-theoretic functions, which mathematicians at large are better conversant with, are *extensional* entities. Nevertheless, as a first approximation, it is reasonable to identify the two notions. The type free nature of the calculus and the observation that each λ-term plays a *dual operator-operand* role suggest the following "domain equation":

  $$D \cong D^D$$

  where D is the domain of algorithms and D^D its set-theoretic function space. Alas, this is impossible because of Cantor's theorem, if the cardinality of D is greater than 1.

- **Incompatibility of $\lambda\beta$ with (minimal) logic: Curry's version of Russell's Paradox**
  Type free λ-calculus cannot reside harmoniously in a conventional logical framework which gives expression to absurdity. In such a framework, as experienced long ago by Frege [Fre03,Acz80] who had a logical system that essentially incorporated the full type free λ-calculus, Russell's Paradox in the form of a fixed point of negation, can be derived. The crux of the apparent logical incompatibility lies in the *internal definability* of such *self-applying* terms as the fixpoint combinator (*e.g.* $\lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$) in the type free regime. Operations such as application have an inherently *strongly typed* nature. Perhaps it should come as no surprise that such inconsistency result does arise when type constraints are relaxed.

A Brief Survey of Mathematical λ-models

Dana Scott was the first to construct a mathematical model [Sco72]. As alluded to earlier, a long-standing challenge in the search for a model of the λ-calculus is the construction of a structure D and the selection of an appropriate subspace of its function space D^D such that the subspace is isomorphic to or a retract of D in the cases of $\lambda\beta\eta$ or $\lambda\beta$ respectively. Scott solved this problem

ingeniously by constructing a complete lattice which is isomorphic to its *Scott-continuous* function space. More fundamentally, Scott established that complete lattices D together with *continuous embeddings* form a *Cartesian closed category.* Further, any complete lattice D can be embedded in a complete lattice, D_∞ which is isomorphic to its own function space (exponentiation) $[D_\infty \rightarrow D_\infty]$. See [SP82,Plo81] for a general category-theoretic treatment of the solution of domain equation of which the above is an important example; and [GHK*80] for a careful study on *Continuous Lattices*, an area pioneered by Scott's seminal 1972 paper of the same title.

Gordon Plotkin [Plo72] later constructed the *graph model* $\wp\omega$ based on a set-theoretic notion of application. Plotkin's idea turned out to coincide with the classical Myhill-Shepherdson-Rogers definition of application in $\wp\omega$ which was originally introduced to define enumeration operators in computability theory (see *e.g.* [Rog67, page 143]). This model was later rediscovered by Dana Scott [Sco76] who used it as the semantic domain of a λ-calculus based programming language *LAMBDA*. The cleanest and simplest set-theoretic construction of λ-models was given in [Eng81]. This model was later christened Plotkin-Scott-Engeler Algebra (PSE-Algebra) and its theoretical properties further investigated in [Lon83].

In [Plo78], an analog of [Sco76], Plotkin introduced another domain for the study of the semantics of programming languages. Motivated primarily by the view that *lattices*, because of the presence of the semantically arcane top element, are not the ideal semantic domains, this paper is an attempt to develop a mathematical theory of computation á la Scott based on the ω-algebraic cpo T^ω — the Cartesian product of denumerably many copies of T, the 3-element truthvalue cpo — as the universal domain. As a λ-model, the semantic partial order of T^ω has a particularly elegant *local syntactic* characterization: a λ-term is *Böhm-tree* less than another — an operational notion, if and only if its denotation in T^ω is less than that of the other [BL80].

In the early Eighties, research activities in the model theory of the pure, untyped λ-calculus seemed to be focused on a class of rather syntactic structures known as *Filter λ-Models*. Types which are traditionally employed to study the functional properties of λ-calculus (see e.g. [CHS72,CDV81]) are shown to give rise to a class of λ-models.
See e.g. [BCD83,CDHL84,DM86,CDZ87].

3.1.2 Desiderata for Models of $\lambda\beta$

Closely related to the quest for concrete mathematical models of λ-calculus is a surprisingly non-trivial question of a more abstract, ontological nature: what

constitutes a model of the λ-calculus and how to characterize it in a canonical way, *e.g.* category-theoretically. Many researchers[1] have addressed this issue. Their findings highlight various salient semantic characteristics of the models of λ-calculus. As a result, a consensus seems to have arisen regarding the precise definition of what constitutes a model of the Strong Combinatory Logic — λ-*Algebra* and what constitutes a model of the λ-calculus — λ-*Model*.

Before we present in brief the various equivalent characterizations of λ-models, it is instructive to examine the various properties that are expected of mathematical structures that model $\lambda\beta$ and $\lambda\beta\eta$.

- **Applicative Structure**
  $\langle D, \cdot \rangle$: Since the juxtaposition (corresponding to formal application) of any two λ-terms is a λ-term, a structure D with a (total) binary application operator is needed — namely, an applicative structure. Equivalently, associated with each element d of the structure D is a map f from D to itself, *i.e.* an element of the function space, D^D. We shall call the function that maps d to f the Fun map. It should be clear that the binary application and Fun are inter-definable. When no confusion is likely to arise, we will usually omit the symbol "$\cdot$" in the application $d \cdot e$.

- **Combinatory Completeness:**
  An element f of D^D is *representable* if

  $$\exists d \in D. \forall x \in D. f(x) = d \cdot x;$$

  d is called a *representative* of the function f. For each such f, we define

  $$\mathsf{rep}(f) \stackrel{\text{def}}{=} \{ d \in D : \forall x \in D. f(x) = d \cdot x \},$$

  which is known as the *extensionality class* of f. Note that a function does not necessarily have a unique representative. The *representable function space*, $[D \to_{\mathsf{r}} D]$ is that subspace of D^D consisting of representable functions. Similarly, for each $n > 1$, we define the n-place representable function space as

  $$[D^n \to_{\mathsf{r}} D] = \{ f \in D^{D^n} : \exists d \in D. \forall \vec{e}. f(\vec{e}) = d\vec{e} \}.$$

  Each λ-term which is built up from the variables by the rules of application and abstraction may act as an *operator* on λ-terms. This implies

[1] [Sco80b,Mey82,Koy82,Koy84,HL80,BK80,Ber81] among others.

that any polynomial term over D and Var (the denumerable set of variables), say $p(\vec{x}, \underline{\vec{d}})$, when regarded as a function of its free variables must be representable in any structure which is to be a model of $\lambda\beta$.

A function from D^n to D is *algebraic* if there exists a polynomial term $p(\vec{x}, \underline{\vec{d}})$ such that

$$\forall \vec{e} \in D^n . f(\vec{e}) = p(\vec{x}, \underline{\vec{d}})[\vec{x} := \underline{\vec{e}}].$$

Clearly any representable function is algebraic, but the converse is not true. An applicative structure whose algebraic functions coincide with its representable functions is described as *combinatory complete* and only such applicative structures can be admitted as candidates for models of $\lambda\beta$. In fact, we have

FACT **3.1.2.1** *An applicative structure is combinatory complete iff it can be expanded into a* combinatory algebra *i.e. a model of the combinatory logic.*

PROOF Easy exercise. Or see [Bar84, page 90]. □

- **Representable Function Space $[D \to_r D]$ is a retract of D:** A central question in the model theory of $\lambda\beta$ is the *denotation of abstraction terms.* β-axiom enunciates the applicative behaviour of an abstraction $\lambda x.M$ as that of an *algorithm* which when applied to an argument N yields the result $M[x := N]$. Provided we accept an extensional perspective, it seems reasonable to denote $\lambda x.M$ as the *function* $f : d \mapsto [\![M[x := \underline{d}]]\!]$, thereby *eviscerating any intensional contents.* If D is combinatory complete, then the function f (which is algebraic) is representable.

  Let Graph or simply Gr be a map from the representable function space $[D \to_r D]$ to D that selects a unique representative, $\mathsf{Gr}(f) \in \mathsf{rep}(f) \subseteq D$ for each representable function f. Gr plays the role of a *choice* function.

  The satisfaction of *β-axiom implies that $[D \to_r D]$ is a retract of D, i.e.* $\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}_{[D \to_r D]}$. To see this, consider any representable function f and one of its representative(s), d. In the augmented calculus $\Lambda(\underline{D})$, we have, for any $e \in D$,

$$\begin{aligned}
f(e) &= d \cdot e \\
&= [\![\underline{d}e]\!] && \beta - \text{axiom} \\
&= [\![(\lambda x.\underline{d}x)\underline{e}]\!] \\
&= [\![\lambda x.\underline{d}x]\!] \cdot [\![\underline{e}]\!] \\
&= [\![(\lambda x.\underline{d}x)]\!] \cdot e && \text{Fun, ``}\cdot\text{'' inter-definability} \\
&= \mathsf{Fun}([\![(\lambda x.\underline{d}x)]\!])e && \text{meaning of abstraction} \\
&= (\mathsf{Fun} \circ \mathsf{Gr}(c \mapsto d \cdot c))e && d \in \mathsf{rep}(f) \\
&= (\mathsf{Fun} \circ \mathsf{Gr}(f))e.
\end{aligned}$$

Hence, we conclude $\mathsf{Fun} \circ \mathsf{Gr}(f) = f$, *i.e.* $\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}_{[D \to D]}$.

- **Weak Extensionality — axiom (ξ):**[2] In the "extensional" framework in which the meanings of λ-terms are functions instead of algorithms, the denotation of an abstraction $\lambda x.M$ is completely characterized by its *applicative behaviour*

  $$d \mapsto [\![M[x := \underline{d}]]\!].$$

  If this holds, then the structure D is said to be *weakly extensional*; equivalently, if the following axiom is satisfied:

  $$(\xi) \qquad D \models [\forall x(M = N)] \Rightarrow \lambda x.M = \lambda x.N;$$

[2] An applicative structure D is *extensional*, if D satisfies the following (ext) axiom:

$$(\text{ext}) \qquad \forall x, y \in D.[\forall d \in D(x \cdot d = y \cdot d) \Rightarrow x = y].$$

It is not to be confused with *weak extensionality, i.e.* $D \models (\xi)$.

- Extensionality, a stronger property, says that *each element* of the structure D is completely determined by its applicative behaviour *i.e.* argument-value correspondence; equivalently, each representable function of D has precisely one representative. It is an easy exercise to see that D is extensional iff D satisfies (ξ) and (η) where (η) is

  $$(\eta) \qquad \forall M \in \Lambda.D \models \lambda y.My = M \quad y \text{ not free in } M.$$

- However, weak extensionality merely says that the *denotation of each abstraction term* is completely determined by its applicative behaviour which, as we have seen, is entirely in keeping with the "abstractions as functions" perspective.

It is interesting to note that there are applicative structures that satisfies (ext) but fail to be λ-models. See [CDHL84] for some characterization results.

which means $\forall M, N.\forall \rho$

$$(\xi) \qquad \{\forall d \in D.[\![M]\!]_{\rho[x:=d]} = [\![N]\!]_{\rho[x:=d]}\} \Rightarrow [\![\lambda x.M]\!]_\rho = [\![\lambda x.N]\!]_\rho.$$

This would be an entirely reasonable stipulation for a model of $\lambda\beta$ were it not for the fact that the structure consisting of all closed λ-terms (which *should* be the archetypal model) does not satisfy the ξ-axiom. [3] Structures that satisfy all the above-mentioned properties but not necessarily axiom (ξ) are known as λ-*algebras*; if in addition they satisfy (ξ), they are known as λ-*models*. Their precise definitions will be given in the sequel.

3.1.3 Equivalent Characterizations of λ-models

In the following, we present three equivalent formulations of λ-models under the headings of *environment λ-models, functional λ-models,* and *first order λ-models* and a *category-theoretic characterization of λ-models.*

I. Environment λ-Models [HL80]
This approach is the most straightforward and obvious one. It delineates from first principles what is required of a model of the λ-calculus. An *environment λ-model* is a structure $\langle D, \cdot, [\![\text{-}]\!]_\text{-}\rangle$ such that

(1) $\langle D, \cdot\rangle$ *is an applicative structure.*

(2) *A well-defined semantic function* $[\![\text{-}]\!]_\text{-} : \Lambda(\underline{D}) \times D^{\mathsf{Var}} \to D$ such that for every environment $\rho \in D^{\mathsf{Var}}$, the map $[\![\text{-}]\!]_\rho : \Lambda(\underline{D}) \to D$ satisfies:

$$[\![x]\!]_\rho = \rho(x),$$

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

(3) D *is a model of* $\lambda\beta$: $\forall M, N \in \Lambda$

$$\lambda\beta \vdash M = N \Rightarrow D \models M = N \quad \textit{i.e.} \quad \forall \rho.[\![M]\!]_\rho = [\![N]\!]_\rho.$$

(4) D *is weakly extensional i.e.* $D \models (\xi)$.

[3] This is an immediate corollary of the ω-*incompleteness* of $\lambda\beta(\lambda\beta\eta)$ due to Gordon Plotkin [Plo74]. ω-rule is the following:

$$(\omega) \qquad \frac{\forall Z \in \Lambda^o.MZ = NZ}{M = N}$$

Plotkin constructed two closed λ-terms M, N such that $\forall Z \in \Lambda^o.MZ = NZ$ but $\lambda\beta \not\vdash M = N$ This reveals the inherent intensional character of the λ-calculus.

II. **Functional λ-Models** [Mey82,Koy82,Koy84]
The functional approach presents the notion of a λ-model at a rather more abstract level, explicating the precise relationship between the ground domain and its (representable) "function space". It is precisely this relationship that the category-theoretic characterization seeks to capture.

A *functional λ-model* is a structure $\langle D, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]^{\mathsf{Gr}} \rangle$ such that

(1) $\langle D, \cdot \rangle$ *is an applicative structure* where "·" is the (canonical) application operation associated with the Fun map

$$[D \to_{\mathsf{r}} D] \overset{\mathsf{Gr}}{\hookrightarrow} D \overset{\mathsf{Fun}}{\twoheadrightarrow} [D \to_{\mathsf{r}} D].$$

We have

$$\forall x, y \in D. x \cdot y = \mathsf{Ap}(\mathsf{Fun}(x), y)$$

where $\mathsf{Ap} : D^D \times D \to D$ is the usual application map. For simplicity, we will omit Ap and represent application by juxtaposition.

(2) $\langle \mathsf{Fun}, \mathsf{Gr} \rangle$ *is a* retract *of the representable function space* $[D \to_{\mathsf{r}} D]$ *into* D (Notationally, $[D \to_{\mathsf{r}} D] \lhd D$).

(3) *Closure of* $[D \to_{\mathsf{r}} D]$ *under λ-definable algebraic functions* [4] i.e. for any λ-term M (which may or may not contain abstractions as subterms), $(d \mapsto [\![M]\!]^{\mathsf{Gr}}_{\rho[x:=\underline{d}]}) \in [D \to_{\mathsf{r}} D]$. More pedantically but equivalently, we require the following associated *partial* semantic function $[\![\mathsf{Gr}]\!]\text{-}\_ : \Lambda(\underline{D}) \times D^{\mathsf{Var}} \rightharpoonup D$ to be *total*:

a. $[\![x]\!]^{\mathsf{Gr}}_{\rho} \overset{\text{def}}{=} \rho(x)$

b. $[\![\underline{d}]\!]^{\mathsf{Gr}}_{\rho} \overset{\text{def}}{=} d$

c. $$[\![MN]\!]^{\mathsf{Gr}}_{\rho} \overset{\text{def}}{=} \begin{cases} [\![M]\!]^{\mathsf{Gr}}_{\rho} \cdot [\![N]\!]^{\mathsf{Gr}}_{\rho} & \text{if } [\![M]\!]^{\mathsf{Gr}}_{\rho} \,\&\, [\![N]\!]^{\mathsf{Gr}}_{\rho} \text{ defined;} \\ \text{undefined} & \text{else.} \end{cases}$$

d. $$[\![\lambda x.M]\!]^{\mathsf{Gr}}_{\rho} \overset{\text{def}}{=} \begin{cases} \mathsf{Gr}(d \mapsto [\![M]\!]^{\mathsf{Gr}}_{\rho[x:=\underline{d}]}) & \text{if } \forall d \in D. [\![M]\!]^{\mathsf{Gr}}_{\rho[x:=\underline{d}]} \text{ is defined } \& \\ & (d \mapsto [\![M]\!]^{\mathsf{Gr}}_{\rho[x:=\underline{d}]}) \in [D \to_{\mathsf{r}} D] \\ \text{undefined} & \text{else.} \end{cases}$$

[4] Given an applicative structure $\langle D, \cdot \rangle$ and an interpretation of λ-terms, a function $f \in D^D$ is *λ-definable algebraic* if $\exists M \in \Lambda.\mathsf{FV}(M) \subseteq \{x\}.\exists \rho.\forall d \in D. f(d) = [\![M[x := \underline{d}]]\!]$. Note that the λ-term M above may well contain abstractions as subterms. Closure under algebraic functions alone will not do.

Conditions (1) and (2) lend themselves to succinct characterization and their significance is readily understood; not so for condition (3).

- What is the model-theoretic import of condition (3)?
- What are the conditions in this functional presentation that correspond to the validity of (β) and (ξ) axioms?

A structure $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$ satisfying (1) and (2) is called a *functional domain.* The following Lemma spells out the significance of condition (3).

LEMMA **3.1.3.1** *Let* $\langle D, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]^{\mathsf{Gr}}\rangle$ *be a functional domain. Then* D *is a* λ*-model iff*

(i) $\langle D, \cdot\rangle$ *is combinatory complete, and*

(ii) The "choice" function, $\mathsf{Gr} \circ \mathsf{Fun} \in [D \to_{\mathsf{r}} D]$ *is internally definable.*

In other words, condition (3) in the definition of functional λ*-model is equivalent to (i) and (ii) above. Observe that* $[\![1]\!] \equiv [\![\lambda xy.xy]\!] = \mathsf{Gr}(\mathsf{Gr} \circ \mathsf{Fun})$.

PROOF

"$\Rightarrow$" (i) is clearly satisfied. We show that the representable function f having $[\![\lambda xy.xy]\!]$ (which denotes in D) as its representative is precisely $\mathsf{Gr} \circ \mathsf{Fun}$. For any $d \in D$,

$$\begin{aligned} [\![\lambda xy.xy]\!] \cdot d &= [\![\lambda y.\underline{d}y]\!] \\ &= \mathsf{Gr}(c \mapsto [\![\underline{d}\underline{c}]\!]) \\ &= \mathsf{Gr}(c \mapsto d \cdot c) \\ &= \mathsf{Gr}(c \mapsto \mathsf{Fun}(d)c) \\ &= \mathsf{Gr} \circ \mathsf{Fun}(d) \end{aligned}$$

"$\Leftarrow$" According to the definition of functional λ-model, it remains to show that with respect to the associated semantic function $[\![\text{-}]\!]$ all λ-definable algebraic functions are representable. That is, for $M(\vec{x}) \in \Lambda(\vec{x})$ which means $\mathsf{FV}(M) \equiv \vec{x} = x_1, \cdots, x_n$ that

(a) $\forall \vec{d} \in D^n.[\![M(\underline{\vec{d}})]\!]$ is defined, and

(b) $(\vec{d} \mapsto [\![M(\underline{\vec{d}})]\!]) \in [D^n \to_{\mathsf{r}} D]$; where $M(\vec{d}) \stackrel{\text{def}}{=} M(\vec{x})[\vec{x} := \vec{d}]$.

We show by structural induction. Consider the harder case of $M(\vec{x}) \equiv \lambda y.N(\vec{x}, y)$. By induction hypothesis,

$$\forall \vec{d} \in D^n . \forall e \in D . [\![N(\underline{\vec{d}e})]\!] \text{ is defined,}$$

and

$$\forall \vec{d} \in D^n . \forall e \in D . \exists c \in D . c\vec{d}e = [\![N(\underline{\vec{d}e})]\!].$$

Now, $(e \mapsto c\vec{d}e) = \mathsf{Fun}(c\vec{d}) \in [D \to_{\mathsf{r}} D]$. Hence,

$$[\![M(\underline{\vec{d}})]\!] = \mathsf{Gr}(\mathsf{Fun}(c\vec{d})) = \epsilon \cdot (c\vec{d}) \text{ exists,}$$

where $\epsilon = \mathsf{Gr}(\mathsf{Gr} \circ \mathsf{Fun})$. Moreover, $(\vec{d} \mapsto \epsilon(c\vec{d}))$ is algebraic and hence representable by combinatory completeness. Thus, we have shown (b).

□

The functional and environment λ-models are completely equivalent.

PROPOSITION **3.1.3.2** *(i) For each functional λ-model $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$, the corresponding $\langle D, \cdot, [\![-]\!]_-\rangle$ is an (environment) λ-model.*

(ii) An environment λ-model $\langle D, \cdot, [\![-]\!]_-\rangle$ defines $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$, a functional λ-model with the associated semantic function $[\![-]\!]_-^{\mathsf{Gr}}$ by putting $\mathsf{Gr}(f) = [\![\lambda x.\underline{d}x]\!]$ where $d \in \mathsf{rep}(f)$.

(iii) Further, the constructions in (i) and (ii) are each other's inverse.

PROOF Straightforward modification of the proof of Proposition 1.4.14 in [Koy84]. □

We can now answer the question we posed earlier on in the section and emphasize an important observation.

COROLLARY **3.1.3.3** *Let $\langle D, \cdot\rangle$ be an applicative structure. Suppose that abstraction terms are given an "extensional" interpretation, i.e. as in statement (3d) in the definition of functional λ-model. Then, (ξ) is satisfied immediately and*

$$\langle \mathsf{Gr}, \mathsf{Fun}\rangle : [D \to_{\mathsf{r}} D] \lhd D \iff D \vDash (\beta)$$

□

III. **First order models** [Mey82,Sco80b]
The theory $\lambda\beta$ is logic free. At first sight, $\lambda\beta$ seems tantalisingly like a purely equational theory. If it were so, then the class of all models of $\lambda\beta$ would be what universal algebraists know as a variety. Upon closer scrutiny however, the λ-abstraction operator is seen to exhibit subtly the strength of a universal quantifier in the presence of the axiom of weak extensionality, *i.e.*

$$\forall x.M = N \iff \lambda x.M = \lambda x.N.$$

That λ-abstraction is the critical "operator" which disqualifies $\lambda\beta$ from being a purely equational theory can be seen from the models of *Strong Combinatory Logic*. *Combinatory Logic* is a sibling system of $\lambda\beta$ in which the effect of λ-abstraction is simulated by combinators **S** and **K**. Strong Combinatory Logic, which is *finitely* axiomatizable over Combinatory Logic (by way of five combinatory axioms known as $\mathbf{A}_\beta$ in [Bar84, page 161]) is an equivalent formulation of $\lambda\beta$ in that they prove exactly the same formulas. The significant point to note is that Strong Combinatory Logic is a purely *equational* theory. Its models are called *λ-algebras*. Perhaps somewhat surprisingly, λ-algebras do not necessarily satisfy the weak extensionality axiom which seems entirely natural and in keeping with the view that λ-abstractions mimick set-theoretic functions. λ-algebras that do satisfy the weak-extensionality axiom turn out to coincide with the two previous equivalent notions of λ-models. We will call λ-models defined using combinatory logic *first-order λ-models*. Both Dana Scott and Albert Meyer proposed slightly different formulations of first-order λ-models. (They are of course equivalent.) We will just mention the neater version due to Meyer.

DEFINITION **3.1.3.4** ([Mey82]) A *combinatory model* is a triple $\langle D, \cdot, \epsilon\rangle$ such that $\langle D, \cdot\rangle$ is a combinatory complete applicative structure (or equivalently, expandable into a combinatory algebra), and $\epsilon \in D$ satisfying:

(ϵ1) $\forall x, y.\epsilon xy = xy$

(ϵ2) $\forall x, y.[\forall z(xz = yz) \Rightarrow \epsilon x = \epsilon y.]$

(ϵ2) is also known as the *Meyer-Scott Axiom* (MS). We say that a combinatory model is *stable* if in addition the following axiom is satisfied:

(Stability) $\epsilon\epsilon = \epsilon.$

Crucially, we observe that a combinatory model is a model which satisfies:

- a *one-sorted equational presentation* comprising the usual equational axioms defining the **S**, **K** combinators and the equational axiom (ϵ1)), and
- a *first-order implication* (ϵ2).

The following proposition establishes that *stable combinatory models* is yet another (equivalent) way of formulating λ-models.

PROPOSITION **3.1.3.5** *(i) There is a one-to-one correspondence between stable combinatory models* $\langle D, \cdot, \epsilon\rangle$ *and functional* λ*-models* $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$.

(ii) A combinatory model $\langle D, \cdot, \epsilon\rangle$ *is stable iff* $\epsilon = [\![\lambda xy.xy]\!]$.

PROOF Let $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$ be a functional λ-model. Define $\epsilon = \mathsf{Gr}(\mathsf{Gr} \circ \mathsf{Fun})$. Let $x, y \in D$. Then,

$$\begin{aligned} \epsilon xy &= \mathsf{Fun}(\mathsf{Fun}(\mathsf{Gr}(\mathsf{Gr} \circ \mathsf{Fun}))x)y \\ &= \mathsf{Fun}(\mathsf{id}(\mathsf{Gr} \circ \mathsf{Fun})x)y \\ &= (\mathsf{Fun}(\mathsf{Gr}(\mathsf{Fun}x)))y \\ &= (\mathsf{Fun}x)y \\ &= xy. \end{aligned}$$

This establishes (ϵ1). Suppose $\forall z.xz = yz$. This means that $x, y \in \mathsf{rep}(f)$ for some representable function f. Now,

$$\epsilon x = \mathsf{Fun}(\mathsf{Gr}(\mathsf{Gr} \circ \mathsf{Fun}))x = \mathsf{Gr}(\mathsf{Fun}(x)).$$

Since $\mathsf{Fun}(x) = \mathsf{Fun}(y)$, we have $\mathsf{Gr}(\mathsf{Fun}(x)) = \mathsf{Gr}(\mathsf{Fun}(y))$ which implies that $\epsilon x = \epsilon y$. We have thus established (ϵ2). Stability can be shown in the similar way.

Conversely, given a stable combinatory model $\langle D, \cdot, \epsilon\rangle$, define $\mathsf{Gr} : [D \to_{\mathsf{r}} D] \to D$ by $\mathsf{Gr}(f) = \epsilon \cdot d$ for any $d \in \mathsf{rep}(f)$. Combinatory completeness holds by definition. It is easy to show that $\langle \mathsf{Gr}, \mathsf{Fun}\rangle : [D \to_{\mathsf{r}} D] \lhd D$ and hence $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$ is a functional domain. Observe that $\mathsf{Gr} \circ \mathsf{Fun}$ is a representable function with the representative ϵ. Hence by Lemma 3.1.3.1, $\langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$ is a functional λ-model. This completes the proof of (i).

For (ii), observe that

$$[\![\lambda xy.xy]\!]_\rho = \mathsf{Gr}(d \mapsto \mathsf{Gr}(e \mapsto d \cdot e)) = \epsilon \cdot \epsilon.$$

□

IV. Category-Theoretic Characterization [Lam80,Sco80b,LM84]
By bringing the machinery of category theory to bear on the model theory of λ-calculus, the following elegant characterization has been obtained: In a category C, we say that a morphism $f \in \mathsf{C}(a, b)$ is *principal* if $\forall g \in \mathsf{C}(a, b).\exists h \in \mathsf{C}(a, a).g = f \circ h$. Then,

(1) a yields a *combinatory algebra* iff $\exists f \in \mathsf{C}(a, a^a)$ principal and $a \times a \lhd a$,

(2) a yields a λ-*algebra* iff $a^a \lhd a$; and

(3) a yields a λ-*algebra satisfying* (η) iff $a^a \cong a$.

In the case of a category $\mathbf{C}$ *having enough points* i.e. if 1 is the terminal object, then

$$\forall h \in \mathbf{C}(1,a).f \circ h = g \circ h \Rightarrow f = g.$$

then (2) and (3) above characterize λ-*models* and *extensional* λ-*models* respectively. A category that has enough points is also described as *concrete*. Crucially, in such a category, the behaviour of a morphism is uniquely determined by the behaviour between the corresponding *global elements*; more precisely, a morphism $f : a \to b$ can be identified with a function from $\mathbf{C}(1,a)$ to $\mathbf{C}(1,b)$. The condition of a category "having enough points" corresponds exactly to the satisfaction of the axiom (ξ). See Chapter 5 for more details.

3.2 Lazy Lambda Models

In this section, we introduce (quasi) *applicative structure with divergence* and its essentially equivalent counterpart (quasi) *applicative transition system* and show how they provide a natural framework in which to define *lazy* λ-*models*. We prove a general *computational adequacy result* for a class of ordered lazy λ-models with structures that are amenable to standard (as in Scott's D_∞) index calculation techniques.

3.2.1 Applicative Transition Systems

The original and definitive account of the notion of applicative structure with divergence and lambda transition system is Chapter 6 of Samson Abramsky's PhD thesis [Abr87], in which he deals with the subject of lazy λ-calculus in the grand framework of the *Stone duality* between domains and their logics of observable properties.

Abramsky introduced the *bisimulation ordering*, $\sqsubseteq^B$ (see Chapter 4) to study operational properties of the lazy λ-calculus. The *Abramsky lazy* λ-*theory* (see Chapter 4), $\lambda\ell$, is derived from a particular operational model — the transition system $\langle \Lambda^o, \Downarrow \rangle$. It is natural to ask what is the general mathematical structure of which the previous transition system is an instance. Abramsky proposed the following notions in reply to the question.

DEFINITION **3.2.1.1 (Abramsky)** (i) A *quasi-applicative transition system* (q-ats) is a structure $\langle A, \mathsf{eval} \rangle$ such that $\mathsf{eval} : A \rightharpoonup A^A$ and $A/\mathsf{dom}(\mathsf{eval}) \neq$

$\emptyset$.

NOTATION:

$$a \Downarrow f \stackrel{\text{def}}{=} a \in \mathsf{dom}(\mathsf{eval}) \;\&\; \mathsf{eval}(a) = f,$$
$$a\Downarrow \stackrel{\text{def}}{=} a \in \mathsf{dom}(\mathsf{eval}),$$
$$a\Uparrow \stackrel{\text{def}}{=} \neg(a\Downarrow).$$

(ii) Let $\langle A, \mathsf{eval}\rangle$ be a q-ats and $\mathsf{Rel}(A) \stackrel{\text{def}}{=} \wp(A\times A)$. Define $F : \mathsf{Rel}(A) \to \mathsf{Rel}(A)$ by,

$$F(R) \stackrel{\text{def}}{=} \{\, (a,b) : a \Downarrow f \Rightarrow b \Downarrow g \;\&\; \forall c \in A.f(c)Rg(c) \,\}.$$

Then, $R \in \mathsf{Rel}(A)$ is an *applicative bisimulation* if $R \subseteq F(R)$. We define $\lesssim^B$, an operational pre-order known as *bisimulation ordering* as follows. For $a, b \in A$, $a \lesssim^B b$ holds if there is an applicative bisimulation R such that aRb. In other words,

$$\lesssim^B \stackrel{\text{def}}{=} \bigcup\{\, R \in \mathsf{Rel}(A) : R \subseteq F(R) \,\}$$

and hence is the maximal fixpoint of the monotonic function F. Since $\Downarrow$ is a partial function, it is easily shown that the closure ordinal of F is less than or equal to ω. $\lesssim^B$ can thus be described more explicitly as follows:

$$a \lesssim^B b \stackrel{\text{def}}{=} \forall k \in \omega.a \lesssim^B_k b$$

where

$$\forall a, b \in A.a \lesssim^B_0 b,$$

$$a \lesssim^B_{k+1} b \stackrel{\text{def}}{=} a \Downarrow f \Rightarrow b \Downarrow g \;\&\; \forall c \in A.f(c) \lesssim^B_k g(c).$$

We abbreviate $a \lesssim^B b \;\&\; b \lesssim^B a$ as $a \sim^B b$. It is easily checked that $\lesssim^B$ is a preorder; hence, $\sim^B$ is an equivalence relation. Denote the equivalence class of a as $[a]$.

(iii) An *applicative transition system* (ats) is a q-ats $\langle A, \mathsf{eval}\rangle$ satisfying:

$$\text{(ats)} \qquad \forall a \in A.a \Downarrow f \;\&\; \forall b, c \in A.b \lesssim^B c \Rightarrow f(b) \lesssim^B f(c).$$

Significantly, an ats has a quotient $\langle A/\sim^B, \mathsf{eval}/\sim^B\rangle$ where

$$\mathsf{eval}/\sim^B [a] \stackrel{\text{def}}{=} \begin{cases} [b] \mapsto [f(b)] & \text{if } a \Downarrow f, \\ \text{undefined} & \text{if } a\Uparrow. \end{cases}$$

In the case of a (total) applicative structure, $\langle D, \cdot \rangle$, and its associated structure, $\langle D, \mathsf{Fun} \rangle$, such that $\mathsf{Fun} : D \to D^D$, we recall that the application operator "$\cdot$" and Fun are essentially the same notion and they are inter-definable. In the same way, we will show how ats is related to *applicative structure with divergence.*

DEFINITION **3.2.1.2** (i) A *quasi-applicative structure with divergence* (q-aswd) is a structure $\langle A, \cdot, \Uparrow \rangle$ such that

- $\langle A, \cdot \rangle$ is an applicative structure;
- $\emptyset \neq \Uparrow \subseteq A$ is a *divergence predicate* satisfying

(left-strictness) $$x\Uparrow \Rightarrow \forall y \in A.x \cdot y\Uparrow.$$

Similarly, given $\langle A, \cdot, \Uparrow \rangle$ we can define a preorder $\sqsubseteq^B$ with the following recursive specification:

$$a \sqsubseteq^B b \stackrel{\text{def}}{=} a\Downarrow \Rightarrow b\Downarrow \;\&\; \forall c \in A.a \cdot c \sqsubseteq^B b \cdot c$$

as the maximal fixpoint of a monotonic function as before, where $\Downarrow \stackrel{\text{def}}{=} \neg(\Uparrow)$.

(ii) An *applicative structure with divergence* (aswd) $\langle A, \cdot, \Uparrow \rangle$ is a q-aswd satisfying

(aswd) $$\forall a, b, c \in A.b \sqsubseteq^B c \Rightarrow a \cdot b \sqsubseteq^B a \cdot c.$$

q-aswd (aswd) and q-ats (ats) are essentially equivalent in a sense which we will make clear in the following.

Given a q-aswd (aswd) $\mathcal{A} = \langle A, \cdot, \Uparrow \rangle$, define the associated evaluation function $\mathsf{eval} : A \rightharpoonup A^A$ by

$$\mathsf{eval}(a) \stackrel{\text{def}}{=} \begin{cases} x \mapsto a \cdot x & \text{if } a\Downarrow; \\ \text{undefined} & \text{if } a\Uparrow. \end{cases}$$

Clearly, the structure $\mathcal{A}^{q-ats} \stackrel{\text{def}}{=} \langle A, \mathsf{eval} \rangle$ thus defined is a q-ats. Also, the axiom (aswd) implies (ats); hence $\mathcal{A}^{ats} \stackrel{\text{def}}{=} \langle A, \mathsf{eval} \rangle$ is an ats if $\mathcal{A}$ is an aswd.

Conversely, given a q-ats $\mathcal{A} = \langle A, \mathsf{eval} \rangle$ such that there exists an (distinguished) element $\perp$ belonging to $A \backslash \mathsf{dom}(\mathsf{eval})$, define the associated binary operation $\cdot : A \times A \to A$ by

$$a \cdot b \stackrel{\text{def}}{=} \begin{cases} f(b) & \text{if } \mathsf{eval}(a) = f, \\ \perp & \text{if } a \notin \mathsf{dom}(\mathsf{eval}); \end{cases}$$

and define $\Uparrow \stackrel{\text{def}}{=} A\backslash\text{dom(eval)}$. Observe that the axiom (ats) implies (aswd). It is clear that $\mathcal{A}^{q-aswd} \stackrel{\text{def}}{=} \langle A, \cdot, \Uparrow\rangle$ is a q-aswd; and $\mathcal{A}^{aswd} \stackrel{\text{def}}{=} \langle A, \cdot, \Uparrow\rangle$ is an aswd if $\mathcal{A}$ is an ats. In fact, it is easy to see $(\mathcal{A}^{q-aswd})^{q-ats} = \mathcal{A}$; and $(\mathcal{A}^{aswd})^{ats} = \mathcal{A}$ if $\mathcal{A}$ is a q-ats and an ats respectively.

However, for a q-aswd $\mathcal{A} = \langle A, \cdot, \Uparrow\rangle$, we only have $(\mathcal{A}^{q-ats})^{q-aswd} = \mathcal{A}$ and $(\mathcal{A}^{ats})^{aswd} = \mathcal{A}$ *provided* $\Uparrow = \{\bot\}$, a singleton set.

To summarize, we have shown

PROPOSITION **3.2.1.3** *(i) Given a q-ats $\mathcal{A}$ such that $\exists \bot \in A\backslash\text{dom(eval)}$, then $\mathcal{A}^{q-aswd}$ as defined above is a q-aswd; similarly, if $\mathcal{A}$ is an ats.*

(ii) Given a q-aswd $\mathcal{A}'$, $\mathcal{A}'^{q-ats}$ as defined above is a q-ats. Similarly, if $\mathcal{A}$ is an aswd.

(iii) Moreover, we always have $(\mathcal{A}^{q-aswd})^{q-ats} = \mathcal{A}$; but $(\mathcal{A}'^{q-ats})^{q-aswd} = \mathcal{A}'$ holds provided $\Uparrow = \{\bot\}$.

□

From now on, unless otherwise specified, we will mostly be concerned with those q-aswd's such that $\Uparrow = \{\bot\}$.

Intuitively, a q-aswd is a specialized applicative structure with the following distinctive features:

(1) q-aswd has a built-in divergent element; "non-termination" is dealt with axiomatically;

(2) application is left-strict.

Let $\langle A, \cdot\rangle$ be an applicative structure and $X \subseteq A$. A representable function $f \in A^A$ is said to be *X-representable* if f has a representative in X. We define $\text{rep}_X(f) \stackrel{\text{def}}{=} X \cap \text{rep}(f)$. Denote $[A \to_X A]$ as the class of all X-representable functions.

3.2.2 Lazy λ-Models

Now, we are in a position to formulate (equivalent) structures which we will propose as candidates for the notion of a *lazy λ-model.*

NOTATION **3.2.2.1** Recall that for $M \in \Lambda^o$, $M\Downarrow \iff \exists N \in \Lambda.\lambda\beta \vdash M = \lambda x.N$.

DEFINITION **3.2.2.2** We give three equivalent formulations of *lazy λ-model.*
I. **Environment Lazy λ-Model** $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_-\rangle$

(1) $\langle A, \cdot, \Uparrow, [\![-]\!]_-\rangle$ is a q-aswd such that $\Uparrow = \{\bot\}$.

(2) $[\![-]\!]_-$ is *homomorphic* with respect to application, *i.e.* $\forall M, N \in \Lambda(\underline{A}), \forall a \in A$,

$$\begin{aligned} [\![\underline{a}]\!]_\rho &= a, \\ [\![x]\!]_\rho &= \rho(x), \\ [\![MN]\!]_\rho &= [\![M]\!]_\rho \cdot [\![N]\!]_\rho. \end{aligned}$$

(3) $\mathcal{A} \models (\beta)$, *i.e.* $\lambda\beta \vdash M = N \Rightarrow \mathcal{A} \models M = N$.

(4) $\mathcal{A} \models (\xi)$ where ξ is as before.

(5) $\forall M \in \Lambda^o.M{\Downarrow} \Rightarrow \mathcal{A} \models M{\Downarrow}$.

II. **Functional Lazy λ-Model** $\mathcal{A} = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]_-\rangle$

(1) $\langle A, \mathsf{Fun}\rangle$ is a q-ats such that $A \backslash \mathsf{dom}(\mathsf{Fun}) = \{\bot\}$;

(2) $\langle \mathsf{Gr}, \mathsf{Fun}\rangle : [A \to_{\Downarrow} A] \lhd A$ *i.e.* $A \xrightarrow{\mathsf{Fun}} [A \to_{\Downarrow} A] \xrightarrow{\mathsf{Gr}} A$ with $\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}_{[A \to_{\Downarrow} A]}$; such that $\mathsf{range}(\mathsf{Gr}) \cap {\Uparrow} = \emptyset$ where the representable function space is defined with respect to $\langle A, \mathsf{Fun}\rangle^{q-aswd}$; (note that (2) implies that $[A \to_{\Downarrow} A]_\bot \lhd A$.)

(3) $[\![-]\!]_- : \Lambda(\underline{A}) \times A^{\mathsf{Var}} \to A$ such that $\forall M, N \in \Lambda(\underline{A}), \forall a \in A$,

a. (homomorphism)

$$\begin{aligned} [\![\underline{a}]\!]_\rho &= a, \\ [\![x]\!]_\rho &= \rho(x), \\ [\![MN]\!]_\rho &= [\![M]\!]_\rho \cdot [\![N]\!]_\rho; \end{aligned}$$

where the application $\cdot$ is as defined in $\langle A, \mathsf{Fun}\rangle^{q-aswd}$;

b. (weak-extensionality) $[\![\lambda x.M]\!]_\rho = \mathsf{Gr}(d \mapsto [\![M]\!]_{\rho[x:=\underline{d}]})$ such that $[A \to_{\Downarrow} A]$ is closed under λ-definable algebraic functions.

III. **First Order Lazy λ-Model** $\mathcal{A} = \langle A, \cdot, \epsilon, [\![-]\!]_-\rangle$

(1) $\langle A, \cdot, {\Uparrow}\rangle$ is a q-aswd such that ${\Uparrow} = \{\bot\}$.

(2) $\langle A, \cdot\rangle$ is a combinatory algebra such that

$$\mathcal{A} \models \mathbf{K}{\Downarrow}, \mathbf{S}{\Downarrow}, \mathbf{K}x{\Downarrow}, \mathbf{S}x{\Downarrow}, \mathbf{S}xy{\Downarrow}.$$

(3) $\mathcal{A}$ satisfies the following axioms:

$$\begin{aligned} &(\epsilon 1) && \forall x, y.\epsilon xy = xy; \\ &(\mathrm{MS}) && \forall x, y.[\forall z.xz = yz] \Rightarrow \epsilon x = \epsilon y; \\ &(\text{stability}) && \epsilon \cdot \epsilon = \epsilon. \end{aligned}$$

PROPOSITION **3.2.2.3** *(i) a. Every environment lazy λ-model $\mathcal{A}$ defines an associated functional lazy λ-model $\mathcal{A}^f$.*

b. Every functional lazy λ-model $\mathcal{A}$ defines an associated environment lazy λ-model $\mathcal{A}^e$.

c. The constructions in (a) and (b) are each other's inverse.

(ii) There is a natural 1-1 correspondence between functional lazy λ-models and first order lazy λ-models.

PROOF We will prove (i) and leave (ii) as an exercise. (a) Given an environment λ-model $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle$, define $\mathcal{A}^f = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]^f_- \rangle$ where $\langle A, \mathsf{Fun} \rangle \stackrel{\text{def}}{=} \langle A, \cdot, \Uparrow \rangle^{q-ats}$ which is a q-ats. Define $\mathsf{Gr} : [A \to_{\Downarrow} A] \to A$ by

$$\mathsf{Gr}(f) = [\![\lambda x.\underline{a}x]\!] \qquad \text{where } a \text{ is a representative of } f.$$

Note that $\mathsf{range}(\mathsf{Gr}) \cap \Uparrow = \emptyset$, because by definition abstractions are convergent in an environment model. Then, for f with representative a and $d \in A$;

$$\begin{aligned} \mathsf{Fun}(\mathsf{Gr}(f))d &= \mathsf{Fun}([\![\lambda x.\underline{a}x]\!])d \\ &= [\![\lambda x.\underline{a}x]\!] \cdot d \\ &= a \cdot d = f(d). \end{aligned}$$

Hence, $\langle \mathsf{Gr}, \mathsf{Fun} \rangle : [A \to_{\Downarrow} A] \lhd A$.

Define $[\![-]\!]^f_\rho : \Lambda(\underline{A}) \times A^{\mathsf{Var}} \to A$ as prescribed by the definition of functional lazy λ-model. We prove by mutual induction the following assertions:

(1) $\forall M \in \Lambda(\underline{A}).[\![M]\!]_\rho = [\![M]\!]^f_\rho$,

(2) $\forall M \in \Lambda(\underline{A}).F_{M(x),\rho} \stackrel{\text{def}}{=} (d \mapsto [\![M]\!]^f_{\rho[x:=\underline{d}]}) \in [A \to_{\Downarrow} A]$; that is to say

$$\forall d \in A.\exists r_{M,\rho} \in A.F_{M,\rho}(d) = r_{M,\rho} \cdot d.$$

The base cases of (1) and (2) are obvious. We prove the inductive cases of (1) first. The case of $M \equiv PQ$ is clear. Suppose $M \equiv \lambda x.P$. Then, $[\![\lambda x.P]\!]^f_\rho = \mathsf{Gr}(F_{P,\rho})$, by an appeal to induction hypothesis of (2). But by induction hypothesis of (1), $[\![P]\!]_{\rho[x:=\underline{d}]} = [\![P]\!]^f_{\rho[x:=\underline{d}]}$, therefore we may choose $r_{P,\rho}$ to be $[\![\lambda x.P]\!]_\rho$. Hence,

$$[\![\lambda x.P]\!]^f_\rho = [\![\lambda x.\underline{[\![\lambda x.P]\!]_\rho}x]\!]_\rho = [\![\lambda x.P]\!]_\rho.$$

We prove the inductive cases for (2). Suppose $M \equiv PQ$. Then,

$$\begin{aligned}
[\![PQ]\!]^f_{\rho[x:=\underline{d}]} &= [\![P]\!]^f_{\rho[x:=\underline{d}]}[\![Q]\!]^f_{\rho[x:=\underline{d}]} \quad \text{Ind. Hyp. (2)} \\
&= (r_{P,\rho} \cdot d) \cdot (r_{Q,\rho} \cdot d) \\
&= [\![\lambda x.\underline{r_{P,\rho}} x(\underline{r_{Q,\rho}} x)]\!] \cdot d.
\end{aligned}$$

Suppose $M(x) \equiv \lambda y.P(x,y)$. For any d,

$$F_{M(x),\rho}(d) = [\![\lambda y.P(x,y)]\!]^f_{\rho[x:=\underline{d}]} = \mathsf{Gr}(e \mapsto [\![P(\underline{d},y)]\!]^f_{\rho[y:=\underline{e}]}).$$

Now, by induction hypothesis (2),

$$\exists r_{P(\underline{d},y),\rho}.\forall e \in A.[\![P(\underline{d},y)]\!]^f_{\rho[y:=\underline{e}]} = r_{P(\underline{d},y),\rho} \cdot e.$$

But, by induction hypothesis (1),

$$[\![P(\underline{d},y)]\!]^f_{\rho[y:=\underline{e}]} = [\![P(\underline{d},y)]\!]_{\rho[y:=\underline{e}]}.$$

Hence, taking $r_{P(\underline{d},y),\rho}$ to be $[\![\lambda y.P(\underline{d},y)]\!]_\rho$, we have:

$$\begin{aligned}
F_{M(x),\rho}(d) &= [\![\lambda z.[\![\lambda y.P(\underline{d},y)]\!]_\rho z]\!] \\
&= [\![\lambda x.\lambda z.(\lambda y.P(x,y))z]\!]_\rho \cdot d \\
&= [\![\lambda xy.P(x,y)]\!]_\rho \cdot d.
\end{aligned}$$

Observe that the above argument is valid for all $d \in A$; and we are done.

(b) Given a functional lazy λ-model $\mathcal{A} = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]_- \rangle$. Define

$$\mathcal{A}^e = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle \text{ where } \langle A, \cdot, \Uparrow \rangle \stackrel{\text{def}}{=} \langle A, \mathsf{Fun} \rangle^{q-aswd}.$$

Then, it should be clear that $\langle A, \cdot, \Uparrow \rangle$ is a q-aswd. It is easy to check that $\mathcal{A}^e \models (\beta)$ & (ξ). Finally, (c) follows immediately from $[\![-]\!] = [\![-]\!]^f_-$. □

Let $\mathcal{A} = \langle A, \cdot, \Uparrow \rangle$ be a q-aswd. Define a binary relation $\sim_\Downarrow \subseteq A \times A$ as follows:

$$x \sim_\Downarrow y \stackrel{\text{def}}{=} [x\Downarrow \iff y\Downarrow] \;\&\; \forall z \in A.xz = yz.$$

Clearly, $\sim_\Downarrow$ is an equivalence relation and there is a 1-1 correspondence between $[A \to_\Downarrow A]_\perp$ and $A/\sim_\Downarrow$ as follows:

$$\begin{aligned}
\perp &\longleftrightarrow [\perp]_{\sim_\Downarrow}, \\
\langle 0, f \rangle &\longleftrightarrow [a]_{\sim_\Downarrow} \text{ where } \forall z \in A.f(z) = a \cdot z.
\end{aligned}$$

We define a few axioms.

DEFINITION **3.2.2.4** Let $\mathcal{A}$ be a lazy λ-model.

- •adequacy[5] $\forall M \in \Lambda^o.\mathcal{A} \models M{\Downarrow} \Rightarrow M{\Downarrow}$;
- •η_{cond} $\forall a \in A.\mathcal{A} \models [a{\Downarrow} \Rightarrow \lambda x.\underline{a}x = a]$;
- •Ext_{cond} $\forall x, y \in A.x \sim_{\Downarrow} y \Rightarrow x = y$.

LEMMA **3.2.2.5** *Let* $\mathcal{A} = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]_- \rangle$ *be a functional lazy* λ*-model such that* $\mathsf{dom}(\mathsf{Fun}) = A\backslash\{\perp\}$. *Then, the following are equivalent:*

(1) $\mathcal{A} \models (\text{Ext}_{\text{cond}})$,

(2) $\mathsf{Gr} \circ \mathsf{Fun} = \mathsf{id}_{\Downarrow_A}$ *which implies that* $A \cong [A \to_{\Downarrow} A]_{\perp}$,

(3) $\mathcal{A} \models (\eta_{\text{cond}})$.

PROOF "(1) $\iff$ (2)" First observe that $\forall x, y \in {\Downarrow}.x \sim_{\Downarrow} y \iff \mathsf{Fun}(x) = \mathsf{Fun}(y)$. Suppose (1). Since $\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}_{[A \to_{\Downarrow} A]}$ — by definition of lazy λ-model, for any convergent x, $\mathsf{Fun}(\mathsf{Gr}(\mathsf{Fun}(x))) = \mathsf{Fun}(x)$, and so, $\mathsf{Gr}(\mathsf{Fun}(x)) \sim_{\Downarrow} x$ by the preceding observation. By $(\text{Ext}_{\text{cond}})$, we have $\forall x{\Downarrow}.\mathsf{Gr}(\mathsf{Fun}(x)) = x$, *i.e.* the assertion (2). Suppose (2). Then, for $x{\Downarrow}$ & $y{\Downarrow}$, $x \sim_{\Downarrow} y \iff \mathsf{Fun}x = \mathsf{Fun}y \Rightarrow \mathsf{Gr}(\mathsf{Fun}(x)) = \mathsf{Gr}(\mathsf{Fun}(y))$. Applying (2), we get $x = y$.
"(2) $\iff$ (3)"follows from the observation that for convergent a, $[\![\lambda x.\underline{a}x]\!] = \mathsf{Gr}(\mathsf{Fun}(a))$. □

3.2.3 A General Adequacy Result

A denotational semantics of a programming language is said to be *computationally adequate* (or simply adequate) w.r.t. the operational semantics if the denotation of any program is well-defined precisely when the evaluation of that program converges. In this subsection, we prove an adequacy result for a class of "continuous" lazy λ-models whose structure lends themselves to labelled reduction analysis as in the case of Scott's D_∞ model. We prove the following *adequacy* result.

PROPOSITION **3.2.3.1** (**Adequacy**) *Let* $\mathcal{M} = \langle D, \mathsf{Fun}, \mathsf{Gr} \rangle$ *be a lazy* λ*-model that has lazy approximable application and that* Gr *is continuous. Then,*

$$\forall M \in \Lambda^o.\mathcal{M} \models M{\Downarrow} \Rightarrow M{\Downarrow}.$$

□

DEFINITION **3.2.3.2** A lazy λ-model $\langle D, \cdot, \overbrace{\Uparrow}^{[\![-]\!]} \rangle$ has *approximable application* iff

(i) $\langle D, \sqsubseteq \rangle$ is a cpo with least element $\perp$ — the axiomatic divergent element and that "$\cdot$":$D \times D \to D$ is ω-continuous.

(ii) There is a map Approx:$D \times \omega \to D$ such that for any fixed $d \in D$ and $n \in \omega$, Approx$(-,n)$ and Approx$(d,-)$ are continuous and monotonic respectively. For any $d, e \in D$:

$$(1)\ d = \bigsqcup_{n\in\omega} d_n \qquad (2)\ d_0 = \bot$$
$$(3)\ \bot \cdot e = \bot \qquad (4)\ d_{n+1} \cdot e \sqsubseteq (d \cdot e_n)_n$$
$$(5)\ (d_n)_m \sqsubseteq d_{\min(n,m)}$$

Note that (5) can be derived from axioms (1) and (2).

The main machinery for the proof of the above Proposition is that of *labelled reduction* á la Hyland and Wadsworth. See *e.g.* [Hyl76] where the technique is employed to prove the classic Approximation Theorem and Local Structure Theorem for the λ-models: D_∞ and $\wp\omega$. A thorough account of this methodology and the associated notion of *approximate relation* can be found in [Wad78] or [Bar84, chapter 14]. In [MP87], Plotkin and Mosses give two alternative proofs of the approximation theorem (vis-à-vis the λ-model D_∞) using technically pleasing and powerful methods. The first utilizes a certain kind of intermediate semantics for proving correspondences between denotational and operational semantics. The other illustrates a technique of Milne [Mil74,MS76], employing *recursively-specified inclusive predicates* (otherwise known as *logical relations*). See Chapter 4 for a brief discussion on logical relations and their application in the construction of fully-abstract models. We refer the reader also to the adequacy result of a general typed meta-language in [Plo85].

DEFINITION **3.2.3.4** (i) The terms of the *labelled λ-calculus (with $\bot$)*, $\Lambda^\omega\bot$, is defined by the following grammar:

$$\Lambda^\omega\bot \ni M ::= \bot | x | (\lambda x.M) | (MN) | (M^n)$$

where x and n range over Var and ω respectively. Elements of $\Lambda^\omega\bot$ are known as *labelled $\lambda\bot$-terms*. A *labelled λ-term* is a labelled $\lambda\bot$-term that does not contain $\bot$ as subterms. The collection of labelled λ-term is denoted Λ^ω. $|M|$ is the underlying $\lambda\bot$-term obtained from M by erasing the labels. For the rest of this section, the usual meta-variables for λ-terms like M, N, P, Q, etc. will range over labelled $\lambda\bot$-terms unless otherwise stated. Let $M \in \Lambda$. Define subterm(M) as the set of all subterms of M; and Seq* the set of all non-empty finite sequences of ω. To facilitate easy reading of the following proofs, we formalize a labelled λ-term M as a pair (M, I^M) such that

$$I^M : \mathsf{subterm}(|M|) \to \mathsf{Seq}^* \cup \{\infty\}$$

where I^M maps $N \in \mathsf{subterm}(|M|)$ to a non-empty sequence corresponding to the (nested) labels ascribed to N. We adopt the convention that if a subterm $N \in \mathsf{subterm}(|M|)$ has no labels then $I^M(N) = \infty$. An example should clarify the definition. Take

$$M = [(((\lambda x.x)^2(y^3)^5)^1 z)^4]^5.$$

Then,

$$\begin{aligned} I^M(|M|) &= \langle 4,5\rangle, \\ I^M(\lambda x.x) &= \langle 2\rangle, \\ I^M(z) &= \infty. \end{aligned}$$

We will sometimes write M as $(((\lambda x.x)^{\langle 2\rangle} y^{\langle 3,5\rangle})^{\langle 1\rangle} z^{\langle\infty\rangle})^{\langle 4,5\rangle}$. A labelled λ-term $M \in \Lambda^\omega\bot$ is *completely-labelled* if every subterm of $|M|$ is labelled. More precisely, a completely-labelled λ-term is a pair (M, I) where

$$\mathsf{range}(I^M) \subseteq \mathsf{Seq}^*.$$

We abbreviate "I is a complete labelling of M" as $I \in \mathcal{CL}(M)$.

(ii) Define on $\Lambda^\omega\bot$ the following notions of reduction:

$$\begin{aligned} &(\mathbf{lab}) && (M^m)^n \to M^{\min(m,n)}, \\ &(\beta_+) && (\lambda x.M)^{n+1}N \to (M[x := N^n])^n, \\ &(\beta_0) && M^0 \to \bot, \\ &(\bot) && \bot M \to \bot, \\ &(\bot_{\mathbf{n}}) && \bot^n \to \bot \equiv \bot^0. \end{aligned}$$

We define $\mathbf{llab} \equiv \mathbf{lab} \cup \beta_+ \cup \beta_0 \cup \bot \cup \bot_{\mathbf{n}}$. (**llab** stands for "lazy labelled" reduction.)
Our definition of **llab** differs from **lab**.β in [Bar84, §14.3.1 page 364] in that the notion of reduction: $\lambda x.\bot \to \bot$ is not admitted in our version; also β_0 is different from $\beta_\bot$ in *op. cit.* This should come as no surprise in our *lazy regime*, since the *leitmotif* of our work is

Abstraction legitimizes its function body.

When there is a need to compare and contrast the two slightly different labelled reductions, we will distinguish our reduction from that in [Bar84] by qualifying the latter as "strict".

(iii) A *lazy labelled reduction strategy*:
First, we define a map $\min : \mathsf{Seq}^* \cup \{\infty\} \to \omega \cup \{\infty\}$ by

$$\min(\vec{l}) = \begin{cases} \text{minimun of } \vec{l} & \text{if } \vec{l} \in \mathsf{Seq}^*; \\ \infty & \text{if } \vec{l} = \{\infty\}. \end{cases}$$

We define a notion of reduction on $\Lambda^\omega\bot$ by the following proof system:

$$\frac{}{((\lambda x.P)^{\vec{l}_1}Q)^{\vec{l}_2} \to_\omega ((P[x := Q^n])^n)^{\vec{l}_2}},$$

provided $\min(\vec{l_1}) = n+1$ or ∞, in which case $n = \infty$;

$$\frac{M \to_\omega M'}{(MN)^{\vec{l}} \to_\omega (M'N)^{\vec{l}}};$$

where M, M', P and Q range over $\Lambda^\omega\bot$ and $\vec{l}$ over $\mathsf{Seq}^* \cup \{\infty\}$. In addition, we introduce the following as per normal:

$$\begin{array}{lcl} \twoheadrightarrow_\omega & \stackrel{\text{def}}{=} & \text{the reflexive, transitive closure of } \to_\omega, \\ M{\uparrow_\omega} & \stackrel{\text{def}}{=} & \exists\{M_n : n \in \omega\} \;\&\; \forall n.M_n \to_\omega M_{n+1}, \\ M \not\to_\omega & \stackrel{\text{def}}{=} & M \notin \mathsf{dom}(\to_\omega), \\ \mathsf{nf}(\to_\omega) & \stackrel{\text{def}}{=} & \{M \in \Lambda^\omega\bot : M \not\to_\omega\}, \\ M \downarrow_\omega N & \stackrel{\text{def}}{=} & M \twoheadrightarrow_\omega N \;\&\; N \not\to_\omega. \end{array}$$

Clearly, $\downarrow_\omega$ is a partial function; it defines a *deterministic* reduction strategy.

(iv) The denotation of a labelled $\lambda\bot$-terms in M, a lazy λ-model with lazy approximable application, is given below by customary structural induction:

$$\begin{array}{lcl} [\![(\bot, I)]\!]_\rho & \stackrel{\text{def}}{=} & \bot \\ [\![(x, I)]\!]_\rho & \stackrel{\text{def}}{=} & ([\![x]\!]_\rho)_{\min(I(x))} \\ [\![(MN, I)]\!]_\rho & \stackrel{\text{def}}{=} & ([\![(M, I)]\!]_\rho \cdot [\![(N, I)]\!]_\rho)_{\min(I(MN))} \\ [\![(\lambda x.M, I)]\!]_\rho & \stackrel{\text{def}}{=} & (\mathsf{Gr}(d \mapsto [\![(M, I)]\!]_{\rho[x:=\underline{d}]}))_{\min(I(\lambda x.M))} \end{array}$$

REMARK **3.2.3.5** (i) It should be clear that $\mathsf{nf}(\to_\omega)$ consists precisely of the following mutually exclusive syntactic classes:
N.B. In the following some indices are left out.

(1) $(\lambda x.P)^{\vec{l}}$,

(2) $(\cdots(x^{\vec{l}}M_1)^{\vec{l}_1}\cdots M_n)^{\vec{l}_n}$, $(\cdots(\bot^{\vec{l}}M_1)^{\vec{l}_1}\cdots M_n)^{\vec{l}_n}$,

(3) $(\cdots(((\lambda x.P)^{\vec{l}}Q)^{\vec{l}'}L_1)^{\vec{l}_1}\cdots L_n)^{\vec{l}_n}$ such that $\min(\vec{l}) = 0$,

where P, Q, L_i and M_i range over $\Lambda^\omega\bot$ and $n \geqslant 0$.

(ii) It should be straightforward to see that the reduction $\rightarrow_\omega$ is just the "lifted" version of the lazy reduction $\rightarrow_l$ in the following sense:

$$\forall M, N \in \Lambda^\omega.M \twoheadrightarrow_\omega N \Rightarrow |M| \twoheadrightarrow_l |N|.$$

Also, $\rightarrow_\omega$ is a *conservative* extension of $\rightarrow_l$, *i.e.*

$$\forall M, N \in \Lambda.M \rightarrow_l N \iff M \rightarrow_\omega N.$$

(iii) There is really no substantive difference between Λ^ω and $\Lambda^\omega\bot$ since to all intents and purposes all terms with outermost label 0 have the same behaviour as $\bot^n \equiv \bot$.

Recall that a reduction $\mathbf{R}$ on Λ is *strongly normalizing* if $\forall M \in \Lambda$, there is no infinite $\mathbf{R}$-reduction path starting from M.

PROPOSITION **3.2.3.6 (Strong Normalization)** *Let $M \in \Lambda^\omega\bot$ be completely labelled. Then, every* llab*-reduction starting from M terminates.*

PROOF This is an immediate corollary of the Strong Normalization of the "strict" lab.β reduction. See [Bar84, Theorem 14.1.12, page 358] for a proof. □

COROLLARY **3.2.3.7** *The reduction $\rightarrow_\omega$ restricted to completely-labelled terms is* Noetherian, *or terminating. That is to say, for $M \in \Lambda$,*

$$\forall I \in \mathcal{CL}(M).\exists N.(M, I) \downarrow_\omega N \in \mathsf{nf}(\rightarrow_\omega)$$

PROOF This is immediate from the observation that $\twoheadrightarrow_\omega$ specifies a reduction strategy in **llab**. □

COROLLARY **3.2.3.8** *Let $M \in \Lambda$ such that M is strongly unsolvable. Then, for all complete labelling I,*

$$(M, I) \downarrow_\omega (\lambda x.P)^0 Q\vec{L}. \quad \textit{(some indices are left out)}$$

PROOF Since I is a complete labelling, so by Corollary 3.2.3.7,

$$\exists N \in \Lambda^\omega.(M, I) \downarrow_\omega N.$$

But, by Remark 3.2.3.5(ii), we have $M \downarrow_l |N|$. By the Operational Characterization of Strong Unsolvability, *i.e.* $L \in \mathbf{PO}_0 \iff L\uparrow_1$, we deduce that $|N|$ is in $l\beta$rf. The result then follows by reference to Remark 3.2.3.5(i). □

LEMMA **3.2.3.9** *Let D be given as in the Proposition. Then, for any $M \in \Lambda$,*

$$D \models M = \bigsqcup\{ (M, I) : I \in \mathcal{CL}(M) \}.$$

PROOF Let $(M, I) \in \Lambda^\omega$ be a completely-labelled term. We prove by structural induction on M. This is clearly valid for $M \equiv x$. Suppose $M \equiv PQ$. First note that

$$\{ [\![(P, I_i^P)]\!]_\rho : I_i^P \in \mathcal{CL}(P) \} \text{ is directed.}$$

Then, by Axiom (1) of lazy approximable application and the continuity of "$\cdot$", we have

$$\begin{aligned}
[\![M]\!]_\rho &= \sqcup_{n\in\omega}([\![PQ]\!]_\rho)_n \\
&= \sqcup_{n\in\omega}([\![P]\!]_\rho \cdot [\![Q]\!]_\rho)_n && \text{Ind. Hyp.} \\
&= \sqcup_{n\in\omega}(\sqcup_{I_i^P \in \mathcal{CL}(P)} [\![(P, I_i^P)]\!]_\rho \cdot \sqcup_{I_j^Q \in \mathcal{CL}(Q)} [\![(Q, I_j^Q)]\!]_\rho)_n \\
&= \sqcup_{n\in\omega} \sqcup([\![(P, I_i^P)]\!]_\rho \cdot [\![(Q, I_j^Q)]\!]_\rho)_n \\
&= \sqcup_{I\in\mathcal{CL}(M)} [\![(M, I)]\!]_\rho.
\end{aligned}$$

The last two steps are justified by the continuity of $(-)_n$ and by definition of $[\![(PQ, I)]\!]$ respectively.

The inductive step for the case $M \equiv \lambda x.P$ is similar and we need to use the continuity of Gr. The details are omitted. □

LEMMA **3.2.3.10** *Let D be given as in the Proposition. Let $M, N \in \Lambda^\omega$, then*

$$M \to_\omega N \Rightarrow D \models M \sqsubseteq N.$$

PROOF This is immediate from the axioms of lazy approximable application. □

Now, we are in a position to prove the Proposition.

PROOF Let $M\Uparrow$. Then, by Lemma 3.2.3.9:

$$\begin{aligned}
D \models M &= \sqcup\{ (M, I) : I \in \mathcal{CL}(M) \} && \text{Corollary 3.2.3.8 and} \\
& && \text{Lemma 3.2.3.10} \\
&\sqsubseteq \sqcup\{ ((\lambda x.P)Q\vec{L}, J) : J(\lambda x.P) = 0 \} \\
&= \bot.
\end{aligned}$$

□

3.3 Plotkin-Scott-Engeler Algebras and Models

In this section, we introduce a family of combinatory algebras called *Plotkin-Scott-Engeler (PSE) Algebras* after G. Longo in [Lon83]. PSE-Algebras are defined (and some can be constructed inductively from a base set) in a very natural set-theoretic way and may be seen as a generalization of earlier ideas in [Plo72,Sco76]. The notion of application introduced here to interpret the formal application of the λ-terms is reminiscent of the classical Myhill-Shepherdson definition of application and the enumeration operators in $\wp\omega$. A PSE-algebra can be expanded into a λ-model, which we shall call a *PSE-model*, by stipulating a *graph* function which serves as a choice function in the sense described earlier.

3.3.1 Motivation and Comparison with $\wp\omega$

DEFINITION **3.3.1.1** Let $\langle D, \cdot \rangle$ be a combinatory complete applicative structure. $\langle D, \cdot \rangle$ is a *categorical* λ-model (λ-algebra, combinatory algebra) if it has a unique *expansion*, $\langle D, \cdot, k, s \rangle$, turning D into a λ-model (λ-algebra, combinatory algebra respectively).

We will show in this section that the PSE-models D_A is a family of particularly "*soft*"[6] λ-models (as opposed to *hard*). Essentially, this means that D_A has many more elements than those which are denotations of closed λ-terms.

DEFINITION **3.3.1.2 (Plotkin, Scott, Engeler and Longo)** Let B be a non-empty set such that

$$\text{(io-pair)} \qquad \beta \subseteq_{\mathrm{f}} B \;\&\; b \in B \iff (\beta; b) \in B.$$

where β (similarly for $\beta_0, \beta_1, \cdots, \gamma_i, \cdots$) ranges over *finite* sets. We define an applicative structure $\langle \wp B, \cdot \rangle$, known as the *Plotkin-Scott-Engeler (PSE) Algebra*, where the application "$\cdot$":$\wp B \times \wp B \to \wp B$ is defined as

$$d \cdot e \stackrel{\text{def}}{=} \{\, b : (\beta; b) \in d \;\&\; \beta \subseteq_{\mathrm{f}} e \,\}.$$

The reader will note the resemblance of the above notion of application with that in the λ-model $\wp\omega$. Recall that in $\wp\omega$, the following codings provide the essential translation between the *operator-* and *operand roles* assumed by each element in $\wp\omega$:

- **bijective coding of ordered pairs:** $(-,-) : \omega \times \omega \to \omega$

[6] Let $\mathcal{M}$ be a λ-model. Define $\mathcal{M}^o$ as the sub-structure of $\mathcal{M}$ consisting of all λ-definable elements. $\mathcal{M}$ is *hard* if $\mathcal{M} = \mathcal{M}^o$.

$$(n,m) \stackrel{\text{def}}{=} (1/2)(m+1)(n+m+1)+m$$

- **bijective coding of finite sets:** $e_{\_} : \omega \to \{\, x \in \wp\omega : x \text{ finite} \,\}$

  $$e_n \stackrel{\text{def}}{=} \{\, k_0, \cdots, k_{m-1} \,\}$$

  defined by $n = \sum_{i<m} 2^{k_i}$ such that $k_0 < k_1 < \cdots < k_{m-1}$.

Application is defined as

$$u \cdot x \stackrel{\text{def}}{=} \{\, m : e_n \subseteq_{\mathsf{f}} x \;\&\; (n,m) \in u \,\}.$$

Observe that each atomic component $d = (n,m) \in u \in \wp\omega$ relates some "input" n to some "output" m. The application $u \cdot x$ may be read as follows:

> If the finite set encoded by the "input" component n approximates the operand x *i.e.* $e_n \subseteq_{\mathsf{f}} x$, then the result $u \cdot x$ is approximated by the "output" component m.

The analogy between the PSE-algebra D and $\wp\omega$ should be clear and has pedagogical value: each element of $D \equiv \wp B$ is a *set* of "input-output" pairs of the form $(\beta; b)$. A pleasing feature of the PSE-algebra is that no encoding is necessary owing to the explicit presentation of the elements of B as ordered-pairs which is deliberately suggestive of the "input-output" correspondence. PSE-algebras are *homomorphic generalizations* of the $\wp\omega$ model, as shown by Longo and Bruce.

FACT **3.3.1.3 (Longo,Bruce)** *(i) $\wp\omega$ is a categorical λ-model whereas the (well-founded) PSE-algebra D_A generated by a ground set A (see Definition 3.3.1.6) is not.*

(ii) The (well-founded) PSE-algebra $\langle D_A, \cdot \rangle$ generated by a ground set A and $\langle \wp\omega, \cdot \rangle$ can be homomorphically embedded one into the other; but for no *A are they isomorphic as applicative structures.*

PROOF See [Lon83] and [BL84]. □

Let D, E be topological spaces. Denote $[D \to E]$ as the set of continuous functions from D to E.

LEMMA **3.3.1.4** *Let B be a non-empty set satisfying (io-pair). Then*

(i) $\langle \wp B, \cdot \rangle$ is a complete algebraic lattice. By elementary domain theory, the Scott topology on $\wp B$ is given by the basis

$$\{\, d \in \wp B : \beta \subseteq_{\mathsf{f}} d \,\}.$$

Also, $f \in [\wp B^n \to \wp B]$ *iff* f *is continuous argument-wise iff*

$$f(\vec{d}) = \bigcup\{ f(\vec{\beta}) : \vec{\beta} \subseteq_f \vec{d} \}.$$

(ii) **Coincidence of representable and continuous function space:**

$$[\wp B^n \to_r \wp B] = [\wp B^n \to \wp B].$$

PROOF

(i) Routine.

(ii) The direction "$\subseteq$" follows immediately from the continuity of "$\cdot$" which is an easy exercise. More interesting is the other direction. Observe that

$$\{ (\beta_1; (\beta_2, \cdots (\beta_n; b) \cdots)) : \vec{\beta} \subseteq_f B.b \in f(\beta_1, \cdots, \beta_n) \}$$

represents $f \in [\wp B^n \to \wp B]$.

□

It should be clear that any PSE-algebra D is a combinatory algebra. In what follows, we will show that there are two canonical ways to expand D into a λ-model. λ-expansion corresponds to the specification of the *graph* function — Gr, which selects a *unique* representative $\mathsf{Gr}(f)$ for each representable function f. The first canonical λ-expansion, which we will refer to as *strict* (for reasons to be explained in the sequel), is presented as follows:

PROPOSITION **3.3.1.5** *Let* $\langle \wp B, \cdot \rangle$ *be a PSE-algebra. Define* $\mathsf{Gr}_\perp : [\wp B \to \wp B] \to \wp B$ *by*

$$\mathsf{Gr}_\perp(f) \stackrel{\text{def}}{=} \{ (\beta; b) : \beta \subseteq_f B \ \& \ b \in f(\beta) \}.$$

Then $\langle \wp B, \mathsf{Fun}, \mathsf{Gr} \rangle$ *is a* λ*-model, which we will call the (canonical)* **strict PSE-model.** *The subscript "*$\perp$*" attests to the fact that* $\mathsf{Gr}_\perp$ *selects the* least *representative from* $\mathsf{rep}(f)$ *for each* f*, as we will show later.*

PROOF It should be clear that $\langle \wp B, \mathsf{Fun}, \mathsf{Gr} \rangle$ is a functional domain. Continuity of "$\cdot$" and the coincidence of representable and continuous function space implies that $\wp B$ is combinatory complete. Continuity of $\mathsf{Gr}_\perp$ (easy) implies that

$$\mathsf{Gr}_\perp \circ \mathsf{Fun} \in [\wp B \to \wp B] = [\wp B \to_r \wp B].$$

By Lemma 3.1.3.1, $\langle \wp B, \mathsf{Fun}, \mathsf{Gr}_\perp \rangle$ is a λ-model. □

Free PSE-Algebras

For the rest of our study on the PSE-algebras, we will focus exclusively on the class of *free* PSE-algebras generated by a set A, D_A. Freely generated PSE-models are *well-founded* (in a sense to be made precise later) and their *constructive* nature is amenable to investigations of their *local structure.*[7]

DEFINITION **3.3.1.6** (Longo) Let $A \neq \emptyset$. Define successively a family of sets $\{ B_n : n \in \omega \}$ as follows:

$$B_0 \stackrel{\text{def}}{=} A, \quad B_{n+1} \stackrel{\text{def}}{=} B_n \cup \{ (\beta; b) : \beta \subseteq_{\mathsf{f}} B_n \,\&\, b \in B_n \},$$
$$B \stackrel{\text{def}}{=} \bigcup B_n, \quad D_A \stackrel{\text{def}}{=} \wp B.$$

Recall that throughout our work, β ranges over *finite* subsets. We call A the set of *atomic* information and decree that no element of A shall have the "input-output" form: $(\beta; b)$. Intuitively, this constraint is tantamount to saying that no element of A is capable of assuming an operator role. We call D_A the *PSE-algebra freely generated by A.*

Given any continuous function $f \in [D_A \to D_A]$, it should not be difficult to see that the *extensionality class* of f includes the following subset of $\wp B$:

$$\mathsf{rep}_\lambda(f) \stackrel{\text{def}}{=} \{ G_\perp(f) \cup X : X \in \wp A \}.$$

Recall that $G_\perp(f) = \{ (\beta; b) : b \in f(\beta) \}$; and we define $\mathsf{Gr}_X(f) \stackrel{\text{def}}{=} G_\perp(f) \cup X$ for $X \subseteq A$. Observe that $\mathsf{rep}_\lambda(f)$ forms a complete lattice whose structure is isomorphic to $\langle \wp A, \subseteq \rangle$. Each graph function Gr_X parametrized by an element $X \in \wp A$ determines a *λ-expansion*, $\langle D_A, \mathsf{Fun}, \mathsf{Gr}_X \rangle$, of the underlying PSE-algebra. That $\langle D_A, \mathsf{Fun}, \mathsf{Gr}_X \rangle$ is indeed a λ-model follows from the same argument in the proof of Proposition 3.3.1.5. Clearly, there are at least 2^α different λ-expansions of D_A, where α is the cardinality of A.

---

[7] Barendregt in his treatise on Lambda Calculus [Bar84] distinguishes between *local* and *global* structures of λ-models. The *local structure* of a λ-model $\mathcal{M}$ is

$$Th(\mathcal{M}) = \{ M = N : \mathcal{M} \models M = N. M, N \in \Lambda \}.$$

If $\mathcal{M}$ is a cpo, then it is also interesting to characterize the ordering between denotations of λ-terms by means of a syntactic preorder R. That is, find a syntactic characterization $R \subseteq \Lambda \times \Lambda$ such that

$$R \stackrel{\text{def}}{=} \{ (M, N) : \mathcal{M} \models M \sqsubseteq N \}.$$

The *global structure* of a model $\mathcal{M}$ consists of properties true in $\mathcal{M}$ other than equations (not necessarily first order), for example, extensionality, satisfaction of the ω-rule, *hardness* or *richness.*

Saturatedness

Each element of the set $\mathsf{rep}_\lambda(f)$ satisfies the property of *saturatedness*. Formally, we say that an element x of $\wp B$ is *saturated* iff

$$(\beta; b) \in x \Rightarrow [\forall \text{ finite } \gamma \supseteq \beta.(\gamma; b) \in x].$$

Saturatedness is a natural condition which is rather reminiscent of monotonicity in the logic of entailment. We may read it as: if it is the case that any input approximated by β yields an output approximated by b; then the output information should at least be preserved if there is a gain in the input information.

Following Longo, we introduce the following notation

NOTATION **3.3.1.7** Given $P \subseteq D_A^n \equiv \underbrace{D_A \times \cdots \times D_A}_{n}, n \geqslant 1$. We say that $\langle \beta_1, \cdots, \beta_n \rangle$ are *least such* $P(\beta_1, \cdots, \beta_n)$ which we abbreviate as

$$\langle \beta_1, \cdots, \beta_n \rangle \text{ least } P$$

iff

- $P(\beta_1, \cdots, \beta_n)$ holds, and
- If $\forall i.\gamma_i \subseteq_{\mathsf{f}} \beta_i$ but $\vec{\gamma} \neq \vec{\beta}$, then $\neg P(\gamma_1, \cdots, \gamma_n)$.

Perhaps somewhat surprisingly, not every element in the extensionality class of f, $\mathsf{rep}(f)$, is saturated. Consider the set, X_f, parametrized by $f \in [D_A \to D_A]$, as follows:

$$X_f \stackrel{\text{def}}{=} \{ (\beta; b) : \beta \text{ least } (b \in f(\beta)) \}.$$

Clearly, X_f is not saturated and that it is a representative of f. Hence, we can conclude that

$$\mathsf{rep}_\lambda(f) \subset \mathsf{rep}(f)$$

i.e. the inclusion is strict. In fact, we can say more:

THEOREM **3.3.1.8 (Longo)** *Suppose* $\langle D_A, \mathsf{Fun}, \mathsf{Gr} \rangle$ *is a* λ*-expansion of the free PSE-algebra* $\langle D_A, \cdot \rangle$. *Then* $\forall f \in [D_A \to D_A]$, $\mathsf{Gr}(f)$ *is saturated.*

PROOF See proof of Lemma 4.3 (Main Structural Lemma) in [Lon83]. □

The other canonical expansion of a PSE-algebra which we call *lazy PSE-model* is defined as follows:

DEFINITION **3.3.1.9** We define the (free) *lazy PSE-model* to be $\langle D_A, \mathsf{Fun}, \mathsf{Gr}_\top \rangle$ where $\mathsf{Gr}_\top$ is defined to be:

$$\mathsf{Gr}_\top(f) = \mathsf{Gr}_\bot(f) \cup A$$

i.e. for every representable function f, $\mathsf{Gr}_\top$ picks the *largest* (it should be easy to see) representative $\mathsf{Gr}_\top(f)$ from the extensionality class of f.

NOTATION **3.3.1.10** We denote the semantic function of the lazy PSE-model as $[\![\text{-}]\!]^\top$ and that of the strict PSE-model as $[\![\text{-}]\!]^\bot$. For the rest of the Chapter, however, unless otherwise specified, $[\![\text{-}]\!]$ shall mean the semantic function in lazy free PSE-models.

3.3.2 Semantic Characterization of Unsolvability in Lazy PSE models

In this section, we will examine the semantic properties of *lazy PSE-models freely generated by a set A* vis-á-vis the denotation of λ-terms. The results in this subsection were first obtained by G. Longo in [Lon83]. These results will be needed in the proof of the Local Structure Theorem which is the main result of this Chapter. The (free) lazy PSE-models provide a semantic classification of the class of *unsolvable* λ-terms. In short, it is a *fully lazy λ-model, i.e.*

$$\forall m, n \in \omega + 1. \forall M \in \mathbf{PO}_{\mathrm{m}}. \forall N \in \mathbf{PO}_{\mathrm{n}}. [\mathcal{M} \models M = N \iff m = n].$$

Another example of an fully lazy λ-model is the initial solution of the domain equation:

$$D \cong [D \to D]_\bot;$$

whose semantic properties are studied in Chapter 4.

DEFINITION **3.3.2.1** Define $A_{(n)} \stackrel{\text{def}}{=} [\![\lambda x_1 \cdots x_n.\emptyset]\!]$.

LEMMA **3.3.2.2** *(i)* $A_{(0)} = \emptyset; A_{(1)} = A$.

(ii) For all $n \in \omega$:

$$\begin{aligned} A_{(n+1)} &= [\![\lambda x.A_{(n)}]\!] \\ &= \mathsf{Gr}_\top(d \mapsto A_{(n)}) \\ &= [\![\lambda x_1 \cdots x_n.A]\!]^\bot \cup [\![\lambda x_1 \cdots x_{n-1}.A]\!]^\bot \cup \cdots \cup A. \end{aligned}$$

Note that $[\![\lambda x_1 \cdots x_n.A]\!]^{\perp} = \{ (\beta_{n-1}; \cdots (\beta_1; a) \cdots) : a \in A \& \beta_i \subseteq_{\mathsf{f}} B \}$. *Consequently, we have* $\forall n \in \omega . A_{(n)} \subseteq A_{(n+1)}$.

(iii) $\forall d \in D_A . A_{(n+1)} \cdot d = A_{(n)}$.

(iv) $[\![\lambda x_1 \cdots x_n.M]\!]^{\top} = [\![\lambda x_1 \cdots x_n.M]\!]^{\perp} \cup A_{(n)}$.

PROOF Easy induction. □

Observe that

$$\forall b \in B . \exists \beta_n, \cdots, \beta_1 \subseteq_{\mathsf{f}} B . \exists a \in A . b = (\beta_n; \cdots; (\beta_1; a) \cdots).$$

This is a property of the underlying PSE-algebra which we call its *well-founded property.* Crucially, this property is immediately seen to hold true for all *freely generated* PSE-algebras. Well-foundedness fails for an interesting class of PSE-algebras studied by Longo which is obtained by quotienting the free PSE-algebra over appropriate equivalence relations $\sim$. Clearly, not every equivalence relation on B will turn $\wp\widetilde{B}$ into a non-trivial applicative structure. Two such non-trivial examples are obtained by forcing $a = (\{a\}; a)$ and $a = (\emptyset; a)$ for all $a \in A$ respectively.

PROPOSITION **3.3.2.3** *(i) The (free) lazy PSE-model* $\mathcal{M} \equiv \langle D_A, \mathsf{Fun}, \mathsf{Gr}_{\top} \rangle$ *is a lazy* λ*-model which has lazy approximable application.*

(ii) It is the unique λ*-expansion of the free PSE-algebra* D_A *that satisfies*

$$(\eta^-) \qquad \forall d \in D_A . d \subseteq [\![\lambda x.dx]\!].$$

PROOF

(i) $\langle D_A, \cdot, \Uparrow \rangle$ with $\Uparrow = \{ \emptyset \}$ is easily seen to be a quasi-applicative structure with divergence. By definition, $\mathsf{range}(\mathsf{Gr}_{\top}) \cap \Uparrow = \emptyset$. That $\mathcal{M}$ is a lazy λ-model then follows form the observation that it is a λ-model. Now, we show that $\mathcal{M}$ has lazy approximable application. Following Longo, we define a valuation function $|-|$ with range in ω *simultaneously* on the elements as well as *finite* subsets of B as follows:

$$|b| \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } b \in A, \\ |\beta| + |c| & \text{if } b = (\beta; c); \end{cases}$$

$$|\beta| = \max\{ |c| : c \in \beta \} + 1.$$

We can now define the approximation function $\mathsf{Approx} : D_A \times \omega \to D_A$. For $d \in D_A$, define $d_n = \{ b \in d : |b| \leqslant n \}$. Clearly, $|\beta|, |b| < |(\beta; b)|$. We will only check axiom (4) of Definition 3.2.3.2, the rest is straightforward. Observe that

$$\begin{aligned} d_{n+1}e &= \{\, b : \exists \beta \subseteq_{\mathsf{f}} e.(\beta; b) \in d \;\&\; |\beta| + |b| \leqslant n+1 \,\} \\ &\subseteq \{\, b : \exists \beta \subseteq_{\mathsf{f}} e_n.(\beta; b) \in d \;\&\; |b| \leqslant n \,\} \\ &= (de_n)_n. \end{aligned}$$

since $\forall \beta, \forall b.|\beta|, |b| \geqslant 1$ and $|\beta| \leqslant n$ implies $(\beta \subseteq_{\mathsf{f}} e \Rightarrow \beta \subseteq_{\mathsf{f}} e_n)$.

(ii) Let $d \in D_A$. Wlog, say, $d{\Downarrow}$. Choose $f \in [D_A \to D_A]$ and $d \in \mathsf{rep}(f)$ such that $e \stackrel{\mathsf{def}}{=} \bigsqcup \mathsf{rep}(f)$. Now, $[\![\lambda x.\underline{d}x]\!]^{\top} = \mathsf{Gr}_{\top}(\mathsf{Fun}(d))$. Recall that Fun maps any element d to the *unique* function (in this case f) that has d as a representative, *i.e.* $\mathsf{Fun}d = f$. Since $\mathsf{Gr}_{\top}$ picks the *largest* representative from $\mathsf{rep}(f)$ for *every* representable function f. Hence,

$$d \subseteq \bigsqcup \mathsf{rep}(f) = \mathsf{Gr}_{\top}(f) = [\![\lambda x.\underline{d}x]\!]^{\top}.$$

□

NOTATION **3.3.2.4** From now on, we will denote the free lazy PSE-model generated by A as $D_A^{\top}$.

COROLLARY **3.3.2.5** *The graph functions corresponding to the strict and lazy (free) PSE-models are the least and the largest λ-expansions of the PSE-algebra $\langle D_A, \cdot \rangle$ respectively. More precisely, for any λ-expansion $\langle D_A, \mathsf{Fun}, \mathsf{Gr} \rangle$,*

$$\forall f \in [D_A \to D_A].\mathsf{Gr}_{\bot}(f) \subseteq \mathsf{Gr}(f) \subseteq \mathsf{Gr}_{\top}(f).$$

PROOF An easy corollary of Theorem 3.3.1.8. □

COROLLARY **3.3.2.6 (Adequacy)** *Let M be a λ-term. Then,*

$$M{\uparrow_1} \iff M \in \mathbf{PO}_0 \iff D_A^{\top} \models M{\Uparrow}.$$

PROOF This follows directly from the previous Proposition and Proposition 3.2.3.1. □

The following result says that $D_A^{\top}$ is a *fully lazy λ-model.*

THEOREM **3.3.2.7 (Longo)** *Let M be a λ-term. Then,*

(i) $\forall n \in \omega.N \in \mathbf{PO}_n \iff [\![N]\!]_{\rho} = A_{(n)}$.

(ii) $M \in \mathbf{O}_n \iff [D_A^{\top} \models M \supseteq A_{(m)} \Leftrightarrow m \leqslant n]$.

(iii) $M \in \mathbf{O}_{\infty} \iff D_A^{\top} \models M = B = \top$.

PROOF See [Lon83, Theorem 3.7, page 166]. □

COROLLARY **3.3.2.8** *Let M be a λ-term. Then,*

$$M \text{ is unsolvable} \iff \exists! n \in \omega + 1 . D_A^{\top} \models M = A_{(n)};$$

and n is precisely the order of unsolvability of M. □

DEFINITION **3.3.2.9** Let $f \in [D_A^n \to D_A]$ where $n \in \omega$. Define $[\![\lambda x_1 \cdots x_n . f(x_1, \cdots, x_n)]\!]^-$ as

$$\{ (\beta_1; \cdots (\beta_n; c) \cdots) \in [\![\lambda x_1 \cdots x_n . f(x_1, \cdots, x_n)]\!]^{\perp} : . \vec{\beta} \text{ least } (c \in f(\vec{\beta})) \}.$$

REMARK **3.3.2.10** We remark that any $b \in [\![\lambda x_1 \cdots x_n . f(x_1, \cdots, x_n)]\!]^-$ has the form

$$(\beta_1; \cdots (\beta_m; a) \cdots) \text{ where } a \in A \text{ and } m \geqslant n.$$

REMARK **3.3.2.11 (A Minor Technical Problem in [Lon83])**
This might be an appropriate place to point out a minor error on page 186, in Appendix B of *op. cit.*

According to Definition B.5, $F_n^- \equiv \lambda^- x_1 \cdots x_n . f(x_1, \cdots, x_n)$ consists precisely of elements b of the shape

$$(\beta_1; \cdots; (\beta_n; c) \cdots)$$

such that $\vec{\beta}$ least $c \in f(\vec{\beta})$. It follows that no elements of the form

$$(\beta_1; \cdots; (\beta_m; a_o) \cdots) \quad \text{where} \quad m < n$$

can possibly belong to F_n^-.

Therefore, for any $d_1, \cdots, d_p$ where $p < n$

$$F_n^- d_1 \cdots d_p = \{ c : \vec{\beta} \subseteq \vec{d} \,\&\, (\beta_1; \cdots; (\beta_p; c) \cdots) \in F_n^- \}.$$

Clearly, $a_o \notin F_n^- d_1 \cdots d_p$.

However, the first two lines of the proof of Lemma B.6(2) assert the opposite. If one redefines F_n^- as the *union* of the old value of F_n^- and

$$\{ a_o \} \cup \{ (\emptyset; a_o) \} \cup \cdots \cup \{ \underbrace{(\emptyset; \cdots; (\emptyset}_{n-1}; a_o) \cdots) \},$$

then the proof remains valid. □

Let a_0 be an arbitrary element of A for the rest of the Chapter.

PROPOSITION **3.3.2.12** *Let $f \in [D_A \to D_A]$. Then,*

(i) $[\![\lambda x_1 \cdots x_n.f(x_1,\cdots,x_n)]\!]^- d_1 \cdots d_n = f(d_1,\cdots,d_n)$.

(ii) Suppose

$$(\neg\emptyset) \qquad \forall d_1,\cdots,d_{n-1}.\exists d_n.f(d_1,\cdots,d_{n-1},d_n) \neq \emptyset.$$

We have: if $0 \leqslant p < n$, then $[\![\lambda x_1 \cdots x_n.f(x_1,\cdots,x_n)]\!]^- d_1 \cdots d_p$ does not contain a_0 and it is not saturated.

PROOF (i) follows immediately from the definition. (ii) Let

$$F \stackrel{\text{def}}{=} [\![\lambda x_1 \cdots x_n.f(x_1,\cdots x_n)]\!]^-.$$

Then, any

$$b \in [\![\lambda x_1 \cdots x_n.f(x_1,\cdots,x_n)]\!]^- d_1 \cdots d_p$$

is such that $\exists \vec{\beta} \subseteq_{\mathrm{f}} \vec{d}.(\beta_1;\cdots;(\beta_p;b)\cdots) \in F$. By Remark 3.3.2.10, b cannot belong to A, because $p < n$ implies that b must be of the form $(\beta;b')$. As for non-saturation, observe that

$$\begin{aligned} f(e_1,\cdots,e_n) &= Fe_1\cdots e_n \\ &= \{b : \vec{\beta} \subseteq_{\mathrm{f}} \vec{e}\ \&\ (\beta_1;\cdots(\beta_n;b)\cdots) \in F\}. \end{aligned}$$

Consequently, by the assumption $(\neg\emptyset)$ on f,

$$\forall p < n.\forall d_1,\cdots,d_p.\exists \beta_1 \subseteq_{\mathrm{f}} d_1,\cdots,\exists \beta_p \subseteq_{\mathrm{f}} d_p,$$

$$\exists \beta_{p+1},\cdots,\exists \beta_n,\exists b.(\beta_1;\cdots(\beta_n;b)\cdots) \in F.$$

Crucially, recall that by definition of F, these $\vec{\beta}$ are least such, thus in particular,

$$\forall \gamma \supset \beta_{p+1}.(\gamma;(\beta_{p+2};\cdots(\beta_n;b)\cdots)) \notin Fd_1\cdots d_p,$$

and we are done. □

DEFINITION **3.3.2.13** For $m \geqslant 2$, we define *Böhm permutators* of degree m as follows:

$$\mathbf{P}_m \stackrel{\text{def}}{=} \lambda x_0 \cdots x_m.x_m x_o \cdots x_{m-1}.$$

These permutators play pivotal roles in the so-called "Böhm-out" technique.

For $m > 2$, we define semantic Böhm permutators, p_m as follows:

$$p_m \stackrel{\text{def}}{=} [\![\lambda x_0 \cdots x_m.x_m x_0 \cdots x_{m-1}]\!]^-.$$

COROLLARY **3.3.2.14 (Böhm Permutators)** *Let x_i, d_i range over D_A. Then,*

(i) $D_A^{\mathsf{T}} \vDash p_m x_0 \cdots x_m = x_m x_o \cdots x_{m-1}$.

(ii) If $l \leqslant m$, then $p_m d_1 \cdots d_l$ is not saturated and does not contain a_0.

PROOF We note that $p_m \stackrel{\text{def}}{=} [\![\lambda x_0 \cdots x_m.x_m x_0 \cdots x_{m-1}]\!]^-$ satisfies the condition $(\neg\emptyset)$ in the Proposition. □

3.4 Local Structure of Free Lazy PSE-Models

In this section, we introduce a new operational preorder $\lesssim$ which we shall call the *lazy Plotkin-Scott-Engeler preorder*. The least λ-terms w.r.t $\lesssim$ are the strongly unsolvables; and the *top* elements are the unsolvables of infinite order. $\lesssim$ respects the η^--axiom (to be defined in the sequel or see [Sco80a]) and induces the same λ-theory as that induced by the Longo tree preorder.

- The lazy PSE ordering has a *sound* interpretation in a class of *lazy* λ-models satisfying certain *monotonicity* conditions. Two such lazy λ-models that we will study closely are the free lazy PSE-model and (in Chapter 4) the initial solution of the domain equation: $D \cong [D \to D]_\perp$ in the category of Scott domains where $[- \to -]$ is the function space construction in the category of cpos and $(-)_\perp$ the usual lifting operation.
- We establish a syntactic characterization of the local structure of the class of free *lazy* PSE-models $D_A^\top = \langle D_A, \mathsf{Fun}, \mathsf{Gr}_\top \rangle$ with respect to the interpretation of pure untyped λ-terms. We show that for any λ-terms M, N,

  $$M \lesssim N \iff D_A^\top \models M \subseteq N.$$

  This result confirms and strengthens a conjecture of Longo's (see Remark 3.9 in *op. cit.*), and complements local structure results already known about well-known *sensible* λ-models: for example $\wp\omega$ and D_∞ in [Hyl76], T^ω in [BL80].

3.4.1 Lazy PSE-ordering and Soundness Theorem

The ordering is presented via a family of inductively defined operational preorders indexed over the natural numbers. The preorders are defined recursively over the syntactic shapes or substructures of λ-terms. This way of presenting operational preorders is obviously reminiscent of the approach adopted by Robin Milner and David Park to define observational equivalences in concurrency, see *e.g.* [HM85], as well as Martin Hyland's work in *op. cit.* Alternatively, such preorders may be presented in terms of labelled trees like the classic Böhm trees and Longo trees (see Chapter 2). This is the approach adopted by Henk Barendregt in his treatise on λ-calculus [Bar84]. The former approach seems to lend itself to neater local characterization proofs whereas the latter offers a more perspicuous view of how terms differ in their respective operational behaviours.

DEFINITION **3.4.1.1 (Lazy PSE-ordering)**

- $\forall M, N. M \lesssim^0 N$.
- $M \lesssim^{k+1} N \stackrel{\text{def}}{=}$

  [1] $M \in \mathbf{PO}_0$ or

[2] $N \in \mathbf{PO}_\infty$ or

[3] a. $M \in \mathbf{PO}_n, n \geqslant 1 \Rightarrow N \in \mathcal{F}_m, m \geqslant n$, &

b. $M \twoheadrightarrow_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m \Rightarrow$

$N \twoheadrightarrow_\beta \lambda x_1 \cdots x_{n+l}.yN_1 \cdots N_{m+l}$ &

$\forall 1 \leqslant i \leqslant m.M_i \lesssim^k N_i$ &

$\forall 1 \leqslant j \leqslant l.x_{n+j} \lesssim^k N_{m+j}$ for $m, n, l \geqslant 0$.

- $M \lesssim N \stackrel{\text{def}}{=} \forall k \in \omega.M \lesssim^k N$.

Recall that $M \in \mathcal{F}_m \stackrel{\text{def}}{=} \exists N \in \Lambda.M =_\beta \lambda x_1 \cdots x_m.N$.

REMARK **3.4.1.2** (i) The least and greatest elements w.r.t. $\lesssim$ are the $\mathbf{PO}_0$- and $\mathbf{PO}_\infty$-elements respectively. In fact,

$$\bot \lesssim \lambda x.\bot \lesssim \lambda x_1 x_2.\bot \lesssim \cdots \lesssim \lambda x_1 \cdots x_n.\bot \lesssim \cdots \lesssim \top;$$

and $\lambda x_1 \cdots x_n.\bot \lesssim \lambda x_1 \cdots x_n.M$ where $\bot$ and $\top$ represent any $\mathbf{PO}_0$- and $\mathbf{PO}_\infty$-terms respectively.

That $\lesssim$ has top elements may be given an operational justification as follows. In the lazy regime, convergence to an abstraction is the only kind of "information quantum" observable. It follows that the more deeply-nested an abstraction a term is convertible to, the more information it contains. To extrapolate this argument further, terms which have the greatest amount of information are precisely those which are convertible to unboundedly deeply nested abstractions — unsolvables of infinite order.

(ii) It is not difficult to see that

$$\forall M, N \in \Lambda.M \precsim_L N \Rightarrow M \lesssim N;$$

recall that $\precsim_L$ is the preorder induced by Longo trees. For the benefit of those familiar with the technical jargons of the classical λ-calculus, $\lesssim$ is, roughly speaking, Longo tree preorder with *one-sided η-expansion* (see [Bar84, page 232]) and the additional constraint that the singleton tree with label $\top$ is the top element (not merely maximal).

(iii) It is easy to see that the equivalence relation on λ-terms induced by the Longo tree preorder is the same as that induced by the lazy PSE-preorder.

The first main result we will establish in this subsection is the following Soundness Theorem.

THEOREM **3.4.1.3** (**Soundness**) *Let $M, N \in \Lambda$ and $\mathcal{M} = \langle D, \mathsf{Fun}, \mathsf{Gr} \rangle$, a lazy λ-model that has approximable application and satisfies the monotonicity conditions as listed in the Crucial Proposition (later in the section). Then,*

$$M \lesssim N \Rightarrow \mathcal{M} \models M \sqsubseteq N.$$

□

The rest of this subsection will be devoted to the proof of this Theorem. We will also prove an *Approximation Theorem* á la Wadsworth. Firstly, we define the set of *lazy approximants* of a λ-term M.

DEFINITION **3.4.1.4**

(i) Let $M, N \in \Lambda^\omega\bot$, $M \lesssim N \stackrel{\text{def}}{=} |M|[\bot := \Omega] \lesssim |N|[\bot := \Omega]$ where $\Omega \stackrel{\text{def}}{=} (\lambda x.xx)(\lambda x.xx)$. Let $M \in \Lambda\bot$. Then, $\mathcal{A}(M)$, the set of *lazy approximants* of M, is defined as:

$$\mathcal{A}(M) \stackrel{\text{def}}{=} \{N \in \Lambda\bot : N \lesssim M \,\&\, N \text{ is a } \beta\bot\text{-nf}\}.$$

Recall that $(\bot)$ is the following reduction rule:

$$(\bot) \qquad \bot\vec{M} \to \bot.$$

(ii) Define $\mathcal{A}$, the set of *weak normal forms* (wnf) as the least set satisfying:

$\lambda\vec{x}.\bot, \mathsf{Var} \subseteq \mathcal{A}$;

$\vec{A} \subseteq \mathcal{A} \Rightarrow \lambda\vec{x}.x\vec{A} \in \mathcal{A}$.

$\mathcal{A}$ consists precisely of all $\beta\bot$-nfs. Clearly, for any M, $\mathcal{A}(M) \subseteq \mathcal{A}$.

LEMMA **3.4.1.5** *Each completely-labelled $\lambda\bot$-term has a normal form.*

PROOF Easy. Or by the same reasoning in [Bar84, §7.23]. □

The interpretation of labelled $\lambda\bot$-terms in $\mathcal{M}$ is as defined in Definition 3.2.3.4.

LEMMA **3.4.1.6** *Let $M, N \in \Lambda^\omega\bot$. If* $\mathbf{llab} : M \to N$, *then*

(i) $\mathcal{M} \vDash M \sqsubseteq N$,

(ii) $N \lesssim M$.

PROOF (i) Immediate from the assumption that $\mathcal{M}$ is a lazy λ-model which has lazy approximable application. (ii) It suffices to show that $N \trianglelefteq_L M$ by virtue of Remark 3.4.1.2. Let $M, N, P \in \Lambda^\omega\bot$. If N has no zero labels then the result is trivially true. The only non-trivial approximation occurs when $M \equiv C[P] \stackrel{\Delta}{\to} C[\bot] \equiv N$ where $\Delta : P \to \bot$ is either (β_0) or $(\bot)$. $C[\bot] \trianglelefteq_L C[P]$ is valid by Lemma 2.3.1.17. □

PROPOSITION **3.4.1.7 (Crucial)** *Let* $\mathcal{M} = \langle D, \mathsf{Fun}, \mathsf{Gr}\rangle$ *be a lazy* λ*-model that has lazy approximable application. We assume* D *is a poset with a top element and that* $\mathcal{M}$ *satisfies the following* monotonicity conditions*:*

(1) λ-monotonicity *i.e.* Gr *is monotonic.*

(2) $\mathcal{M}$ *is a model of the* fully lazy λ-theory *i.e.* $\forall m, n \in \omega + 1$

$$\forall M \in \mathbf{PO}_{\mathrm{m}}.\forall N \in \mathbf{PO}_{\mathrm{n}}.\mathcal{M} \models M = N \iff m = n.$$

(3) $\forall M \in \mathbf{PO}_{\infty}.\mathcal{M} \models M = \top$.

(4) $\mathcal{M} \models (\eta^-) \stackrel{\mathrm{def}}{=} \forall d \in D.\mathcal{M} \models \underline{d} \sqsubseteq \lambda x.\underline{d}x$.

Then, for $M, N \in \Lambda^{\perp}$, $N \in \mathcal{A}(M) \Rightarrow \mathcal{M} \models N \sqsubseteq M$.

PROOF Firstly, we establish the following result:
CLAIM 1: $\forall n \geqslant 1.\forall d \in D.\mathcal{M} \models \underline{d} \sqsubseteq \lambda x_1 \cdots x_n.\underline{d}x_1 \cdots x_n$.

We prove by induction on n. The base case of $n = 1$ is just (η^-) — condition (4). Suppose the desired result is valid for $n - 1$. Then

$$\mathcal{M} \models \underline{d}x_1 \sqsubseteq \lambda x_2 \cdots x_n.(\underline{d}x_1)x_2 \cdots x_n.$$

By monotonicity of Gr — condition (1), we have

$$\mathcal{M} \models \lambda x_1.\underline{d}x_1 \sqsubseteq \lambda x_1 \cdots x_n.\underline{d}x_1 \cdots x_n.$$

Then, applying (η^-) again, we establish the inductive case.
NOTATION: For simplicity, we sometimes abbreviate $\mathcal{M} \models N \sqsubseteq M$ as $N \sqsubseteq M$.

We prove the Proposition by structural induction on N (recall that N is a wnf) as follows:
I. Base cases:

1. $N = \bot$. This is clear, since $[\![\bot]\!]_\rho = \bot$, the least element in the domain D.
2. $N = \lambda x_1 \cdots x_n.\bot, n \geqslant 1$. By supposition $N \lesssim M$, we have

   $$M \in \mathcal{F}_{\mathrm{m}}, m \geqslant n \text{ i.e. } M =_\beta \lambda x_1 \cdots x_m.M' \text{ for some } M' \in \Lambda.$$

   By λ-monotonicity, $\mathcal{M} \models N \sqsubseteq M$.
3. $N = x$. By supposition,

   either M belongs to $\mathbf{PO}_\infty$, in which case, since $\mathbf{PO}_\infty$-terms are interpreted as the top element $\top$ in the lattice D — condition (3), trivially $\mathcal{M} \models N \sqsubseteq M$;

   or $M =_\beta \lambda x_1 \cdots x_l.xM_1 \cdots M_l$ for some suitable M_is.

Let I be an arbitrary complete labelling of x.

CLAIM 2: $\forall n \in \omega. \mathcal{M} \models (x, I) \equiv x^n \sqsubseteq M$; from which we deduce that $x \sqsubseteq M$ (by axiom (1) of lazy approximable application). We prove by induction on $I(x)$.

- $I(x) = 0$. Then $[\![(x, I)]\!]_\rho = ([\![x]\!]_\rho)_0 = \bot \sqsubseteq M$ trivially.
- $I(x) = n + 1$. Let the induction hypothesis be:

$$(*) \qquad \forall M \in \Lambda. x \lesssim M \Rightarrow \forall m \leqslant n. \mathcal{M} \models x^m \sqsubseteq M.$$

We have, by Claim 1:

$$\mathcal{M} \models x^{n+1} \sqsubseteq \lambda x_1 \cdots x_l. x^{(n+1)} x_1 \cdots x_l.$$

By axiom (4) of lazy approximable application: $d_{n+1}e \sqsubseteq (de_n)_n$, we deduce that

$$\mathcal{M} \models x^{(n+1)} x_1 \cdots x_l \sqsubseteq x x_1^n x_2^{(n-1)} \cdots x_l^{(n-l+1)}.$$

Hence, by λ-monotonicity,

$$\mathcal{M} \models x^{(n+1)} \sqsubseteq \lambda x_1 \cdots x_l. x x_1^n x_2^{(n-1)} \cdots x_l^{(n-l+1)}.$$

Now, since $N \equiv x \lesssim M$, so, by definition of $\lesssim$, for $1 \leqslant i \leqslant l, x_i \lesssim M_i$. By hypothesis $(*)$, $\mathcal{M} \models x_i^{(n-i+1)} \sqsubseteq M_i$. Therefore,

$$\mathcal{M} \models \lambda x_1 \cdots x_l. x x_1^n x_2^{(n-1)} \cdots x_l^{(n-l-1)} \sqsubseteq \lambda x_1 \cdots x_l. x M_1 \cdots M_l.$$

Hence, $\mathcal{M} \models x^{(n+1)} \sqsubseteq M$.

II. Inductive case: Let $m \neq 0$ and $n \neq 0$ and

$$N = \lambda x_1 \cdots x_n. y N_1 \cdots N_m, \quad M = \lambda x_1 \cdots x_{n+l}. y M_1 \cdots M_{m+l}$$

where $\forall 1 \leqslant i \leqslant m. N_i \lesssim M_i$ and N_i is a $\beta\bot$-nf; and $\forall 1 \leqslant j \leqslant l. x_{n+j} \lesssim M_{m+j}$. Applying the induction hypothesis repeatedly, then by monotonicity of "." and Claim 1:

$$\begin{aligned} y N_1 \cdots N_m &\sqsubseteq \lambda x_{n+1} \cdots x_{n+l}. y M_1 \cdots M_m x_{n+1} \cdots x_{n+l} \\ &\sqsubseteq \lambda x_{n+1} \cdots x_{n+l}. y M_1 \cdots M_{m+l}. \end{aligned}$$

Finally, applying the monotonicity of Gr repeatedly, we get,

$$\mathcal{M} \models \lambda x_1 \cdots x_n. y N_1 \cdots N_m \sqsubseteq \lambda x_1 \cdots x_{n+l}. y M_1 \cdots M_{m+l}.$$

□

THEOREM **3.4.1.8** (**Approximation**) *Let $\mathcal{M}$ satisfy the premises of the Crucial Proposition. Then,*

$$\mathcal{M} \models M = \bigsqcup\{ N : N \in \mathcal{A}(M) \}.$$

PROOF

$$\begin{array}{lll} \mathcal{M} \models M & = & \bigsqcup\{ (M,I) : I \in \mathcal{CL}(M) \} \\ & \sqsubseteq & \bigsqcup\{ P : (M,I) \twoheadrightarrow P \ \& \ |P| \in \mathcal{A}(M) \} \quad \text{Lem. 3.4.1.5,3.4.1.6(ii)} \\ & \sqsubseteq & \bigsqcup\{ N : N \in \mathcal{A}(M) \} \quad \text{by } \mathcal{M} \models P \sqsubseteq |P| \\ & \sqsubseteq & M. \quad \text{Crucial Proposition} \end{array}$$

□

Given $M \in \Lambda$ and $n \geqslant 1$, recall the *n-diagonal approximant* of M, $M^{\langle n \rangle}$ defined in Definition 2.3.1.11. The intuition behind $M^{\langle n \rangle}$ is that it effects approximation in two "dimensions" (see Chapter 2 for further discussion), namely:

- "Depth-wise", $M^{\langle n \rangle}$ is essentially the Longo tree (see Chapter 2) of M truncated after depth n (where the root node corresponds to depth 0);
- "Breadth-wise", unsolvable terms of order greater than n are not distinguished in $M^{\langle n \rangle}$. This provides a means of approximating the $\mathbf{PO}_\infty$-term.

This definition is needed in the lazy regime because as a consequence of *not evaluating under the abstraction* the following approximations become non-trivial:

$$\Omega \sqsubseteq \lambda x.\Omega \sqsubseteq \cdots \sqsubseteq \lambda x_1 \cdots x_n.\Omega \sqsubseteq \cdots \sqsubseteq (\mathbf{YK}).$$

LEMMA **3.4.1.9** (i) $\forall M \in \Lambda . \forall n \in \omega . M^{\langle n \rangle} \in \mathcal{A}$.

(ii) $\forall n \in \omega . M^{\langle n \rangle} \lesssim M$.

(iii) $M \lesssim N \Rightarrow \forall n \in \omega . M^{\langle n \rangle} \lesssim N$.

PROOF (i), (ii) by induction on n. (iii) follows from (ii). □

LEMMA **3.4.1.10** *Let $\mathcal{M}$ satisfy the premises of the Crucial Proposition. Then,*

$$\mathcal{M} \models M = \bigsqcup M^{\langle n \rangle}.$$

PROOF Let $N \in \mathcal{A}(M)$. Let k be greater than the sum of

$$\max\{ n : \exists \alpha \in \mathrm{Seq}. LT(N)\alpha = \lambda x_1 \cdots x_n.\bot \}$$

and the depth of $\mathsf{LT}(N)$, the Longo tree of N which is finite. Then $N \lesssim M^{\langle n \rangle}$. Since $N \in \mathcal{A}(M^{\langle n \rangle})$, by the Crucial Proposition $\mathcal{M} \models N \sqsubseteq M^{\langle n \rangle}$ and the result follows from the Approximation Theorem and that $\mathcal{M} \models M^{\langle n \rangle} \sqsubseteq M$. □

We can now prove the Soundness Theorem.
PROOF

$$\begin{aligned} M \lesssim N &\Rightarrow \forall n \in \omega. M^{\langle n \rangle} \lesssim N \\ &\Rightarrow \forall n \in \omega. \mathcal{M} \models M^{\langle n \rangle} \sqsubseteq N \quad \text{Crucial Proposition} \\ &\Rightarrow \mathcal{M} \models M = \bigsqcup M^{\langle n \rangle} \sqsubseteq N. \quad \text{Lemma 3.4.1.10} \end{aligned}$$

□

Since the free lazy PSE-model and D, the initial solution of $D \cong [D \to D]_\perp$, satisfy the premises of the Crucial Proposition, we remark that the the lazy PSE-ordering is sound with respect to them.

3.4.2 Completeness Theorem for Free Lazy PSE-models

This subsection will be devoted to the proof of the *completeness* part of the *local structure* of free lazy PSE-models.

THEOREM **3.4.2.1 (Local Structure)** *Let $M, N \in \Lambda$. Then,*

$$M \lesssim N \iff D_A^\top \models M \subseteq N.$$

□

In the following, we formalize the notion of the *occurrence* of a subterm of a λ-term and the number of descendents a subterm of a λ-term has (with reference to the Longo tree of the term).

DEFINITION **3.4.2.2** Let $\alpha, \beta \in \mathrm{Seq}$ and $i \in \omega$. Define M_α inductively as follows:

- $M_{\langle\rangle} \stackrel{\text{def}}{=} M$;
- $M_{\langle i \rangle * \beta} \stackrel{\text{def}}{=} \begin{cases} (M_i)_\beta & \text{if } M \twoheadrightarrow_\beta \lambda x_0 \cdots x_{n-1}.yM_0 \cdots M_{m-1}, 0 \leqslant i \leqslant m-1, \\ \text{undefined} & \text{else.} \end{cases}$

If M_α is defined and $M_\alpha \twoheadrightarrow_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m, m \geqslant 0$, we say the *branching factor of* M_α is m. This corresponds precisely to the number of descendents at node α of the tree $\mathsf{LT}(M)$, the Longo tree of M. If M_α is undefined, then we say the branching factor of M at α is 0.

The following formulation follows similar notions which Barendregt defined to prove the local structure theorem of the λ-model $\wp\omega$ in [Bar84].

DEFINITION **3.4.2.3** Let $M, N \in \Lambda, \alpha \in \mathrm{Seq}$. Let p range over ω.

(i) p is *large enough for* α-M-N if p is larger than the branching factor of M_β and N_β for any $\beta \leqslant \alpha$.

(ii) An α-M-N *substitutor* is a map $+_m : \Lambda \to \Lambda(\underline{D_A})$ such that for some variables $y_0, \ldots, y_n$ and some m large enough for α-M-N,

$$P^{+_m} \stackrel{\text{def}}{=} P[y_0 := \mathbf{P}_m y_0] \cdots [y_n := \mathbf{P}_m y_n]$$

for $P \in \Lambda$ where $\mathbf{P}_m$ is the Böhm permutator of degree m introduced earlier. For simplicity, we usually write $+$ for $+_m$.

LEMMA **3.4.2.4** *Let* $M = \lambda x_1 \cdots x_m . y N_1 \cdots N_l$. *Then,*

$$\forall r > l . D_A^\top \models A_{(q)} \subseteq M^{+r} \iff q \leqslant m.$$

PROOF Wlog, and by Lemma 3.3.2.2, say,

$$[\![M^+]\!]_\rho^\top = [\![\lambda x_1 \cdots x_m . \mathbf{P}_r y N_1^+ \cdots N_l^+]\!]_\rho^\perp \cup A_{(m)}.$$

"$\Leftarrow$" Follows immediately from: $A_{(q)} \subseteq A_{(m)}$ iff $q \leqslant m$. "$\Rightarrow$" We prove the contraposition. For $q > m, \beta_i \subseteq_f B$, we have

$$b \equiv (\beta_m; (\beta_{m-1}; \cdots (\beta_1; a_0) \cdots) \in A_{(q)}.$$

Now $b \notin A_{(m)}$, by Theorem 3.3.2.7. By Lemma 3.3.2.14, $[\![\mathbf{P}_r y N_1^+ \cdots N_l^+]\!]_\rho^\perp$ does not contain a_0, hence $b \notin [\![\lambda x_1 \cdots x_m . \mathbf{P}_r y N_1^+ \cdots N_l^+]\!]_\rho^\perp$. We therefore conclude that $A_{(q)} \not\subseteq [\![M^+]\!]_\rho^\top$. □

NOTATION **3.4.2.5** Let $k \in \omega$. We denote $x^{\sim k} \stackrel{\text{def}}{=} \underbrace{x \cdots x}_{k}$.

LEMMA **3.4.2.6** *Let* $M =_\beta y M_1 \cdots M_m, N =_\beta z N_1 \cdots N_n$. *Then,*

$$y \neq z \text{ or } m \neq n \Rightarrow D_A^\top \not\models M \subseteq N.$$

PROOF If $y \neq z$, then take ρ such that $\rho(y) = B$ and $\rho(z) = \emptyset$. If $y = z$ and $n \neq m$, wlog say, $n = m + k$. Then, suppose for a contradiction, $M \subseteq N$. Substituting $\mathbf{P}_{n+1} y$ for y, *i.e.* $+ = [y := \mathbf{P}_{n+1} y]$ and applying both sides to $x^{\sim k} w$, for some variable $w \neq x$,

$$\mathbf{P}_{n+1} y M_1^+ \cdots M_m^+ x^{\sim k} w \subseteq \mathbf{P}_{n+1} y N_1^+ \cdots N_n^+ x^{\sim k} w.$$

We have, $w y M_1^+ \cdots M_m^+ x^{\sim k} \subseteq x y N_1^+ \cdots N_n^+ x^{\sim (k-1)} w$ which contradicts the first part of the proof. □

PROPOSITION **3.4.2.7** $M \not=^1 N \Rightarrow \forall + \in \langle\rangle\text{-}M\text{-}N.D_A^\perp \not\models M^+ \subseteq N^+$.

PROOF I. Suppose $M \in \mathbf{PO}_n, n \geqslant 1$ and we consider the various cases in which $\neg(N \in \mathcal{F}_m, m \geqslant n)$.

a. $N \in \mathbf{PO}_0$. By Theorem 3.3.2.7(i).

b. $N =_\beta yN_1 \cdots N_l$. Suppose for a contradiction, $\exists + \in \langle\rangle\text{-}M\text{-}N$

$$(*) \qquad M^+ \subseteq N^+.$$

(Throughout this proof and the next, we will employ such "proof by contradiction" technique.) Then, wlog (for if $+$ does not involve substitution for y, we will consider $+' = + \cup [y := \mathbf{P}_p y]$. Note that $(*)$ implies $M^{+'} \subseteq N^{+'}$)

$$N^+ = \mathbf{P}_p y N_1^+ \cdots N_l^+$$

which is not saturated by Lemma 3.3.2.14(ii), since $p > l$. Now, for $n > 1, [\![M^+]\!] = A_{(n)}$ is saturated, contradicting $(*)$. For $M \in \mathbf{PO}_1$,

$$[\![M^+]\!] = A_{(1)} = A \ni a_0 \notin [\![N^+]\!]_\rho$$

by Lemma 3.4.2.4.

c. $N \in \mathbf{O}_m$; by supposition, $m < n$. If $N \in \mathbf{PO}_m$, then apply Theorem 3.3.2.7(i) and (ii).
Suppose $N = \lambda x_1 \cdots x_m.yN_1 \cdots N_l$, then apply Lemma 3.4.2.4.

II. Suppose $M \in \mathbf{O}_n \backslash \mathbf{PO}_n$ and $N \notin \mathbf{PO}_\infty$. Let $M =_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m$.

a. $N \in \mathbf{PO}_l$, for $l \in \omega$.
Again, suppose for a contradiction, $\exists + \in \langle\rangle\text{-}M\text{-}N$ such that $M^+ \subseteq N^+$. As before, we may assume $+$ involves y. Applying both sides to $x^{\sim(n+1)}$, we have

$$\mathbf{P}_p y M_1^+ \cdots M_m^+ x \subseteq N^+ x^{\sim(n+1)} = A_{(l')}, \text{ some } l' \in \omega.$$

lhs is $xyM_1^+ \cdots M_m^+$. Choose ρ such that $\rho(x) = B$ to get the contradiction.

b. Suppose $M =_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m, N =_\beta \lambda x_1 \cdots x_{n'}.zN_1 \cdots N_{m'}$.

- Suppose $n > n'$. Apply Lemma 3.4.2.4.
- Suppose $n \leqslant n'$ and $y \neq z$ and suppose $\exists + \in \langle\rangle\text{-}M\text{-}N$

$$M^+ \subseteq N^+.$$

As before, assume the same stipulations about $+$ involving y and z, and applying both sides to $x^{\sim k}$, for sufficiently large k. We get

$$\mathbf{P}_p y M_1^+ \cdots M_m^+ x^{\sim k} \subseteq \mathbf{P}_p z N_1^+ \cdots N_{m'}^+ x^{\sim k'};$$

$$xy M_1^+ \cdots M_m^+ x^{\sim(k-1)} \subseteq xz N_1^+ \cdots N_{m'}^+ x^{\sim(k'-1)}.$$

Interpreting x as I, we have,

$$y M_1^+ \cdots M_m^+ I^{\sim(k-1)} \subseteq z N_1^+ \cdots N_{m'}^+ I^{\sim(k'-1)},$$

contradicting Lemma 3.4.2.6.
Now suppose $n \leqslant n'$ and $y = z$. Say

$$M =_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m; N =_\beta \lambda x_1 \cdots x_{n'}.yN_1 \cdots N_{m'}.$$

Suppose $m' - n' \neq m - n$ and that $\exists + \in \langle\rangle\text{-}M\text{-}N.M^+ \subseteq N^+$. Applying both sides to $x^{\sim k}$, for large k,

$$\mathbf{P}_p y M_1^+ \cdots M_m^+ x^{\sim(k-n)} \subseteq \mathbf{P}_p y N_1^+ \cdots N_{m'}^+ x^{\sim(k-n')}.$$

We get, $xyM_1^+ \cdots M_m^+ x^{\sim(k-n-1)} \subseteq xyN_1 \cdots N_{m'}^+ x^{\sim(k-n'-1)}$. Since $m+k-n \neq m'+k-n'$, we have a contradiction with Lemma 3.4.2.6.

□

REMARK **3.4.2.8** Let $k \geqslant 1$. Observe that if $M \precsim^1 N$ according to clauses [1],[2] or [3]a in Definition 3.4.1.1, then $\forall k \geqslant 1$, $M \precsim^k N$, *i.e.* $M \precsim N$. Hence, if $M \precsim^k N$ and $M \not\precsim^{k+1} N$ for some $k \in \omega$, then we can deduce from the definition of $\precsim^k$ that

(1) $M \twoheadrightarrow_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m$,
$N \twoheadrightarrow_\beta \lambda x_1 \cdots x_{n+l}.yN_1 \cdots N_{m+l}$, for some $n, m, l \geqslant 0$; and

(2) $[\exists 1 \leqslant i \leqslant m.M_i \not\precsim^k N_i]_1$ or $[\exists 1 \leqslant j \leqslant l.x_{n+j} \not\precsim^k N_{m+j}]_2$.

Suppose $[\cdots]_1$ above is true, say, $M_i \not\precsim^{k+1} N_i$, then notice that $M_i \precsim^{k-1} N_i$. One can continue in this fashion, and arrive at $P \not\precsim^1 Q$, where P, Q are respective *homologous* subterms of M and N, *i.e.* P and Q have the same *occurrences* in M and N respectively. Roughly speaking, P and Q are the "largest" homologous pair (of respective subterms of M and N) which gives rise to $M \not\precsim N$.

We formalize the notion as follows:

DEFINITION **3.4.2.9** Let $\alpha \in \mathrm{Seq}$. Inductively, for $M \not\precsim N$ we define $M \not\precsim N \mathbin{@} \alpha$ as follows:

- If $M \not\precsim^1 N$, then $M \not\precsim N \mathbin{@} \langle\rangle$.
- Suppose $M \precsim^k N$ and $M \not\precsim^{k+1} N$ for $k \geqslant 1$. Then, with reference to the notations in the preceding Remark, $M \not\precsim N \mathbin{@} \langle i \rangle * \beta$ or $\langle m+j \rangle * \beta$ according as $M_{\langle i \rangle} \not\precsim N_{\langle i \rangle} \mathbin{@} \beta$ or $x_{n+j} \not\precsim N_{\langle m+j \rangle} \mathbin{@} \beta$ respectively.

Note that $M \lesssim^k N$ and $M \not\lesssim^{k+1} N$ iff $M \not\lesssim N \ @ \ \alpha$ where $\mathsf{lh}(\alpha) = k$.

Intuitively, $M \not\lesssim N \ @ \ \alpha$ means that α is the least occurrence such that $M_\alpha \not\lesssim^1 N_\alpha$.

THEOREM **3.4.2.10** *Suppose $M \not\lesssim N$ such that $M \not\lesssim N \ @ \ \alpha$. Then,*

$$\forall + \in \alpha\text{-}M\text{-}N.D_A^\top \not\models M^+ \subseteq N^+.$$

PROOF By induction on the length of α.
- Base case: $k = 0$, or equivalently, $M \not\lesssim^1 N$. This is precisely Proposition 3.4.2.7.
- Suppose $M \lesssim^k N$ and $M \not\lesssim^{k+1} N$ for $k \geqslant 1$. Then, by definition of $\lesssim$,

$$M \twoheadrightarrow_\beta \lambda x_1 \cdots x_n.yM_1 \cdots M_m,$$

$$N \twoheadrightarrow_\beta \lambda x_1 \cdots x_{n+l}.yN_1 \cdots N_{m+l}, l \geqslant 0$$

and that $M \not\lesssim N \ @ \ \alpha = \langle i \rangle * \beta$ or $\{\langle i + m \rangle * \beta\}$ for $1 \leqslant i \leqslant m$ or $\{1 \leqslant i \leqslant l\}$ respectively. Then, $M_i \not\lesssim N_i \quad \{x_{n+i} \not\lesssim N_{m+i}\} \ @ \ \beta$. The induction hypothesis implies that,

$$(\dagger) \qquad \forall + \in \beta\text{-}M_i\text{-}N_i.D_A^\top \not\models M_i^+ \subseteq N_i^+$$

$$\{\forall + \in \beta\text{-}x_{n+i}\text{-}N_{m+i}.D_A^\top \not\models x_{n+i}^+ \subseteq N_{m+i}^+\}$$

respectively. Suppose for a contradiction, $\exists + \in \alpha\text{-}M\text{-}N$

$$(*) \qquad D_A^\top \models M^+ \subseteq N^+.$$

Wlog, suppose $+$ involves substitution for y. Applying both sides of $(*)$ to $x_1, \cdots, x_{n+l}, z^{\sim k}$, for k sufficiently large, we have,

$$\mathbf{P}_p y M_1^+ \cdots M_m^+ x_{n+1} \cdots x_{n+l} z^{\sim k} \subseteq \mathbf{P}_p y N_1^+ \cdots N_m^+ N_{m+1}^+ \cdots N_{m+l}^+ z^{\sim k};$$

$$zyM_1^+ \cdots M_m^+ x_{n+1} \cdots x_{n+l} z^{\sim(k-1)} \subseteq zyN_1^+ \cdots N_m^+ N_{m+1}^+ \cdots N_{m+l}^+ z^{\sim(k-1)}.$$

Interpreting z as $[\![\mathbf{U}_i^{m+l+k-1}]\!]\ \{[\![\mathbf{U}_{m+i}^{m+l+k-1}]\!]\}$ where $\mathbf{U}_i^r \stackrel{\text{def}}{=} \lambda x_0 \cdots x_r.x_i$. Then, we have $M_i^+ \subseteq N_i^+$ or $\{x_{n+i}^+ \subseteq N_{m+i}^+\}$. Observe that since $+ \in \alpha\text{-}M\text{-}N$, $+ \in \beta\text{-}M_i\text{-}N_i$ or $\{+ \in \beta\text{-}x_{n+i}\text{-}N_{m+i}\}$, contradicting the induction hypothesis $(\dagger)$. □

COROLLARY **3.4.2.11 (Completeness)** *Let $M, N \in \Lambda$. Then,*

$$D_A^\top \models M \subseteq N \Rightarrow M \lesssim N.$$

PROOF Straightforward corollary of Theorem 3.4.2.10. Just consider the trivial $\alpha\text{-}M\text{-}N$ substitution that performs no substitution at all. □

THEOREM **3.4.2.12 (Precongruence of $\precsim$)** *Let $M, N \in \Lambda$. Then,*

$$M \precsim N \Rightarrow \forall C[\,].(C[M] \precsim C[N]).$$

PROOF Recall that "$\cdot$" and "λx" are (continuous) monotonic operations in the free lazy PSE-model. Hence,

$$\forall C[\,] \in \Lambda.D_A^\top \models M \subseteq N \Rightarrow C[M] \subseteq C[N].$$

Result then follows from the Local Structure Theorem of $\precsim$ in $D_A^\top$. □

Chapter 4

Fully Abstract Models of Lambda Transition Systems

Synopsis of the Chapter

The precursor of the work reported in this Chapter is Abramsky's application of the *Stone duality* between domains and their logics of observable properties [Abr87,Abr88] to tackle the problem of full abstraction [Plo77,Mil77] reformulated in the lazy λ-calculus. The study is based on a paradigmatic functional programming language $\lambda\ell$ called the pure lazy language which has closed λ-terms as terms (program phrases). The evaluation mechanism is governed by the reduction relation $\Downarrow$ (we read $M \Downarrow \lambda x.P$ as "M reduces *lazily* to *weak head normal form* $\lambda x.P$"):

$$\frac{}{\lambda x.P \Downarrow \lambda x.P}$$

$$\frac{M \Downarrow \lambda x.P \quad P[x := Q] \Downarrow N}{MQ \Downarrow N}.$$

We say that a term M *converges* if $M \Downarrow \lambda x.P$; otherwise it *diverges*. Under the reduction strategy $\Downarrow$, the possible "results" are of a particularly simple, indeed *atomic* kind. A term M either converges to an abstraction (and according to this strategy, we have no clue as to the structure "under" the abstraction); or it diverges. In contrast to typed λ-calculi with ground constants, say Plotkin's PCF language [Plo77], the computational "observables" in our case is *convergence to abstraction*. As it stands, the relation $\Downarrow$ is too "shallow" to furnish enough information about the behaviour of the system.

Inspired by the work of Milner and Park on concurrency, Abramsky [Abr88] introduced an operational preorder on λ-terms called *applicative bisimulation*

which yields a tool enabling much deeper comparisons between the operational contents of terms to be made using $\Downarrow$ as basic "building blocks". The applicative bisimulation preorder has the following recursive specification:

$$M \precsim^B N \iff M \Downarrow \lambda x.P \Rightarrow N \Downarrow \lambda x.Q \;\&\; [\forall R \in \Lambda^o.P[x := R] \precsim^B Q[x := R]\,].$$

Intuitively, $M \precsim^B N$ holds if "everything that can be observed about M under all tests (=applying M to all possible arguments hereditarily) can also be observed about N". Bisimulation equivalent terms are operationally indistinguishable. We define an (in)equational theory $\lambda\ell \stackrel{\text{def}}{=} \langle \Lambda^o, \sqsubseteq, = \rangle$, which we call the *Abramsky lazy λ-theory* where:

$$\lambda\ell \vdash M \sqsubseteq N \stackrel{\text{def}}{=} M \precsim^B N,$$

$$\lambda\ell \vdash M = N \stackrel{\text{def}}{=} M \sim^B N.$$

(We will refer to the pure lazy language and the λ-theory induced by the associated bisimulation equivalence $\sim^B$ by the same symbol $\lambda\ell$.)

The Abramsky lazy λ-theory is based on an operational model — the transition system $\lambda\ell = \langle \Lambda^o, \Downarrow \rangle$. A general class of structures called *lambda transition systems*, of which the former transition system is an instance, is defined which includes the *language* (whose operational properties are captured in) $\lambda\ell$ and the canonical denotational *model* D — the initial solution of the domain equation $D \cong [D \to D]_\perp$. The full abstraction problem recast in the lazy regime is the following: Is it true that

$$\forall M, N \in \Lambda.M \precsim^B N \iff D \models M \sqsubseteq N?$$

The answer is no and a counter-example is provided. Abramsky [Abr87] achieved full abstraction for D with respect to the language $\lambda\ell_{\mathsf{P}}$ which is $\lambda\ell$ augmented with a parallel convergence construct P. This Chapter continues the study of full abstraction of the canonical model D (and its retracts) with respect to various enriched variants of the language $\lambda\ell$. We present an outline of the Chapter as follows:

- Section 1 begins with an introduction to the *Abramsky lazy λ-theory*, $\lambda\ell$, which is shown to be *Hilbert-Post complete* with respect to the class of *fully lazy λ-theories* (see Chapter 2). The model-theoretic counterpart of $\lambda\ell$ are structures known as *lambda transition systems* (lts) which are essentially lazy λ-models that identify bisimulation equivalent elements. A short historical account on the full abstraction problem is given in Section 2 together with a reformulation of the problem in the untyped lazy regime.

- D, the canonical model for the lazy λ-calculus is the subject matter of Section 3. We study some basic properties of the lts D. Two such properties crucial to the construction of fully abstract retracts of D are:
  - The binary application operator in the lts D *left-preserves arbitrary joins.*
  - The projection functions $\psi_n : D \to D_n$, where D_n is the n-th approximant of D, are $\lambda\mathsf{C}$-*definable*, where C is the *convergence testing* constant.
- We introduce *lazy λ-calculus with convergence testing* in Section 4. The *language* $\lambda\ell_{\mathsf{C}} = \langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}} \rangle$ has expressive power between $\lambda\ell$ and $\lambda\ell_{\mathsf{p}}$. The formal system $\lambda\beta\mathsf{C}$, which is obtained by augmenting $\lambda\beta$ with the convergence testing constant C, is shown to be *Church-Rosser*. A *Standardization Theorem* for the associated reduction is proved. Plotkin's problem of *simulating* call-by-value evaluation in the call-by-name regime is revisited. We show that there is a translation of terms from $\langle \Lambda^o, \Downarrow_{\mathsf{v}} \rangle$ (Plotkin's call-by-value language) to $\lambda\ell_{\mathsf{C}}$ which preserves call-by-value convergence exactly.
- Section 5 presents two *non full abstraction* results. We show that both $\lambda\ell$ and $\lambda\ell_{\mathsf{C}}$ are not fully abstract with respect to D. *Incompleteness* of *fully lazy topological λ-models* (following Honsell and Ronchi della Roccas' terminology) with respect to fully lazy λ-theories is shown as a corollary.
- In Section 6, a general method using *bisimulation logical relations* to construct retracts of D which are fully abstract models for a class of suitably expressive languages (lts's) which includes $\lambda\ell_{\mathsf{C}}$ is presented. We refer the reader to the beginning of Section 6 for an overview of the method used. The problem of constructing a retract of D which is fully abstract with respect to $\lambda\ell$ is reduced by this method to an open question of *conservativity* of $\lambda\ell_\omega$ (a labelled version of $\lambda\ell$) over $\lambda\ell$. This Section ends with an investigation into the relationships between the various lazy operational preorders introduced in this thesis.

4.1 The Lazy Lambda Calculus à la Abramsky

In this section, we introduce Abramsky's *applicative bisimulation ordering* and relate it to *observational* and *contextual preorders*. These lead up to $\lambda\ell$, the *Abramsky lazy λ-theory*, which is shown to be *fully lazy*. In fact, $\lambda\ell$ is *Hilbert-Post complete* (*i.e.* maximal consistent) with respect to the class of fully lazy λ-theories. The natural λ-model theoretic counterpart of the bisimulation ordering and Abramsky lazy λ-theory are structures known as *lambda transition systems* (lts) which are essentially lazy λ-models that do not distinguish between

bisimulation equivalent elements. Perhaps not surprisingly, *adequate* lts's induce fully lazy λ-theories.

4.1.1 Introduction

In his thesis [Abr87], Abramsky elegantly demonstrated a synthesis of a number of hitherto separate developments in Theoretical Computer Science and Logic:

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

The key to the synthesis is the mathematical theory of *Stone duality*, which provides a junction between semantics (topological spaces) and the *logic of observables properties* (locales). As a major case study of the general theory developed in his thesis, Abramsky turned to the foundations of functional programming in Chapter 6 of his PhD thesis [Abr87] entitled *Applications to Functional Programming: The Lazy Lambda Calculus* which constitutes the main precursor of the work in this Chapter. (His other major case study concerns concurrency, documented in Chapter 5 of *op. cit.* entitled *Applications to Concurrency: A Domain Equation for Bisimulation.*)

We begin by formalizing a reduction strategy on closed λ-terms which we will regard as a quintessential functional programming language.

DEFINITION **4.1.1.1** The relation $M \Downarrow N$ ("M converges to *principal weak head normal form* N") is defined inductively over Λ^o as follows:

(abs-$\Downarrow$)
$$\frac{}{\lambda x.M \Downarrow \lambda x.M}$$

(β_N)
$$\frac{M \Downarrow \lambda x.P \quad P[x := Q] \Downarrow N}{MQ \Downarrow N}$$

The transition system $\lambda\ell \stackrel{\text{def}}{=} \langle \Lambda^o, \Downarrow \rangle$ is called the *pure lazy language.*

The reduction strategy captured by the above formulation is precisely the *leftmost* reduction with the proviso that all reductions terminate upon reaching abstractions. The relation $\Downarrow$ is essentially the same as the "eval" relation in Plokin's *call-by-name* programming language defined in [Plo75]; hence the subscript "N" in the label (β_N) of the second inference rule.

NOTATION **4.1.1.2**

$$M{\Downarrow} \stackrel{\text{def}}{=} \exists N.M \Downarrow N \quad (\text{“}M \text{ converges”})$$
$$M{\Uparrow} \stackrel{\text{def}}{=} \neg(M{\Downarrow}) \quad (\text{“}M \text{ diverges”})$$

Plainly, the reduction relation $\Downarrow$ defines a partial function from Λ^o to the class of principal weak head normal forms (the “range” of $\Downarrow$ is the collection of all closed abstractions, to be precise).

4.1.2 Bisimulation Ordering

Under the reduction strategy $\Downarrow$, the possible “results” are of a particularly simple, indeed *atomic* kind. That is to say, a term M either converges to an abstraction (and according to this strategy, we have no clue as to the structure “under” the abstraction); or it diverges. In contrast to typed λ-calculi with ground constants, say Plotkin's PCF language [Plo77], the computational “observables” in our case is “convergence to abstraction” — the satisfaction of a semi-decidable predicate. One may indeed question whether the relation $\Downarrow$ as it stands now is not too “shallow” to furnish any relevant information about the behaviour of the system. To see why this is so, just note that although $\Downarrow$ distinguishes between *functional* terms (*i.e.* those β-convertible to abstractions and hence capable of an *operator* role) and *non-functional* terms (*i.e.* those which are incapable of playing an operator role genuinely, e.g. $(\lambda x.xx)(\lambda x.xx)$); it does not distinguish one abstraction from the other.

Inspired by the work of Robin Milner and David Park on concurrency, Abramsky introduced an operational preorder on λ-terms called *applicative bisimulation*[1] which yields a tool enabling much deeper comparisons between the operational contents of terms to be made. Indeed, there is theoretically pleasing evidence (*i.e.* applicative bisimulation preorder is precisely characterized by observability under all contexts, Proposition 4.1.3.5) corroborating the view that applicative bisimulation is perhaps the *finest* extensional or behavioural preorder one would wish to impose on “lazy programs”.

Applicative bisimulation is consonant with the computational perspective that regards *applicative behaviour* as the touchstone for comparison between programs. Here, applicative behaviour of a program refers to the computational outcome that is *observable* when the program is applied to all possible input “arguments”.

[1] As pointed out to me by an Edinburgh audience, in keeping with the usual distinction between *simulation* and *bisimulation* in concurrency, the operational preorder should be named more accurately as *applicative simulation*. However, to avoid further confusion of terminology and in deference to the author of the preorder, we will adhere to applicative *bisimulation*.

Recall that in λ-calculus, no *a priori* distinction is made between "programs" and "datum". The grammar of the calculus implicitly allows for any λ-term to act as an *operator* as well as an *operand* in a program of which it is a part, according to its (applicative) position in the program.

Divergent terms are the least informative terms with respect to bisimulation ordering. (Closed) convergent terms M and N are compared by performing "tests" on them. The suite of tests consists in applying M and N respectively to each closed program $P \in \Lambda^o$ in turn and noting their respective applicative behaviour under the tests; and then performing the same suite of tests on their respective applicative outcome in a hereditary fashion.

We prescribe a recursive specification of the applicative bisimulation ordering $M \lesssim^B N$ (which we read as "M is bisimulation less then N"), for the moment restricted to closed terms M and N, as follows:

$$M \lesssim^B N \iff M \Downarrow \lambda x.P \Rightarrow \{ N \Downarrow \lambda x.Q \;\&\; \forall R \in \Lambda^o . P[x := R] \lesssim^B Q[x := R] \}.$$

Applicative bisimulation may be defined as the conjunction of a family of inductively defined preorders as follows:

DEFINITION **4.1.2.1 (Abramsky)** We define a sequence of binary relations $\{ \lesssim^B_k : k \in \omega \}$ on Λ^o as follows:

- $\forall M, N . M \lesssim^B_0 N$.
- $$M \lesssim^B_{k+1} N \stackrel{\text{def}}{=} M \Downarrow \lambda x.P \Rightarrow N \Downarrow \lambda x.Q \;\&\; \forall R \in \Lambda^o . P[x := R] \lesssim^B_k Q[x := R].$$
- $M \lesssim^B N \stackrel{\text{def}}{=} \forall k \in \omega . M \lesssim^B_k N$.

The definition is then extended to all λ-terms by considering closures in the usual way, *i.e.* for $M, N \in \Lambda$,

$$M \lesssim^B N \stackrel{\text{def}}{=} \forall \sigma : \mathsf{Var} \to \Lambda^o . M_\sigma \lesssim^B N_\sigma$$

(where e.g. M_σ means the result of simultaneously substituting $\sigma(x)$ for each occurrence of x in M with x ranging over Var). We abbreviate $M \lesssim^B N \& N \lesssim^B M$ as $M \sim^B N$.

It should not be difficult to see that an equivalent definition of applicative bisimulation is the following:

$$M \lesssim^B N \iff \forall \vec{P} \subseteq \Lambda^o . M\vec{P}\Downarrow \Rightarrow N\vec{P}\Downarrow.$$

DEFINITION **4.1.2.2** We define an (in)equational theory $\lambda\ell \stackrel{\text{def}}{=} \langle \Lambda^o, \sqsubseteq, = \rangle$ which we call *Abramsky lazy λ-theory* where:

$$\lambda\ell \vdash M \sqsubseteq N \stackrel{\text{def}}{=} M \lesssim^B N,$$

$$\lambda\ell \vdash M = N \stackrel{\text{def}}{=} M \sim^B N.$$

Of course, we need to verify that $\lambda\ell$ is indeed a λ-theory (in fact, it is a *fully lazy λ-theory*); and this we will do in the following subsection.

Note that we refer to the pure lazy language and the λ-theory induced by the associated bisimulation equivalence $\lesssim^B$ by the same name $\lambda\ell$.

4.1.3 Contextual and Observable Pre-orders

Let Q be a proper subset of Λ^o closed under α- and β-conversion. We call Q the "observables". Following Morris [Mor68], we define a binary relation $\lesssim_Q$ on Λ^o (we read $M \lesssim_Q N$ as "M is Q-observable less than N"):

$$M \lesssim_Q N \stackrel{\text{def}}{=} \forall C[\,] \in \Lambda^o . C[M] \in Q \Rightarrow C[N] \in Q.$$

We abbreviate $M \lesssim_Q N \;\&\; N \lesssim_Q M$ as $M \sim_Q N$, which we call *Q-observable equivalence.* Note that

$$M \sim_Q N \iff \forall C[\,] \in \Lambda^o . C[M] \in Q \Leftrightarrow C[N] \in Q.$$

LEMMA **4.1.3.1** *Let Q be as before. Then, $\lesssim_Q$ is a pre-congruence i.e.*

$$\forall M, N \in \Lambda^o . M \lesssim_Q N \Rightarrow \forall C[\,] \in \Lambda^o . C[M] \lesssim_Q C[N].$$

Consequently, $\sim_Q$ is a congruence relation.

PROOF We prove the Lemma by a sequence of assertions. We claim: For all $M, N, P \in \Lambda$:

(i) $M \lesssim M$.

(ii) $M \lesssim N \;\&\; N \lesssim P \Rightarrow M \lesssim P$.

(iii) $M \lesssim N \Rightarrow M[x := P] \lesssim N[x := P]$.

(iv) $M \lesssim N \Rightarrow P[x := M] \lesssim P[x := N]$.

(v) $\lambda x.M \lesssim \lambda y.M[x := y]$.

(vi) $M \lesssim N \Rightarrow \lambda x.M \lesssim \lambda x.N$.

(vii) $M_i \precsim N_i \quad (i = 1,2) \Rightarrow M_1 M_2 \precsim N_1 N_2$.

(i) and (ii) are trivial. For (iv), assume $M \precsim N$. Suppose for some context $C[\,]$, $C[P[x := M]] \in Q$. Consider the context $D[\,] \equiv C[(\lambda x.P)[\,]]$. Since Q is closed under β-conversion, we deduce that $D[M] \in Q$. By assumption, $D[N] =_\beta C[P[x := N]] \in Q$. Proofs for the rest are similar. For (vii), an appeal to (ii) is needed. □

Recall that a λ-theory $\mathcal{T}$ is a consistent extension of the formal theory $\lambda\beta$. Given Q as above, define

$$\mathcal{T}_Q \stackrel{\text{def}}{=} \{ M = N : M, N \in \Lambda^o . M \sim^Q N \}.$$

PROPOSITION **4.1.3.2** *Let Q be as above. Then $\mathcal{T}_Q$ is a λ-theory.*

PROOF Since Q is closed under β-conversion, $\lambda\beta \subseteq \mathcal{T}_Q$. Since $\mathcal{T}_Q \vdash M = N$ implies $\forall C[\,] \in \Lambda^o . C[M] \sim^Q C[N]$, we conclude that $\mathcal{T}_Q = \mathsf{Th}(\mathcal{T}_Q)$. To show that $\mathcal{T}_Q$ is consistent, choose $M \in Q$ and $N \notin Q$. Then, trivially $\mathcal{T}_Q \nvdash M = N$. □

The converse is also true. We say that a λ-theory $\mathcal{T}$ is *contextual* if $\exists Q \subseteq \Lambda^o . \mathcal{T} = \mathcal{T}_Q$ with Q a proper subset of Λ^o closed under α- and β-conversions. The following result is attributed to Dana Scott in [HdR88].

PROPOSITION **4.1.3.3 (Scott)** *Every λ-theory is contextual.*

PROOF Given a λ-theory $\mathcal{T}$. Define

$$Q \stackrel{\text{def}}{=} \{ M : \exists U, V . M =_\mathcal{T} [U, V] \,\&\, U =_\mathcal{T} V \}.$$

Recall that $[U, V] \stackrel{\text{def}}{=} \lambda x.xUV$ where $x \notin \mathsf{FV}(U) \cup \mathsf{FV}(V)$. It is straightforward to see that $\mathcal{T} \subseteq \mathcal{T}_Q$. To show $\mathcal{T}_Q \subseteq \mathcal{T}$, suppose $\mathcal{T} \nvdash M = N$ and let $C[\,] \equiv \lambda x.x[\,]N$. Clearly, $C[N] \in Q$. Suppose, for a contradiction, $C[M] \in Q$. Then, for some $U =_\mathcal{T} V$, we have

$$C[M] =_\mathcal{T} \lambda x.xUV.$$

Now, applying the preceding equation to $\mathbf{K}$ and $\mathbf{F} \equiv \lambda xy.y$ respectively, we get:

$$M =_\mathcal{T} U, \qquad N =_\mathcal{T} V.$$

But $U =_\mathcal{T} V$ by supposition, leading to a contradiction. □

Properties of $\precsim^B$ and $\lambda\ell$

We return now to the bisimulation ordering $\precsim^B$ and ask if it can be characterized by a Morris-style Q-observable precongruence. The answer, due to Samson Abramsky, is yes. First, a definition.

DEFINITION **4.1.3.4** *The binary relation $\precsim^C$ on Λ^o is defined as*

$$M \precsim^C N \stackrel{\text{def}}{=} \forall C[\,] \in \Lambda^o.C[M]\Downarrow \Rightarrow C[N]\Downarrow.$$

PROPOSITION **4.1.3.5 (Abramsky)** $\precsim^B = \precsim^C$.

PROOF We refer the reader to Abramsky's proof *op. cit.* which makes essential use of domain logic, although the statement of the Proposition does not mention domains at all. I know of no other proof. □

The importance of the above Proposition is that it follows that the Abramsky bisimulation preorder is a *logical relation* [Plo73] (see definition later in the Chapter) — a highly non-trivial property given the way $\precsim^B$ is defined. Another consequence of this Proposition is that application is monotonic w.r.t $\precsim^B$.

In view of the results we have hitherto obtained for Q-observable precongruence and Proposition 4.1.3.5, it should now be clear that

- $\lambda\ell$ is a λ-theory and that
- the "observables" are $\mathcal{F}$, the set of *functional terms*, *i.e.* λ-terms convertible to abstractions. We have

$$\lambda\ell \text{ (equational)} = \mathcal{T}_{\mathcal{F}}.$$

We can say rather more about the λ-theory $\lambda\ell$. In fact, $\lambda\ell$ is *Hilbert-Post complete* w.r.t. fully lazy λ-theories, *i.e.* it is the *maximal consistent* fully lazy λ-theory. Before we prove the assertion, it is helpful to recall:

FACT **4.1.3.6** *Let $L \in \mathbf{PO}_0, M \in \mathbf{PO}_{n+1}$ for $n \in \omega$ and $N \in \mathbf{PO}_\infty$. Then, for any $P \in \Lambda$,*

(1) $LP \in \mathbf{PO}_0$,

(2) $MP \in \mathbf{PO}_n$,

(3) $NP \in \mathbf{PO}_\infty$.

□

The reader should now refresh his memory of such λ-theoretic notions as *pre-lazy* and *fully lazy* λ-theories as introduced in Chapter 2.

LEMMA **4.1.3.7** *$\lambda\ell$ is a* fully lazy *λ-theory, i.e. for $m, n \in \omega + 1$*

$$\forall M \in \mathbf{PO}_m . \forall N \in \mathbf{PO}_n . [\lambda\ell \vdash M = N \iff m = n].$$

PROOF By considering the observable applicative behaviour, it is easy to see that all strongly unsolvable terms are identified in $\lambda\ell$, similarly for $\mathbf{PO}_\infty$-terms. Hence, by Lemma 2.4.2.4, λ is pre-lazy. Since $\mathbf{PO}_1$ and $\mathbf{PO}_2$ are not provably equal in $\lambda\ell$, applying Corollary 2.4.2.6, we conclude that $\lambda\ell$ is fully lazy. □

The following Proposition says that $\lambda\ell$ is a *maximal* fully lazy λ-theory.

PROPOSITION **4.1.3.8** *The Abramsky lazy λ-theory, $\lambda\ell$, is Hilbert-Post complete with respect to the class of fully lazy λ-theories, i.e. for $P, Q \in \Lambda$ such that $\lambda\ell \nvdash P = Q$, either $\lambda\ell + (P = Q)$ is inconsistent or it is not fully lazy.*

PROOF Suppose $\lambda\ell \nvdash M = N$. Then,

$$\exists C[\,] \in \Lambda^o . C[M]\Downarrow \;\&\; C[N]\Uparrow.$$

- Suppose $C[M]$ is solvable. Then, by definition, $(\lambda\vec{x}.C[M])\vec{P} = \mathbf{I}$ for some $\vec{x}, \vec{P}$; but $(\lambda\vec{x}.C[N])\vec{P} \equiv L$ is unsolvable, by Corollary 8.3.4 in [Bar84]. Wlog, we only consider the cases where the order of unsolvability of L is 0 and ∞ respectively. Suppose $L \in \mathbf{PO}_0$, then

$$\begin{aligned} \lambda\ell + (M = N) \vdash \mathbf{I} &= (\lambda\vec{x}.C[M])\vec{P} = (\lambda\vec{x}.C[N])\vec{P} \\ &= \mathbf{\Omega}_3 \stackrel{\text{def}}{=} (\lambda x.xxx)(\lambda x.xxx). \end{aligned}$$

  Now, every λ-theory which equates $\mathbf{I}$ and $\mathbf{\Omega}_3$ is inconsistent, for

$$\mathbf{I} = \mathbf{\Omega}_3 = \mathbf{\Omega}_3(\lambda x.xxx) = \mathbf{I}(\lambda x.xxx) = (\lambda x.xxx);$$

  i.e. two distinct nf's are equated, and so, by Böhm's Theorem, the theory is inconsistent.
  If $L \in \mathbf{PO}_\infty$, then similarly, we have

$$\lambda\ell + (M = N) \vdash \mathbf{I} = \mathbf{YK} = \mathbf{K}(\mathbf{YK}) = \lambda x.\mathbf{I};$$

  and so, by Böhms Theorem again, $\lambda\ell + (M = N)$ is inconsistent.

- Suppose $C[M]$ is unsolvable. Then $C[M] \in \mathbf{PO}_n$ where $0 \neq n \in \omega + 1$. We have

$$\lambda\ell + M = N \vdash \mathbf{PO}_n \ni C[M] = C[N] \in \mathbf{PO}_0;$$

  i.e. $\lambda\ell + M = N$ is not a fully lazy λ-theory.

□

4.1.4 Lambda Transition Systems

At this point, the reader should refresh his memory of such notions as quasi-applicative transition systems, quasi-applicative structures with divergence and lazy λ-models as introduced in Chapter 3. A lambda transition system is essentially a lazy λ-model which satisfies a very strong extensionality principle ($\mathrm{Ext}_{\mathrm{bisim}}$). It was introduced by Abramsky [Abr87, Chap 6] as a *general* class of lazy λ-models in which to interpret the domain logic corresponding to the lazy λ-calculus. The significant point to note is that *both* the *language* $\lambda\ell$ and the canonical *model* D are instances of the same class (in fact, D is *final* in the category **LTS**, see page 196 in *op. cit.*). This enables general *logical* properties to be transferred from one structure to the other.

Recall that an applicative structure with divergence (aswd), $\mathcal{A} = \langle A, \cdot, \Uparrow \rangle$, is a q-aswd which satisfies the following axiom:

(aswd) $$\forall a, b \in A. \forall c \in A. a \sqsubseteq b \Rightarrow c \cdot a \sqsubseteq c \cdot b;$$

where $\sqsubseteq$ is the associated *bisimulation ordering*. We abbreviate $a \sqsubseteq b \;\&\; b \sqsubseteq a$ as $a \sim b$. Note that

$$a \sqsubseteq b \iff \forall \vec{d} \subseteq A. a\vec{d}\Downarrow \Rightarrow b\vec{d}\Downarrow.$$

DEFINITION **4.1.4.1** An aswd $\mathcal{A} = \langle A, \cdot. \Uparrow \rangle$ satisfies the *bisimulation extensionality principle* if the following axiom is satisfied:

($\mathrm{Ext}_{\mathrm{bisim}}$) $$\forall a, b \in A. a \sim b \Rightarrow a = b.$$

DEFINITION **4.1.4.2**
Lambda Transition System is a structure $\mathcal{A} = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]_- \rangle$ such that

(1) $\langle A, \mathsf{Fun} \rangle$ is an applicative transition system (with $\mathrm{dom}(\mathsf{Fun}) = A \backslash \{\bot\}$),

(2) $\mathcal{A}$ is a functional lazy λ-model,

(3) $\mathcal{A} \models (\mathrm{Ext}_{\mathrm{bisim}})$.

A lambda transition system (lts) $\mathcal{A}$ is *adequate*[2] if $\forall M \in \Lambda^o. \mathcal{A} \models M\Downarrow \Rightarrow \lambda\ell \vdash M\Downarrow$.

REMARK **4.1.4.3** (i) Let $\mathcal{A} = \langle A, \mathsf{Fun} \rangle$ be an ats. It is easy to see that

$$\mathcal{A} \models (\mathrm{Ext}_{\mathrm{bisim}}) \Rightarrow \mathcal{A} \models (\mathrm{Ext}_{\mathrm{cond}});$$

[2]Abramsky calls this notion *sensibleness*. We prefer to call it *adequacy* so as to reserve sensibleness for the classical sense of identifying the unsolvables.

the converse is not true in general. Hence, by Lemma 3.2.2.5, if $\mathcal{A}$ is a lts, then condition (3) above implies that

$$\mathcal{A} \models (\eta_{\text{cond}}), (\text{Ext}_{\text{cond}}) \;\&\; \mathsf{Gr} \circ \mathsf{Fun} = \mathsf{id}_{A\setminus\{\bot\}}.$$

(ii) In view of the equivalence between environment and functional lazy λ-models (Proposition 3.2.2.3), lambda transition system may be defined equivalently as the structure $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle$ with

(1) $\langle A, \cdot, \Uparrow \rangle$ is an applicative structure with divergence such that $\Uparrow = \{\bot\}$,

(2) $\mathcal{A}$ is an environment lazy λ-model,

(3) $\mathcal{A} \models (\text{Ext}_{\text{bisim}})$.

(iii) Our formulation of lts is essentially equivalent to that of Abramsky's. The only point of departure is that in our formulation, bisimulation equivalent elements are identified *axiomatically.*

LEMMA **4.1.4.4** *Let $\mathcal{A} = \langle A, \mathsf{Fun}, \mathsf{Gr}, [\![-]\!]_- \rangle$ be a lts. Then, with respect to the bisimulation ordering, A has unique least and greatest elements denoted $\bot$ and $\top$ respectively. The top element, $\top$, is the interpretation of any $\mathbf{PO}_\infty$-element. Also, $\bot$ and $\top$ are the only solutions, divergent and convergent respectively, to the following equation:*

$$\forall a \in A. x \cdot a = x.$$

PROOF It is easy to see that $\bot$, the only divergent element, is the least element w.r.t. $\sqsubseteq$; and satisfies the equation. The top element is that $x \in A$ such that $\forall y \in A. x\vec{y}\Downarrow$. By $(\text{Ext}_{\text{bisim}})$, such x is unique.
CLAIM: If x is a convergent solution to $\forall a \in A. x \cdot a = x$, then x is the greatest element with respect to $\sqsubseteq^B$.

By definition, $M \in \mathbf{PO}_\infty \Rightarrow MN \in \mathbf{PO}_\infty$. Since all $\mathbf{PO}_\infty$-terms have the same denotation in $\mathcal{A}$, we have $\forall x \in A. lx = l$ with l being the denotation of $\mathbf{PO}_\infty$-terms. Since $l\Downarrow$, we have $\forall \vec{y} \subseteq A. l\vec{y}\Downarrow$; and so, $l = \top$. □

COROLLARY **4.1.4.5** *If a lts $\mathcal{A}$ is* adequate, *then $\mathcal{A}$ is a fully lazy λ-model, i.e. for $m, n \in \omega + 1$,*

$$\forall M \in \mathbf{PO}_m, \forall N \in \mathbf{PO}_n. [\mathcal{A} \models M \sqsubseteq N \iff m \leqslant n].$$

PROOF Easy. □

It follows that any adequate lts induces a fully lazy λ-theory. Is every fully lazy λ-theory induced by an adequate lts? In other words, are adequate lts's *complete* with respect to fully lazy λ-theories? We will return to this question later in the Chapter.

EXAMPLE **4.1.4.6** (i) D, the initial solution of the domain equation $D \cong [D \to D]_\perp$ in the category of cpos and continuous functions is an lts. In fact, it is *adequate* and *internally fully abstract i.e.*

$$\forall x, y \in D. x \lesssim^B y \iff x \sqsubseteq y;$$

where $\lesssim^B$ is the associated bisimulation ordering of the q-aswd D.

(ii) $\langle \Lambda^o/\sim^B, \Downarrow \rangle$ is a lts. That the quotient is well-defined — the validity of the axiom (aswd) — follows from Proposition 4.1.3.5.

4.2 The Full Abstraction Problem — A Brief Introduction

This brief section introduces the background to the *full abstraction* problem as studied by Samson Abramsky in Chapter 6 of his PhD thesis. We review some basic notions and notations and state the main problem to which the rest of this Chapter is a partial solution.

4.2.1 A Historical Introduction

The *full abstraction* problem was first studied by Gordon Plotkin in the seminal paper [Plo77], and shortly after by Robin Milner [Mil77]. Informally stated, it is concerned with the problem of finding a denotational semantic definition for a programming language which is not "over-generous" with respect to a notion of operational equivalence defined by observational indistinguishability. Two program fragments are equivalent if their respective observable computational outcome under all program contexts are identical. Consider a programming language with a class of *observable* C consisting typically of a collection of ground constants, and $\mathcal{L}$, the class of well-formed *terms* (which may be typed or untyped) including a subclass of *programs*; equipped with the following:

- A deterministic *evaluation* mechanism eval : $M \rightharpoonup C$ where M ranges over programs. This we regard as specifying the operational behaviour of the language.
- $[\![-]\!] : \mathcal{L} \to D$ is the denotational semantic definition where D is an ordered semantic domain satisfying the following *soundness* or *adequacy* condition: for any *program* M,

$$\mathsf{eval}(M) = c \iff [\![M]\!] = [\![c]\!] \text{ and}$$

$$\mathsf{eval}(M) \text{ undefined} \iff [\![M]\!] = \bot.$$

We shall assume the semantic function is *homomorphic* with respect to program constructors which are denoted by continuous functions in the appropriate domains.

In program analysis, especially in the design of optimizing compilers or transformation tools where the preservation of the meanings of programs is vital, knowledge of whether a program fragment or term may be *safely* replaced by another in all program contexts is indispensable. We formalize this as follows. Let **Cxt** be the class of *program contexts* and for programs M and N, we write $M \leqslant N$ to mean

"if $\mathsf{eval}(M)$ is defined, then so is $\mathsf{eval}(N)$ and that $\mathsf{eval}(M) = \mathsf{eval}(N)$".

Now, define an operational preorder $\sqsubseteq$ on terms as follows (we read $M \sqsubseteq N$ as "M *safely* approximates N in all contexts"):

$$M \sqsubseteq N \stackrel{\text{def}}{=} \forall C[\,] \in \mathbf{Cxt}.C[M] \leqslant C[N].$$

Since reasoning with the relation $\sqsubseteq$ as defined typically involves rather unwieldy syntactic calculations, we would like a denotational semantic definition which corresponds to the operational semantics sufficiently closely so that reasonings such as "substitutivity under all contexts" may be carried out in the *semantic* domain. More precisely, we seek the following *fully abstract* denotational semantics: for all terms M, N,

$$M \sqsubseteq N \iff [\![M]\!] \sqsubseteq [\![N]\!].$$

In [Plo77], Plotkin posed and studied the full abstraction problem for PCF which is a simply-typed λ-calculus augmented with elementary arithmetic operations. The *standard* cpo-domains for PCF consist of two base domains, namely, flat integer domain, and flat truth values domain (both *without* "top" elements); and all continuous function spaces built up successively from them. We call the class of domains built up in the same way from the two base domains with top-elements added *standard* lattice-domains. Plotkin showed that PCF is not fully abstract with respect to the standard cpo-domains — in fact PCF is not fully abstract with respect to any *homomorphic retracts* of the standard cpo-domains; however PCF augmented with a "parallel or" construct is. An important advance in the problem was achieved by Robin Milner who showed in [Mil77] how to construct a fully abstract model from any collection of consistently complete

ω-complete cpo's $\mathcal{D}$ for the class of simply-typed λ-calculi augmented with continuous first-order functions over $\mathcal{D}$. The construction may be seen as mirroring the well-known Scott's construction of a solution to the domain equation $D \cong [D \rightarrow D]$ *syntactically*. Milner also identified the conditions under which there exists, for the class of the extended λ-calculi, a unique fully-abstract model.

Two Approaches to Full Abstraction

Full abstraction is attained if all the compact elements of the (algebraic) semantic domains are *definable* in the language. Plotkin's and Milner's approaches exemplify two natural directions in which to achieve full-abstraction:

- The *expansive* approach consists in enriching the language, enhancing its denotational expressiveness, as in the introduction of parallel or to PCF, thereby enabling all *finite* semantic information to be represented syntactically as program phrases.
- The *restrictive* approach is to "cut down" (as in "quotienting out" by an appropriate operational equivalence relation) the existing "over-generous" semantic domain to that sub-structure that "fits" the prescribed language. Milner's construction exemplifies this approach.

The restrictive approach to the full abstraction problem is notoriously difficult. Although Milner showed that a fully abstract model for PCF exists, there has been little real progress as regards the construction of a satisfactory (mathematical) fully abstract *sub-model* during the intervening years. A number of researchers have nonetheless been motivated by the problem to investigate related areas of interests. Their work, which has enriched the general area of denotational semantics and domain theory include: sequentiality [Ber78], stability [Ber78], concrete data structure [KP78], [BC82] and event structures [Win80], [Win87].

More recently, Ketan Mulmuley [Mul86] constructed a fully-abstract model for PCF which is a *retract* (but not a *sub-model*) of the standard *lattice*-domains by using the technique of *logical relations* (or *inclusive predicates*) whose relevance to the problem of definability was first recognized and studied by Plotkin in a pioneering paper [Plo73].

4.2.2 Full Abstraction Problem in the Lazy Regime

A fully abstract semantics of a programming language equates two program phrases provided their respective observable computational outcome when embedded in all program contexts are identical. The first question that arises when

the problem of full abstraction is recast in the untyped regime possibly *without* any constants (as in the *pure lazy language* $\langle \Lambda^o, \Downarrow \rangle$) is what should the *observables* be. Abramsky's approach is to posit that *convergence to abstraction* according to a certain deterministic reduction strategy constitutes *all* that can be observed in the pure untyped regime.

Consider the full abstraction problem for the pure lazy language $\langle \Lambda^o, \Downarrow \rangle$. The semantic domain is D, the initial solution of the domain equation $D \cong [D \to D]_\perp$ in the category of cpos (see Chapter 6 of Abramsky's thesis or the following section for justification). By an appeal to Proposition 4.1.3.5, the full abstraction problem may be equivalently presented as follows: Is it true that

$$\forall M, N \in \Lambda. M \lesssim^B N \iff D \models M \sqsubseteq N?$$

It turns out that D is *not* fully abstract for the pure lazy language, a proof of which is presented in the sequel. Abramsky shows in his thesis that D is fully abstract for the pure lazy language extended with a *parallel convergence constant* P; but it is not so for a weaker language $\lambda\ell_{\mathsf{C}} = \langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}} \rangle$ which is the pure lazy language extended by a *convergence testing constant.* His proof of the full abstraction result exploits the Stone duality between the domain D and its logic, the details of which are beyond the scope of this thesis.

Abramsky achieved full abstraction with respect to D by augmenting the pure lazy language with a sufficiently expressive constant so that all compact elements of D are representable — an *expansive* approach. In the Chapter, we consider the *restrictive* approach *i.e.* the question of whether there are *reasonable substructures* of D which are fully abstract with respect to various enriched variants of $\lambda\ell$.

4.3 Basic Properties of D — Canonical Model for Pure Lazy Language

This section studies the basic properties of D which we regard as the canonical model for the pure lazy language. D is an *internally fully abstract, adequate* lts whose application left-preserves *arbitrary joins.* Some index calculation results are presented. The projection maps $\psi_n : D \to D_n$, where D_n is the n-th approximant of D, are shown to be λC-*definable*, where C is the convergence testing constant to be introduced in the sequel.

4.3.1 Introduction

In the *standard* theory, λ-calculus may be regarded as being characterized by the type equation

$$D = [D \to D].$$

Regarded as a domain equation, it has non-trivial solutions in many categories including those of cpo's, Scott domains, and complete lattices (all these with continuous functions as morphisms). Note however, the initial solution is trivial — the one-point domain.

In the *lazy* theory, a distinction is drawn between *convergent* elements *i.e.* those which evaluate to functions from D to D; and *divergent* elements whose evaluations do not terminate, *i.e.* those devoid of any functional (or *operator* as opposed to *operand*) content. That is to say, the following structure map is *partial*,

$$\mathsf{eval} : D \rightharpoonup [D \to D].$$

The standard approach to partial maps in domain theory is to make them into total ones by sending undefined arguments to a "bottom element", thereby changing the type of eval to

$$\mathsf{eval} : D \to [D \to D]_\perp.$$

In the language of the model theory of λ-calculus, eval furnishes the "fun" part of the retraction of $[D \to D]_\perp$ into D.

We regard the initial solution of the domain equation

$$D \cong [D \to D]_\perp$$

(which is non-trivial) as the canonical model of the pure lazy language.

Some Basic Definitions

Let D, E be cpo's. We say that $\langle i, j \rangle$ is an *embedding* of D into E, denoted $D \trianglelefteq E$, if i, j are continuous maps $D \overset{i}{\hookrightarrow} E \overset{j}{\twoheadrightarrow} D$ and that

$$i \circ j \sqsubseteq \mathsf{id}_E \text{ and } j \circ i = \mathsf{id}_D.$$

i is the left-adjoint of j and each uniquely determines the other.

Recall the category-theoretic characterization of lifting as the left adjoint to the forgetful functor U:

$$\mathbf{CPO} \xrightarrow{(-)_\perp} \mathbf{CPO}_\perp \xrightarrow{U} \mathbf{CPO}$$

where $\mathbf{CPO}_\perp$ is the sub-category of strict functions. As is standard, we have:

- A natural transformation: $\mathsf{up} : I_{\mathbf{CPO}} \to U \circ (-)_\perp$.

- For each continuous function $f : D \to UE$, its adjoint

$$\mathsf{lift}(f) : (D)_\perp \to_\perp E.$$

Concretely, we have, for $x, y \in D$:

$$\begin{array}{lll}
(D)_\perp & \stackrel{\text{def}}{=} & \{\perp\} \cup \{\langle 0, d\rangle \mid d \in D\}, \\
x \sqsubseteq y & \stackrel{\text{def}}{=} & x = \perp \quad \text{or} \\
 & & [x = \langle 0, d\rangle \;\&\; y = \langle 0, d'\rangle \;\&\; d \sqsubseteq_D d'], \\
\mathsf{up}_D(d) & \stackrel{\text{def}}{=} & \langle 0, d\rangle, \\
\mathsf{lift}(f)(\perp) & \stackrel{\text{def}}{=} & \perp_E, \\
\mathsf{lift}(f)\langle 0, d\rangle & \stackrel{\text{def}}{=} & f(d).
\end{array}$$

Let $\mathsf{dn}_D \stackrel{\text{def}}{=} \mathsf{lift}(\mathsf{id}_D) : D_\perp \to_\perp D$. For example, $\mathsf{dn}_{[D\to D]}(\langle 0, f\rangle) = f$ for $f \in [D \to D]$.

LEMMA **4.3.1.1** *Suppose $\langle i, j\rangle$ is an embedding of D into D'. Then, $\langle i^*, j^*\rangle$ defined as follows:*

$$i^* \stackrel{\text{def}}{=} \lambda f \in [D \to D]_\perp . \mathsf{C}^{[D\to D]_\perp}_{[D'\to D']_\perp} f \mathsf{up}_{[D'\to D']}[\lambda y \in D'. i \circ \mathsf{dn}_{[D\to D]}(f) \circ j(y)],$$

$$j^* \stackrel{\text{def}}{=} \lambda g \in [D' \to D']_\perp . \mathsf{C}^{[D'\to D']_\perp}_{[D\to D]_\perp} g \mathsf{up}_{[D\to D]}[\lambda y \in D. j \circ \mathsf{dn}_{[D'\to D']}(g) \circ i(y))$$

is an embedding of $[D \to D]_\perp$ into $[D' \to D']_\perp$ where $\mathsf{C}^X_Y : X \times Y \to Y$ is defined as

$$\mathsf{C}^X_Y(x, y) \stackrel{\text{def}}{=} \begin{cases} \perp_Y & \text{if } x = \perp_X; \\ y & \text{else.} \end{cases}$$

PROOF Clearly i^* and j^* are continuous. Also, it is straightforward to check that $i^* \circ j^* \sqsubseteq \mathsf{id}_{[D'\to D']_\perp}$ and $j^* \circ i^* = \mathsf{id}_{[D\to D]_\perp}$. □

4.3.2 Construction of the Initial Solution

Define $D_0 \stackrel{\text{def}}{=} \mathbf{1}$ (the one-point domain) and $i_0 : D_0 \to [D_0 \to D_0]_\perp$ such that $i_0(\perp) = \perp$; and inductively,

$$D_{n+1} \stackrel{\text{def}}{=} [D_n \to D_n]_\perp, \qquad D_n \stackrel{i_n}{\hookrightarrow} D_{n+1} \stackrel{j_n}{\twoheadrightarrow} D_n,$$

$$\langle i_{n+1}, j_{n+1}\rangle \stackrel{\text{def}}{=} \langle i_n^*, j_n^*\rangle.$$

Writing out the embedding pairs in full, we have

$$i_{n+1} \stackrel{\text{def}}{=} \lambda f \in D_{n+1}.\mathsf{C}_{D_{n+2}}^{D_{n+1}} f \mathsf{up}_{[D_{n+1}\to D_{n+1}]}[\lambda y \in D_{n+1}.i_n \circ \mathsf{dn}_{[D_n\to D_n]}(f) \circ j_n(y)],$$

$$j_{n+1} \stackrel{\text{def}}{=} \lambda g \in D_{n+2}.\mathsf{C}_{D_{n+1}}^{D_{n+2}} g \mathsf{up}_{[D_n\to D_n]}[\lambda y \in D_n.j_n \circ \mathsf{dn}_{[D_{n+1}\to D_{n+1}]}(g) \circ i_n(y)).$$

$\langle D_n, j_n\rangle_{n\in\omega}$ is an inverse system of cpos which are ω-algebraic complete lattices.

We define the *inverse limit*, $D = \mathsf{lim}_{\leftarrow}\langle D_n, j_n\rangle_{n\in\omega}$ as follows: Define, as is standard, $\phi_{m,n} : D_n \to D_m$ by

$$\begin{aligned} m \geqslant n \quad & \phi_{n,n} = \lambda x \in D_n.x, \\ & \phi_{m+1,n} = \phi_{m,n} \circ j_m; \\ m < n \quad & \phi_{m,n+1} = i_n \circ \phi_{m,n}. \end{aligned}$$

and identify the initial solution $D \subseteq \prod_{n\in\omega} D_n$ as

$$D = \{\, \langle x_n : n \in \omega\rangle : x_n \in D_n \;\&\; j_n(x_{n+1}) = x_n \,\}$$

and name the isomorphism pair as

$$D \xrightarrow{\mathsf{Fun}} [D \to D]_\perp \xrightarrow{\mathsf{Gr}} D;$$

the precise definition of Fun and Gr will be spelt out in full in the sequel. Note that direct and inverse limits coincide in Domain Theory (see [SP82]), *i.e.*

$$D = \mathsf{lim}_{\rightarrow}\langle D_n, i_n\rangle_{n\in\omega}.$$

Observe that D is a quasi-applicative transition system (see Chapter 3) with the associated evaluation map $\mathsf{eval} : D \rightharpoonup D^D$ defined as:

$$\mathsf{eval}(d) \stackrel{\text{def}}{=} \begin{cases} f & \text{if } \mathsf{Fun}(d) = \langle 0, f\rangle, \\ \text{undefined} & \text{if } \mathsf{Fun}(d) = \perp. \end{cases}$$

Thus, we may write $d \Downarrow f$ and $d\Uparrow$ etc. Equivalently, we may recast the above as a quasi-applicative structure with divergence: $\langle D, \cdot, \Uparrow\rangle$ where

$$d \cdot e \stackrel{\text{def}}{=} \begin{cases} f(e) & \text{if } d \Downarrow f, \\ \perp & \text{if } d\Uparrow. \end{cases}$$

In the following, we will work out the definition of the application in terms of projections onto approximants.

NOTATION **4.3.2.1** For simplicity, henceforth, we will write the *n-projection map* $\phi_{\infty,n}$ as ψ_n and the *n-injection map* $\phi_{n,\infty}$ as ϕ_n. Also, we will regard each D_n as a subset of D, *i.e.* we identify $\phi_n(x)$ where $x \in D_n$ with x; and that for $x \in D, \psi_n(x) = x_n \in D_n$. Hence,

$$D = \bigcup_{n\in\omega} D_n.$$

4.3.3 Index Calculations and Extensionality Properties

We will unpack the structure of D somewhat and highlight some basic index-calculation results. These will be used to spell out the definitions of the application operation, and the Fun and Gr functions of D. Finally, we present the extensionality properties of D.

LEMMA **4.3.3.1** *Let $x \in D$. Then,*

(i) $(x_n)_m = x_{min(n,m)}$.

(ii) If $n \leqslant m$, then $x_n \sqsubseteq x_m \sqsubseteq x$.

(iii) $x = \bigsqcup_n x_n$.

(iv) If $x \in D_n$, then $\forall m \geqslant n.x_m = x$.

PROOF We omit the largely straightforward proofs. □

COROLLARY **4.3.3.2** *Let $d \in D$. Then, (note that in D, $d\Uparrow \iff d = \bot$)*

(i) $d\Uparrow \iff \forall n \geqslant 0.d_n\Uparrow$.

(ii) $d\Downarrow \iff \forall n \geqslant 1.d_n\Downarrow$.

PROOF (i): by (iii) of the Lemma. (ii): "$\Leftarrow$" is clear from (iii) of the Lemma. For "$\Rightarrow$", in view of (ii) of Lemma, it suffices to show that $d\Downarrow \Rightarrow d_1\Downarrow$. Suppose the contrary, *i.e.* $d_1\Uparrow$. Since $j_n(d_{n+1}) = d_n$, it then follows from the definition of j_n that $\forall n \in \omega.d_n\Uparrow$ which, by an appeal to (iii) of Lemma, leads to a contradiction. □

Define, for $n \in \omega$, $\mathsf{Ap}_n^\perp : [D_n \to D_n]_\perp \times D_n \to D_n$ by

$$\mathsf{Ap}_n^\perp(f,d) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } f = \bot, \\ g(d) & \text{if } f = \langle 0, g\rangle \text{ with } g \in [D_n \to D_n]. \end{cases}$$

In the following, however, we will abbreviate $\mathsf{Ap}_n^\perp(f,d)$ simply as $f(d)$.

LEMMA **4.3.3.3** *For $n \leqslant k \in \omega$,*

(i) $x_{n+1}(y_n) \sqsubseteq x_{k+1}(y_k)$.

(ii) $(x_{n+1})_{k+1}y_k = x_{n+1}(y_n)$.

(iii) $(x_{k+1}(y_n)_k)_n = x_{n+1}(y_n)$.

PROOF

(i) If $x_{n+1} = \bot$, then $x_{n+1}(y_n) = \bot \sqsubseteq x_{k+1}(y_k)$. Now suppose $x_{n+1} \neq \bot$.

$$\begin{aligned}
x_{n+1}(y_n) &= j_{n+1}(x_{n+2})(j_n(y_{n+1})) && x_{n+2}\Downarrow \\
&= j_n \circ x_{n+2} \circ i_n \circ j_n(y_{n+1}) && i_n \circ j_n \sqsubseteq \mathrm{id} \\
&\sqsubseteq j_n \circ x_{n+2}(y_{n+1}) && x \in D_{n+1} \Rightarrow x_n = j_n(x) \sqsubseteq x \\
&\sqsubseteq x_{n+2}(y_{n+1}).
\end{aligned}$$

(ii) By induction on $k \geqslant n$. Trivial if $k = n$. Consider the case $k+1$. Wlog, assume $x_{n+1}\Downarrow$(equivalently, $(x_{n+1})_{k+2}\Downarrow$). Note that if $x \in D_p, p \leqslant q$ then $i_q(x_q) = x_{q+1}$.

$$\begin{aligned}
(x_{n+1})_{k+2}y_{k+1} &= (i_{k+1}((x_{n+1})_{k+1}))y_{k+1} && \text{definition of } i_{k+1} \\
&= (i_k \circ (x_{n+1})_{k+1} \circ j_k)(y_{k+1}) \\
&= i_k \circ (x_{n+1})_{k+1}(y_k) && x \in D_k \quad i_k(x) = x \\
&= (x_{n+1})_{k+1}(y_k) && \text{induction hypothesis} \\
&= x_{n+1}(y_n).
\end{aligned}$$

(iii) By induction on k. $k = n$ is trivial. Wlog, assume $x_{k+2}\Downarrow$.

$$\begin{aligned}
(x_{k+2}(y_n)_{k+1})_n &= \phi_{k+1,n}(x_{k+2}(y_n)_{k+1}) && y_n \in D_n, n \leqslant k \\
&= \phi_{k,n} \circ j_k \circ x_{k+2} \circ i_k(y_n)_k && \text{definition of } j_{k+1} \\
&= \phi_{k,n} \circ (j_{k+1}(x_{k+2}))(y_n)_k \\
&= (x_{k+1}(y_n)_k)_n && \text{induction hypothesis} \\
&= x_{n+1}(y_n).
\end{aligned}$$

□

Now, we are ready to spell out the binary application "$\cdot$": $D \times D \to D$

$$x \cdot y \stackrel{\text{def}}{=} \bigsqcup_{n \in \omega} \mathsf{Ap}_n^{\perp}(x_{n+1}, y_n).$$

It should be clear that "$\cdot$" is continuous. In fact, we can say more, that "$\cdot$" *left-preserves arbitrary joins*, of which more anon.

Note that if $x \in D_{n+1}, y \in D_n$, then

$$\begin{aligned} x \cdot y &= x_{n+1} \cdot y_n \\ &= \bigsqcup_{i\in\omega}(x_{n+1})_{i+1}(y_n)_i && \text{Lemma 4.3.3.3(ii)} \\ &= \bigsqcup_{i\leqslant n} x_{i+1}y_i && \text{Lemma 4.3.3.3(i)} \\ &= x_{n+1}(y_n). \end{aligned}$$

PROPOSITION **4.3.3.4** *Let $x, y \in D$.*

(i) $x_{n+1} \cdot y = x_{n+1} \cdot y_n = (x \cdot y_n)_n$.

(ii) $x_1 \cdot y = \bot$.

(iii) $x_0 \cdot y = x_0 = \bot$.

PROOF

(i) We have

$$\begin{aligned} x_{n+1} \cdot y &= \bigsqcup_{i\in\omega}(x_{n+1})_{i+1}(y_i) && \text{Lemma 4.3.3.3(ii)} \\ &= \bigsqcup_{i\leqslant n}(x_{n+1})_{i+1}(y_i) && \text{Lemma 4.3.3.3(i)} \\ &= x_{n+1}(y_n) && \text{remark above} \\ &= x_{n+1} \cdot y_n. \end{aligned}$$

$$\begin{aligned} (x \cdot y_n)_n &= (\bigsqcup_{i\in\omega} x_{i+1}(y_n)_i)_n && \text{Continuity of } \psi_n \\ &= \bigsqcup_{i\in\omega}(x_{i+1}(y_n)_i)_n && \text{Lemma 4.3.3.3(iii)} \\ &= \bigsqcup_{i\leqslant n}(x_{i+1}(y_n)_i)_n \\ &= \bigsqcup_{i\leqslant n} x_{i+1}(y_n)_i \\ &= \bigsqcup_{i\leqslant n} x_{i+1}(y_i) && \text{Lemma 4.3.3.3(i)} \\ &= x_{n+1} \cdot y_n. \end{aligned}$$

(ii) $x_1 \cdot y = x_1 \cdot y_0$ (by (i)) $= x_1(\bot) = \bot$.

(iii) $\forall n \in \omega.\mathsf{Ap}^{\perp}_n(\bot, y) = \bot$.

□

In the same way as $\mathsf{Ap}_n^\perp$, we define $\mathsf{Ap}^\perp : [D \to D]_\perp \times D \to D$ as

$$\mathsf{Ap}^\perp(f,d) \stackrel{\text{def}}{=} \begin{cases} g(d) & \text{if } f = \langle 0, g\rangle \text{ with } g \in [D \to D], \\ \perp & \text{if } f = \perp. \end{cases}$$

PROPOSITION **4.3.3.5** *Define, for* $f \in [D \to D]_\perp$

$$\mathsf{rep}(f) \stackrel{\text{def}}{=} \begin{cases} \bigsqcup_{n\in\omega} \mathsf{up}_{[D_n \to D_n]}[\lambda y \in D_n.(\mathsf{Ap}^\perp(f,y))_n] & \text{if } f = \langle 0, g\rangle; \\ \perp & \text{if } f = \perp. \end{cases}$$

Then, $\forall y \in D.\mathsf{Ap}^\perp(f,y) = \mathsf{rep}(f)\cdot y$*; in which case, we say that* f *is representable by* $\mathsf{rep}(f)$.

PROOF Note that all joins exist in D. Wlog suppose $f \neq \perp$.

$$\begin{aligned} \mathsf{rep}(f)\cdot y &= \bigsqcup_{i\in\omega} \mathsf{rep}(f)_{i+1}(y)_i \\ &= \bigsqcup_{i\in\omega} (\mathsf{rep}(f)\cdot y_i)_i \\ &= \bigsqcup_i ((\bigsqcup_n \mathsf{up}_{[D_n\to D_n]}[\lambda y \in D_n.(g(y))_n])\cdot y_i)_i \\ &= \bigsqcup_{i,n} (\mathsf{up}_{[D_n\to D_n]}[\lambda y \in D_n.(g(y))_n]\cdot y_i)_i \\ &= \bigsqcup_{j\in\omega} (\mathsf{up}_{[D_n\to D_n]}[\lambda y \in D_j.(g(y))_j]\cdot y_j)_j \\ &= \bigsqcup_i (g(y_j))_j \\ &= \bigsqcup_n \bigsqcup_i (g(y_i))_n \\ &= \bigsqcup_i g(y_i) \\ &= g(y). \end{aligned}$$

□

We can now spell out the maps (which are both isomorphic and homeomorphic) between D and $[D \to D]_\perp$:

$$D \xrightarrow{\mathsf{Fun}} [D \to D]_\perp \xrightarrow{\mathsf{Gr}} D$$

where

$$\mathsf{Fun}(d) \stackrel{\text{def}}{=} \begin{cases} \mathsf{up}(x \mapsto d\cdot x) & \text{if } d \neq \perp; \\ \perp & \text{else:} \end{cases}$$

and $\mathsf{Gr}(f) \stackrel{\text{def}}{=} \mathsf{rep}(f)$. Note that,

$$x \cdot y = \mathsf{Ap}^{\perp}(\mathsf{Fun}(x), y).$$

It follows from Proposition 4.3.3.5 that

$$\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}_{[D \to D]_{\perp}}.$$

To show $\mathsf{Gr} \circ \mathsf{Fun} = \mathsf{id}_D$, observe first that $\mathsf{Gr}(\mathsf{Fun}(\perp)) = \mathsf{Gr}(\perp) = \perp$. Suppose $d \neq \perp$. Then,

$$\begin{aligned}
\mathsf{Gr}(\mathsf{Fun}(d)) &= \mathsf{Gr}(\langle 0, \lambda x.d \cdot x\rangle) \\
&= \bigsqcup_{n\in\omega} \mathsf{up}_{[D_n \to D_n]}[\lambda y \in D_n.(d \cdot y)_n] \qquad y_n = y \\
&= \bigsqcup_{n\in\omega} \mathsf{up}_{[D_n \to D_n]}[\lambda y \in D_n.(d_{n+1} \cdot y)] \\
&= \bigsqcup_{n\in\omega} \mathsf{up}_{[D_n \to D_n]}[\lambda y \in D_n.d_{n+1}(y)] \\
&= \bigsqcup_{n\in\omega} d_{n+1} \\
&= d.
\end{aligned}$$

Hence, $\mathsf{Gr} \circ \mathsf{Fun} = \mathsf{id}_D$.

The following Corollary states that representable functions of D coincide with its continuous functions.

COROLLARY **4.3.3.6** *Let $f \in D^D$. Then, f is continuous iff f is $\Downarrow$-representable, i.e. $\exists d \in D.d\Downarrow$ & $\forall x \in D.f(x) = d \cdot x$.*

PROOF Let $f \in D^D$, then the Proposition asserts that $\mathsf{rep}(\mathsf{up}(f))$ is a representative of f. Conversely, any *convergent* $d \in D$ is a representative of a continuous map, namely, $\mathsf{dn}(\mathsf{Fun}(d))$. □

Extensionality Properties of D

D does not satisfy the strong extensionality principle[3] — just consider $\perp$ and $\perp_1$ where $\perp_1 \stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}(\lambda x.\perp))$ and that $f_{d,e}$ is the standard *step function* defined as

$$f_{d,e}(x) \stackrel{\text{def}}{=} \begin{cases} e & \text{if } d \sqsubseteq x; \\ \perp & \text{else.} \end{cases}$$

[3]Strong extensionality principle is the following:

$$\forall x, y \in D.[\forall z \in D.x \cdot z = y \cdot z \Rightarrow x = y.$$

However, D satisfies the weaker notion which we call

PROPOSITION **4.3.3.7 (Conditional Strong Extensionality)** *For $d, e \in D$,*

$$d{\Downarrow} \;\&\; e{\Downarrow} \Rightarrow [\forall x \in D. d \cdot x \sqsubseteq e \cdot x \Rightarrow d \sqsubseteq e].$$

PROOF Suppose $d{\Downarrow}$ and $e{\Downarrow}$ and $\forall x \in D. d \cdot x \sqsubseteq e \cdot x$. Let $\mathsf{Fun}(d) = \langle 0, f\rangle$ and $\mathsf{Fun}(e) = \langle 0, g\rangle$ with $f, g \in [D \to D]$. Then, $\forall x. f(x) \sqsubseteq g(x)$; and so, $f \sqsubseteq g$ which implies $\langle 0, f\rangle \sqsubseteq \langle 0, g\rangle$. By monotonicity of Gr and that $\mathsf{Gr} \circ \mathsf{Fun} = \mathsf{id}$,

$$d = \mathsf{Gr}(\langle 0, f\rangle) \sqsubseteq \mathsf{Gr}(\langle 0, g\rangle) = e.$$

□

We prove a result due to Samson Abramsky.

THEOREM **4.3.3.8 (Internal Full Abstraction)** *D is internally fully abstract i.e. $\forall d, e \in D. d \sqsubseteq e \iff d \lesssim^B e$.*

PROOF Unpacking the definitions, we see that for all $d, e \in D$:

$$d \sqsubseteq e \iff d{\Downarrow} f \Rightarrow [e{\Downarrow} g \;\&\; \forall c \in D. f(c) \sqsubseteq g(c)].$$

Thus the domain ordering is an applicative bisimulation, and so is included in $\lesssim^B$. For the converse, we prove a stronger statement. Wlog, suppose $d{\Downarrow}$ and $e{\Downarrow}$. CLAIM: $\forall d, e \in D. \forall k \in \omega. d \lesssim^B_k e \Rightarrow d_k \sqsubseteq e_k$ which clearly implies $d \lesssim^B e \Rightarrow d \sqsubseteq e$ since $d = \bigsqcup_{k\in\omega} d_k$. The base case $k = 0$ is trivial. Suppose true for k. Wlog, assume $d{\Downarrow}$ and $e{\Downarrow}$. Then, let $d \lesssim^B_{k+1} e$. By definition,

$$\forall f \in D. df_k \lesssim^B_k ef_k.$$

Invoking the induction hypothesis, we have

$$\forall f \in D. (df_k)_k \sqsubseteq (ef_k)_k,$$

which implies $\forall f \in D. d_{k+1} f \sqsubseteq e_{k+1} f$. Now, $d_{k+1}{\Downarrow}$ and $e_{k+1}{\Downarrow}$, the result that follows from the above Proposition. □

As a corollary, we see that D is an aswd. Define an interpretation of λ-terms in D as follows: for $d \in D$ and $M \in \Lambda$,

$$\begin{aligned}
[\![x]\!]_\rho &\stackrel{\text{def}}{=} \rho(x),\\
[\![\underline{d}]\!]_\rho &\stackrel{\text{def}}{=} d,\\
[\![MN]\!]_\rho &\stackrel{\text{def}}{=} [\![M]\!]_\rho \cdot [\![N]\!]_\rho,\\
[\![\lambda x.M]\!]_\rho &\stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}_{[D\to D]}[d \mapsto [\![M]\!]_{\rho[x:=\underline{d}]}]).
\end{aligned}$$

It is easy to see that D is a lts.

4.3.4 λC-Definability of the Projection Maps ψ_n

Is there a closed λ-term X that discriminates between the convergent and divergent λ-terms? *i.e.*

$$\forall M \in \Lambda^o. \begin{cases} XM = \mathbf{I} & \text{if } M\Downarrow, \\ XM\Uparrow & \text{if } M\Uparrow; \end{cases}$$

(**I** is the identity). The answer is no.

LEMMA **4.3.4.1** *Let $X \in \Lambda^o$, $\bot$ be any divergent (i.e. strongly unsolvable) λ-term and $\bot_1$ any* least *convergent term with respect to the bisimulation ordering, say, $\bot_1 \stackrel{\text{def}}{=} \lambda x.\Theta$ for any $\Theta\Uparrow$. Then, $X\bot\Uparrow \,\&\, X\bot_1\Downarrow \Rightarrow X =_\beta \mathbf{I}$.*

PROOF Let X satisfy the antecedent. Then, clearly, $X \notin \mathcal{F}_2$[4]. Whence, and because X is closed, $X =_\beta \lambda x.x\vec{P}$. But, if $|\vec{P}| \geqslant 1$, then $X\bot_1\Uparrow$. Hence, $X =_\beta \mathbf{I}$. □

More generally, we have the following definition.

DEFINITION **4.3.4.2** (**Abramsky**) Let $\mathcal{A} = \langle A, \cdot, \Uparrow\rangle$ be an aswd (or equivalently, an ats) and that x, y range over A. *Convergence testing* is definable in $\mathcal{A}$ if for some $c \in A$, $\mathcal{A}$ satisfies:

- $c\Downarrow$,
- $x\Uparrow \Rightarrow cx\Uparrow$,
- $x\Downarrow \Rightarrow cx = \mathbf{I}$.

Convergence testing is definable in D by $c \stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}(f_{\bot_1,i}))$ where $i \equiv [\![\lambda x.x]\!]$ and $\bot_1 \stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}(\lambda x.\bot))$.

DEFINITION **4.3.4.3** Let C be a set of constants with prescribed interpretations in D. For any function $f \in D^D$, we say that f is *λC-definable* in D if

$$\exists M \in \Lambda(C)^o.\forall x \in D.f(x) = [\![M]\!] \cdot x.$$

OBSERVATION **4.3.4.4** Let C be interpreted as the convergence testing c in D. The embedding pairs, $\langle(i_n, j_n)\rangle_{n\in\omega}$, are λC-definable in D by the following λC-terms:

$$\mathrm{i}_{n+1} \stackrel{\text{def}}{=} \lambda f.\mathrm{C}f(\lambda y.\mathrm{i}_n(f(\mathrm{j}_n y))),$$
$$\mathrm{j}_{n+1} \stackrel{\text{def}}{=} \lambda g.\mathrm{C}g(\lambda y.\mathrm{j}_n(g(\mathrm{i}_n y)));$$

with $\mathrm{i}_0 = \mathbf{I}$ and $\mathrm{j}_0 = \lambda x.\bot$. □

[4]Recall that, for $n \in \omega$, $M \in \mathcal{F}_n$ if $\exists N \in \Lambda.\lambda\beta \vdash M = \lambda x_1 \cdots x_n.N$

Given a sequence of functions $\langle p_n : D \to D : n \in \omega \rangle$ defined inductively as follows: $p_0 = \lambda x.\bot$ and

$$(\dagger) \qquad \forall d \in D.\forall n \in \omega.j_n(p_{n+1}d) = p_n(d).$$

Is it true that $\forall n \in \omega.p_n = \psi_n$? Recall that $\psi_n : D \to D_n$ is the canonical projection function from D to D_n.

An extra condition is needed:

$$\forall e \in D.\forall n \in \omega.p_{n+1}(d) \cdot e = p_n(d \cdot p_n(e)).$$

Clearly, to show $\forall n \in \omega.p_n = \psi_n$, it suffices to show

$$\forall n \in \omega.\forall d \in D.p_n(d) = d_n = \psi_n(d);$$

and this we will show by induction on n.

Base case is valid by premise. Now, suppose for some $n \geqslant 1$, $\forall d \in D.p_n(d) = d_n$. If $d\uparrow$, by induction hypothesis $p_n = \psi_n$, hence $p_n(d) = \bot$, and so, by $(\dagger)$, we must have $\forall m > n.p_m(d) = \bot$. Now, suppose $d\Downarrow$. Then, $d_{n+1}\Downarrow$ and $p_{n+1}(d)\Downarrow$. For any $e \in D$,

$$\begin{aligned} d_{n+1} \cdot e &= (d \cdot e_n)_n && \text{Induction Hypothesis} \\ &= p_n(d \cdot p_n(e)) && (\dagger) \\ &= p_{n+1}(d) \cdot e. \end{aligned}$$

By conditional strong extensionality of D, we have $d_{n+1} = p_{n+1}(d)$. Let us crystalize the result as follows:

LEMMA **4.3.4.5** *Let* $\langle p_n : D \to D : n \in \omega \rangle$ *and* $d, e \in D$ *such that*

(1) $p_0 = \lambda x.\bot$.

(2) $\forall n \in \omega.j_n(p_{n+1}d) = p_n d$.

(3) $\forall n \in \omega.p_{n+1}(d) \cdot e = p_n(d \cdot p_n(e))$.

Then, $\forall n \in \omega.p_n = \psi_n$. □

We will now apply the Lemma to show that the canonical projection functions $\langle \psi_n \rangle_{n \in \omega}$ are λC-definable.

DEFINITION **4.3.4.6** We define inductively a sequence of λC-terms, $\langle \Psi_n \rangle_{n \in \omega}$, as follows:

$$\Psi_0 \stackrel{\text{def}}{=} \lambda x.\bot,$$
$$\Psi_{n+1} \stackrel{\text{def}}{=} \lambda x.\mathrm{C}x(\lambda y.\Psi_n(x(\Psi_n y))).$$

LEMMA **4.3.4.7** *(i)* $\forall n \in \omega \forall d \in D.j_n([\![\Psi_{n+1}]\!] \cdot d) = [\![\Psi_n]\!] \cdot d.$
(ii) $\forall n \in \omega.\forall d \in D_n.[\![\Psi_{n+1}]\!] \cdot (i_n d) = [\![\Psi_n]\!] \cdot d.$
(iii) $\forall n \in \omega.\forall d, e \in D.([\![\Psi_{n+1}]\!] \cdot d) \cdot e = [\![\Psi_n]\!] \cdot (d \cdot ([\![\Psi_n]\!] \cdot e)).$

PROOF We will omit the decorated brackets in $[\![\Psi_n]\!]$ and write $d \cdot e$ simply as de in the proof for easy reading. (i) and (ii) are proved by simultaneous induction on n. Wlog, suppose $d\Downarrow$, for the cases for divergent d are readily seen to be true. Base cases are clearly valid. Suppose (i) and (ii) true for $n-1$ for some $n \geqslant 1$. For (i),

$$\begin{aligned}
j_n(\Psi_{n+1}d) &= j_n[(\lambda x.\mathrm{C}x(\lambda y.\Psi_n(x(\Psi_n y))))d] \\
&= j_n(\lambda y.\Psi_n(d(\Psi_n y))) && \text{definition of } j_n \\
&= \lambda x \in D_{n-1}.j_{n-1}(\Psi_n(d(\Psi_n((i_{n-1}x))))) && \text{ind. hyp. (i),(ii)} \\
&= \lambda x \in D_{n-1}.\Psi_{n-1}(d(\Psi_{n-1}x)) && \text{definition of } \Psi_n \\
&= \Psi_n d.
\end{aligned}$$

For (ii), let $d \in D_n$. Then,

$$\begin{aligned}
\Psi_{n+1}(i_n d) &= \lambda y.\Psi_n((i_n d)(\Psi_n y)) && \text{definition of } i_n \\
&= \lambda y.\Psi_n(i_{n-1}(d(j_{n-1}(\Psi_n y)))) && \text{ind. hyp. (ii)} \\
&= \lambda y.\Psi_{n-1}(d(j_{n-1}(\Psi_n y))) && \text{ind. hyp. (i)} \\
&= \lambda y.\Psi_{n-1}(d(\Psi_{n-1}y)) && \text{definition of } \Psi_n \\
&= \Psi_n d.
\end{aligned}$$

(iii) is easy: $(\Psi_{n+1}d)e = (\lambda y.\Psi_n(d(\Psi_n y)))e = \Psi_n(d(\Psi_n e))$. □

Applying Lemma 4.3.4.5, we establish the following result:

PROPOSITION **4.3.4.8** *The canonical projection maps $\psi_n : D \to D_n$ are λC-definable; i.e. $\forall n \in \omega.\forall d \in D.\psi_n(d) = [\![\Psi_n]\!] \cdot d.$* □

4.3.5 Application Preserves Arbitrary Joins

Next, we establish an important property satisfied by the application operation of D: "$\cdot$" *left-preserves arbitrary joins*, a result on which our later application depends. This is a consequence of the *coincidence* of representable and continuous functions of D. First, a technical Lemma.

LEMMA **4.3.5.1** *Let I and J be arbitrary indexing sets and $\mu : I \times J \to D$ and that D is a complete lattice. Then,*

$$\bigsqcup_{i\in I}\bigsqcup_{j\in J}\mu(i,j) = \bigsqcup_{j\in J}\bigsqcup_{i\in I}\mu(i,j).$$

PROOF Observe that all joins exist in D. For any j_0, we have

$$\forall i \in I.\mu(i,j_0) \sqsubseteq \bigsqcup_{j\in J}\mu(i,j).$$

Hence, $\bigsqcup_{i\in I}\mu(i,j_0) \sqsubseteq \bigsqcup_{i\in I}\bigsqcup_{j\in J}\mu(i,j)$. And finally,

$$\bigsqcup_{j\in J}\bigsqcup_{i\in I}\mu(i,j) \sqsubseteq \bigsqcup_{i\in I}\bigsqcup_{j\in J}\mu(i,j).$$

The other direction is entirely symmetrical. □

PROPOSITION **4.3.5.2** *In D, the application "$\cdot$" left-preserves arbitrary joins, i.e.*

$$\forall X \subseteq D.\forall d \in D.(\bigsqcup X)\cdot d = \bigsqcup_{x\in X} x\cdot d.$$

PROOF Let $X \subseteq D$. Wlog, assume that X contains a convergent element. Define a function $F_X : D \to D$ as: for $y \in D$.

$$F_X(y) \stackrel{\text{def}}{=} \bigsqcup_{x\in X} x\cdot y.$$

$F_X(y)$ exists for all $y \in D$ because D is a complete lattice. F_X is monotonic: Let $y_1 \sqsubseteq y_2$. Then $\forall x \in X, x\cdot y_1 \sqsubseteq x\cdot y_2$. Hence $\bigsqcup_{x\in X} x\cdot y_1 \sqsubseteq \bigsqcup_{x\in X} x\cdot y_2$.
Further, F_X is continuous. To see this, let $Y \subseteq D$ be directed. Then,

$$\begin{aligned}
F_X(\bigsqcup Y) &= \bigsqcup_{x\in X} x\cdot(\bigsqcup_{y\in Y} y) && \text{"$\cdot$" is continuous}\\
&= \bigsqcup_{x\in X}(\bigsqcup_{y\in Y} x\cdot y) && \text{Lemma 4.3.5.1}\\
&= \bigsqcup_{y\in Y}(\bigsqcup_{x\in X} x\cdot y)\\
&= \bigsqcup_{y\in Y} F_X(y).
\end{aligned}$$

By the coincidence of representable and continuous function spaces of D,

$$\exists \Box \in D.\Box\Downarrow \,\&\, [\forall y \in D.\Box \cdot y = F_X(y)].$$

We aim to show that $\Box = \bigsqcup X$. Now,

$$\forall y \in D.\forall x \in X.x \cdot y \sqsubseteq (\bigsqcup_{x \in X} x) \cdot y,$$

$$\forall y \in D.F_X(y) = \bigsqcup_{x \in X} (x \cdot y) \sqsubseteq (\bigsqcup_{x \in X} x) \cdot y.$$

Note that $\Box\Downarrow$ and $\bigsqcup X\Downarrow$, hence by conditional strong extensionality of D, we conclude

$$\Box \sqsubseteq \bigsqcup_{x \in X} x.$$

But $\forall x \in X.\forall y \in D.x \cdot y \sqsubseteq \bigsqcup_{x \in X}(x \cdot y) = \Box \cdot y$. Hence, $\forall x \in X.x \sqsubseteq \Box$; from which we conclude $\bigsqcup X \sqsubseteq \Box$. Hence, we have shown $\Box = \bigsqcup X$ and we are done. □

4.4 Lazy Lambda Calculus with Convergence Testing

This section introduces the proof system $\lambda\beta\mathsf{C}$ and the theory $\lambda\ell_{\mathsf{C}}$. Both are obtained by extending the by now familiar logical entities $\lambda\beta$ and $\lambda\ell_{\mathsf{C}}$ respectively with convergence testing constant C in the natural way. $\lambda\beta\mathsf{C}$ is shown to be *Church-Rosser* and a *Standardization Theorem* for the associated $\beta\mathsf{C}$-reduction is proved. We present Abramsky's results for the theory $\lambda\ell_{\mathsf{C}}$. Plotkin's problem [Plo75] of *simulating* call-by-value evaluation in a call-by-name regime is revisited. We show that there is a translation of terms from Λ to $\Lambda(\mathsf{C})$ which preserves call-by-value convergence exactly.

4.4.1 Pure Lazy Language with Convergence Testing $\lambda\ell_{\mathsf{C}}$

As part of his study on the problem of full abstraction, Abramsky introduced an extended lazy language which is obtained by augmenting the pure lazy language by a constant called *convergence testing* in [Abr87, Chapter 6].

CONVENTION **4.4.1.1** By convention, a *reduction* consists of a (possibly empty) sequence of *one-step* reductions. We emphasize that the word reduction is used in this thesis in an *iterated* sense, *i.e.* to describe a reflexively and transitively closed binary relation.

DEFINITION **4.4.1.2** Define a binary reduction relation $\Downarrow_{\mathsf{C}}$ on $\Lambda(\mathsf{C})^o$ by the following rules:

$$(\mathsf{C}\Downarrow_{\mathsf{C}}1)\frac{}{\mathsf{C}\Downarrow_{\mathsf{C}}\mathsf{C}} \qquad (\mathsf{C}\Downarrow_{\mathsf{C}}2)\frac{M\Downarrow_{\mathsf{C}}}{\mathsf{C}M\Downarrow_{\mathsf{C}}\mathbf{I}} \qquad (\mathrm{abs}\Downarrow_{\mathsf{C}})\frac{}{\lambda x.P\Downarrow_{\mathsf{C}}\lambda x.P}$$

$$(\beta_N)\frac{M\Downarrow_{\mathsf{C}}\lambda x.P \qquad P[x:=Q]\Downarrow_{\mathsf{C}}N}{MQ\Downarrow_{\mathsf{C}}N}.$$

The constant C is called *convergence testing*. Since C is not λ-definable, the above definition is clearly not superfluous. As before, we define convergence and divergence predicates, $\Downarrow_{\mathsf{C}}$ and $\Uparrow_{\mathsf{C}}$ respectively. Plainly, $\lambda\ell_{\mathsf{C}} \stackrel{\text{def}}{=} \langle\Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}}\rangle$ is a q-aswd. In fact, it is an aswd — the validity of the axiom (aswd) follows from Corollary 4.4.1.5. Hence we can define the corresponding bisimulation preorder and obtain an (in)equational theory which we denote $\lambda\ell_{\mathsf{C}}$. (As is the case with $\lambda\ell$, we denote the language $\langle\Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}}\rangle$ and the (in)equational theory induced by the associated bisimulation equivalence $\sim^{\mathsf{C}}$ by the same symbol $\lambda\ell_{\mathsf{C}}$.) We will write the associated bisimulation relation of $\langle\Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}}\rangle$ as $\precsim^{\mathsf{C}}$ and the conjunction of $M \precsim^{\mathsf{C}} N$ and $N \precsim^{\mathsf{C}} M$ as $M \sim^{\mathsf{C}} N$.

We define $\Downarrow_{\mathsf{x}}$, a binary reduction relation which is the extension of $\Downarrow_{\mathsf{C}}$ to the whole of $\Lambda(\mathsf{C})$ in the obvious way, *i.e.* $\Downarrow_{\mathsf{x}}$ is defined by exactly the same four rules above, except that the terms appearing in the rules now range over all (possibly open) λC-terms. The associated convergence and divergence predicates are denoted $\Downarrow_{\mathsf{x}}$ and $\Uparrow_{\mathsf{x}}$ respectively.

REMARK **4.4.1.3** (i) It should be clear that $\Downarrow_{\mathsf{C}}$ is a conservative extension of $\Downarrow$.

(ii) However, $\lambda\ell_{\mathsf{C}}$ is not a conservative extension of $\lambda\ell$. To see why this is so, just consider the same M and N in Theorem 4.5.1.1 (Non Full Abstraction 1).

(iii) $\lambda\ell_{\mathsf{C}}$ is a "λ-theory". That is to say, $\forall M, N \in \Lambda.\lambda\beta \vdash M = N \Rightarrow M \sim^{\mathsf{C}} N$.

THEOREM **4.4.1.4 (Contextual Equivalence)** *Let* $M, N \in \Lambda(\mathsf{C})^o$. *Then*

$$M \precsim^{\mathsf{C}} N \iff \forall C[\,] \in \Lambda(\mathsf{C})^o.C[M]\Downarrow_{\mathsf{C}} \Rightarrow C[N]\Downarrow_{\mathsf{C}}.$$

PROOF See [Abr87, Theorem 6.6.11, page 210]. □

COROLLARY **4.4.1.5** $\precsim^{\mathsf{C}}$ *is a pre-congruence, i.e.*

$$\forall M, N \in \Lambda(\mathsf{C}).M \precsim^{\mathsf{C}} N \Rightarrow \forall C[\,] \in \Lambda(\mathsf{C})^o.C[M] \precsim^{\mathsf{C}} C[N];$$

whence $\lambda\ell_{\mathsf{C}}$ *is a congruence.*

PROOF Immediate. □

DISCUSSION **4.4.1.6** C corresponds, strictly speaking, only to a *semi-decision procedure* as divergence cannot be *finitely* observed. Note that the application operation in the pure lazy language $\langle \Lambda^o, \Downarrow \rangle$ is left-strict but not right-strict. The introduction of C enables a version of application which is both left and right strict (as in, say, the application in Plotkin's call-by-name language in [Plo75]) to be *simulated* in $\langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{C}} \rangle$. This result will be formalized and proved later in this section.

$\lambda\ell_{\mathsf{C}}$ is a natural extension of $\lambda\ell$ to investigate. From a theoretical point of view, the introduction of convergence testing *complements* the lazy regime. In fact, the assumption of the definability of C (in the form of specially introduced convergence testing constant extraneous to the underlying language) is crucial to the main characterization results we prove in this thesis:

1. The category-theoretic interpretation of $\lambda_{L,\mathsf{C}}$ — the class of *formal lazy* λ-*calculi* in which convergence testing is definable — in partial categories is sound and complete (see Chapter 5).
2. There is a *retract* of D with respect to which $\lambda\ell_{\mathsf{C}}$ is a *fully abstract model*, a main result of this Chapter.

The introduction of C is reasonable from a functional programming point of view; for it is not uncommon for lazy functional languages to have some version of "strict" or call-by-value Let construct embedded in it for greater flexibility, and arguably, expressiveness in programming style.

DEFINITION **4.4.1.7** In [Plo75], Plotkin introduced a *call-by-value* or *strict language* $\langle \Lambda^o, \Downarrow_{\mathsf{v}} \rangle$ where the binary reduction relation $\Downarrow_{\mathsf{v}}$ on Λ^o is defined as:

(abs$\Downarrow_{\mathsf{v}}$)
$$\frac{}{\lambda x.M \Downarrow_{\mathsf{v}} \lambda x.M}$$

(β_V)
$$\frac{M \Downarrow_{\mathsf{v}} \lambda x.P \quad N \Downarrow_{\mathsf{v}} Q \quad P[x := Q] \Downarrow_{\mathsf{v}} L}{MN \Downarrow_{\mathsf{v}} L}$$

The associated convergence predicate $\Downarrow_{\mathsf{v}}$ and divergence predicate $\Uparrow_{\mathsf{v}}$ are defined in the usual way.

The crucial difference between the call-by-value language and pure lazy language is that the former evaluates the "argument" first and then the function body — *eager* evaluation, whereas the latter does not evaluate the "argument" at all — *lazy* evaluation. Consequently, $(\lambda xy.x)\Omega\Uparrow_{\mathsf{v}}$ but $(\lambda xy.x)\Omega\Downarrow$, for example.

4.4.2 The Proof System $\lambda\beta\mathsf{C}$ and Lazy $\beta\mathsf{C}$-Reduction

In this subsection, we introduce the formal system $\lambda\beta\mathsf{C}$ which is $\lambda\beta$ augmented with a convergence testing constant C. $\lambda\beta\mathsf{C}$ relates to $\lambda\ell_{\mathsf{C}}$ in the same way as $\lambda\beta$ relates to $\lambda\ell$.

DEFINITION **4.4.2.1** (i) Let $\lambda\beta\mathsf{C}$ be the proof system defined on the language $\Lambda(\mathsf{C})$ obtained by augmenting the axioms and rules of $\lambda\beta$ with the following axiom scheme: $\mathsf{C}M = \mathbf{I}$ provided $M\Downarrow_{\mathsf{x}}$. As usual, we write $\lambda\beta\mathsf{C} \vdash$ for provability.

(ii) Define an associated proof system with formulae of the form $M \geqslant N$ as follows: $\lambda\beta\mathsf{C} \vdash M \geqslant N$ if $\lambda\beta\mathsf{C} \vdash M = N$ *without* using the *symmetry rule.*

DEFINITION **4.4.2.2** Define two binary relation schemes on $\Lambda(\mathsf{C})$ as follows:

$$\beta \stackrel{\text{def}}{=} \langle (\lambda x.P)Q, P[x := Q]\rangle,$$
$$\mathbf{C} \stackrel{\text{def}}{=} \langle \mathsf{C}(\lambda x.P), \mathbf{I}\rangle \cup \langle \mathsf{CC}, \mathbf{I}\rangle.$$

As before, we define:

$$\to_{\beta\mathsf{c}} \stackrel{\text{def}}{=} \text{compatible closure of } \beta \cup \mathbf{C},$$
$$\twoheadrightarrow_{\beta\mathsf{c}} \stackrel{\text{def}}{=} \text{reflexive, transitive closure of } \to_{\beta\mathsf{c}},$$
$$\stackrel{+}{\to}_{\beta\mathsf{c}} \stackrel{\text{def}}{=} \text{transitive closure of } \to_{\beta\mathsf{c}}.$$

$\to_{\beta\mathsf{c}}$ is known as the *one-step* $\beta\mathsf{C}$*-reduction.*

DEFINITION **4.4.2.3** (Lazy $\beta\mathsf{C}$-Reduction)
We define *one-step lazy* $\beta\mathsf{C}$*-reduction* (or simply, one-step lazy reduction) on $\Lambda(\mathsf{C})$ as follows:

$$a.\frac{}{\mathsf{CC} \to_{\mathsf{l}} \mathbf{I}} \qquad b.\frac{}{\mathsf{C}(\lambda x.P) \to_{\mathsf{l}} \mathbf{I}} \qquad c.\frac{}{(\lambda x.P)Q \to_{\mathsf{l}} P[x := Q]}$$

$$d.\frac{M \to_{\mathsf{l}} M'}{\mathsf{C}M \to_{\mathsf{l}} \mathsf{C}M'} \qquad e.\frac{M \to_{\mathsf{l}} M'}{MN \to_{\mathsf{l}} M'N}.$$

As usual, we define:

$$
\begin{array}{lll}
\xrightarrow{+}_{\mathrm{l}} & \stackrel{\text{def}}{=} & \text{transitive closure of } \to_{\mathrm{l}}, \\
\twoheadrightarrow_{\mathrm{l}} & \stackrel{\text{def}}{=} & \text{reflexive, transitive closure of } \to_{\mathrm{l}}, \\
M \twoheadrightarrow_{\mathrm{l}}^{p} N & \stackrel{\text{def}}{=} & M \equiv M_0 \to_{\mathrm{l}} M_1 \to_{\mathrm{l}} \cdots \to_{\mathrm{l}} M_p \equiv N, \\
M \not\to_{\mathrm{l}} & \stackrel{\text{def}}{=} & M \notin \mathsf{dom}(\to_{\mathrm{l}}), \\
\mathsf{nf}(\to_{\mathrm{l}}) & \stackrel{\text{def}}{=} & \{ M \in \Lambda(C) : M \not\to_{\mathrm{l}} \}, \\
M \downarrow_{\mathrm{l}} N & \stackrel{\text{def}}{=} & M \twoheadrightarrow_{\mathrm{l}} N \;\&\; N \not\to_{\mathrm{l}}, \\
M \uparrow_{\mathrm{l}} & \stackrel{\text{def}}{=} & \exists \{ M_i : i \in \omega \}. M_0 \equiv M \;\&\; \forall i \in \omega. M_i \to_{\mathrm{l}} M_{i+1}, \\
M \twoheadrightarrow_{\mathrm{l}\geqslant 1} N & \stackrel{\text{def}}{=} & M \equiv M_0 \to_{\beta \mathsf{c}}^{\Delta_1} M_1 \cdots \to_{\beta \mathsf{c}}^{\Delta_n} M_n \equiv N \& \\
 & & \exists 1 \leqslant i \leqslant n. \Delta_i \text{ is lazy}, \\
M \to_{\neq \mathrm{l}} N & \stackrel{\text{def}}{=} & M \to_{\beta \mathsf{c}} N \;\&\; \neg [M \to_{\mathrm{l}} N], \\
\twoheadrightarrow_{\neq \mathrm{l}} & \stackrel{\text{def}}{=} & \text{reflexive, transitive closure of } \to_{\neq \mathrm{l}}.
\end{array}
$$

A context $C[\,] \in \Lambda(\mathsf{C})$ is *lazy* if it has the syntactic shape

$$\underbrace{\mathsf{C}(\cdots(\mathsf{C}}_{n \geqslant 0}([\,]\vec{A})\vec{B})\cdots)\vec{D} \text{ where } \vec{A}, \vec{B}, \cdots, \vec{D} \subseteq \Lambda(\mathsf{C}).$$

REMARK **4.4.2.4** (i) The *one-step lazy reduction* may be presented equivalently as the union of the following relation schemes:

$$
\begin{array}{lll}
\to_{\mathrm{l}} \stackrel{\text{def}}{=} & \langle C[\mathsf{CC}], C[\mathbf{I}] \rangle & \text{case a.} \\
\cup & \langle C[\mathsf{C}(\lambda x.P)], C[\mathbf{I}] \rangle & \text{case b.} \\
\cup & \langle C[(\lambda x.P)Q], C[P[x := Q]] \rangle; & \text{case c.}
\end{array}
$$

where $C[\,]$ ranges over *lazy* contexts.

(ii) By definition, $P \to_{\mathrm{l}} P' \Rightarrow \forall$ lazy $C[\,] \in \Lambda(\mathsf{C}). C[P] \to_{\mathrm{l}} C[P']$.

(iii) It should be easy to see that for $M, N \in \Lambda(\mathsf{C})$,

$$\lambda\beta\mathsf{C} \vdash M \geqslant N \iff M \twoheadrightarrow_{\beta \mathsf{c}} N.$$

(iv) It is easy to see that

$$
\begin{array}{rcl}
\mathsf{nf}(\to_{\mathsf{c}}) & = & \{ \mathsf{C} \} \cup \{ \lambda x.P. P \in \Lambda(\mathsf{C}) \} \cup \\
 & & \{ C[x] : x \in \mathsf{Var} \;\&\; C[\,] \text{ is a lazy context} \}.
\end{array}
$$

4.4.3 Church-Rosser Property and Standardization Theorem

In this subsection, we will establish some basic properties of the proof system $\lambda\beta$C and the associated reduction relation. We generalize the notion of *strong unsolvability* to Λ(C) and prove an *operational characterization* result.

CONVENTION **4.4.3.1** We emphasize that α-convertible terms are implicitly identified.

THEOREM **4.4.3.2 (Church-Rosser)** *The proof system $\lambda\beta$C is Church-Rosser, i.e.*

$$\lambda\beta\mathsf{C} \vdash M \geqslant M_i \text{ for } i = 1,2 \Rightarrow \exists N.\lambda\beta\mathsf{C} \vdash M_i \geqslant N.$$

The proof of the Theorem follows the strategy of Tait and Martin-Löf by using a suitably defined *parallel reduction relation.* Following Plotkin [Plo75], we use the technique of parallel reduction to prove a *Standardization Theorem.* First, a definition.

DEFINITION **4.4.3.3 (Standard Reduction Sequence)**
Define *standard reduction sequence* on Λ(C) inductively as follows:

$$(1)\ \frac{}{\langle x\rangle} \quad \frac{}{\langle \mathsf{C}\rangle} \qquad (2)\ \frac{\langle N_2,\cdots,N_n\rangle \quad N_1 \to_1 N_2}{\langle N_1,N_2,\cdots,N_n\rangle}$$

$$(3)\ \frac{\langle N_1,\cdots,N_n\rangle}{\langle \lambda x.N_1,\cdots,\lambda x.N_n\rangle} \qquad (4)\ \frac{\langle M_1,\cdots,M_m\rangle \quad \langle N_1,\cdots,N_n\rangle}{\langle M_1N_1,\cdots,M_mN_1,M_mN_2,\cdots,M_mN_n\rangle}.$$

THEOREM **4.4.3.4 (Standardization)** *Let $M,N \in \Lambda$(C). Then,*

$$\lambda\beta\mathsf{C} \vdash M \geqslant N \iff \exists \vec{M}.M_1 \equiv M \;\&\; M_m \equiv N \;\&\; \langle M_1,\cdots,M_m\rangle.$$

Note that any standard reduction sequence is composed of a (possibly empty) sequence of one-step lazy βC-reductions followed by a (possibly empty) sequence of one-step βC-reductions in which the C- and β-redexes are systematically contracted in a strictly left to right order.

LEMMA **4.4.3.5**

$$\langle M_1,\cdots,M_m\rangle \Rightarrow \exists 1 \leqslant i \leqslant m.M_1 \twoheadrightarrow_1 M_i \twoheadrightarrow_{\neq 1} M_m.$$

PROOF Straightforward induction on m — the length of the standard reduction sequence. □

The parallel reduction is defined inductively together with a measure of the number of one-step βC-reductions concurrently executed in one parallel move as follows. This measure yields a handle on which to perform induction proofs. Let $\mathrm{occ}(x, M) \stackrel{\mathrm{def}}{=}$ the number of occurrences of x in M.

DEFINITION **4.4.3.6 (Parallel Reduction)** We read $M \geqslant_{\mathrm{p}} N \propto p$ as "M reduces to N in p number of one-step reductions preformed in parallel." p is referred to as the *size* of the proof of $M \geqslant_{\mathrm{p}} N$.

$$(1)\frac{M \geqslant_{\mathrm{p}} M' \propto p_M \quad N \geqslant_{\mathrm{p}} N' \propto p_N}{(\lambda x.M)N \geqslant_{\mathrm{p}} M'[x := N'] \propto (p_M + \mathrm{occ}(x, M')p_N + 1)}$$

$$(2)\frac{}{\mathsf{C}(\lambda x.M) \geqslant_{\mathrm{p}} \mathbf{I} \propto 1} \qquad \frac{}{\mathsf{CC} \geqslant_{\mathrm{p}} \mathbf{I} \propto 1} \qquad (3)\frac{}{M \geqslant_{\mathrm{p}} M \propto 0}$$

$$(4)\frac{M \geqslant_{\mathrm{p}} M' \propto p_M}{\lambda x.M \geqslant_{\mathrm{p}} \lambda x.M' \propto p_M} \qquad (5)\frac{M \geqslant_{\mathrm{p}} M' \propto p_M \quad N \geqslant_{\mathrm{p}} N' \propto p_N}{(MN) \geqslant_{\mathrm{p}} (M'N') \propto (p_M + p_N)}.$$

Firstly, we note that parallel reduction has "deductive power" at least as great as the "uni-directional" (or reduction) version of the proof system βC with formulae of the form $M \geqslant N$, as formalized in the following:

LEMMA **4.4.3.7** $\lambda\beta\mathsf{C} \vdash M \geqslant N \Rightarrow \exists \vec{N}.M \equiv N_1 \geqslant_{\mathrm{p}} \cdots \geqslant_{\mathrm{p}} N_n \equiv N$.

PROOF Straightforward induction. □

The following Substitution Lemma is needed in the proof of both the Church-Rosser and the Standardization Theorems.

LEMMA **4.4.3.8 (Substitution)**

$$M \geqslant_{\mathrm{p}} M' \propto p_M \;\&\; N \geqslant_{\mathrm{p}} N' \propto p_N \Rightarrow M[x := N] \geqslant_{\mathrm{p}} M'[x := N'] \propto p;$$

where $p \leqslant p_M + occ(x, M')p_N$.

PROOF The proof is by induction on the size of M and by cases according to the last rule applied in $M \geqslant_{\mathrm{p}} M'$ which follows that of Lemma 5 in [Plo75, page 137]. □

PROOF OF THEOREM 4.4.3.2 (Church-Rosser)
This is the method of Tait and Martin-Löf. Suppose the antecedent of the Theorem. First note that by Lemma 4.4.3.7, for $i = 1, 2$, $\exists N_1^i, \cdots, N_{n_i}^i.M \equiv N_1^i \geqslant_{\mathrm{p}} \cdots \geqslant_{\mathrm{p}} N_{n_i}^i = M_i$. By the above Substitution Lemma, the Church-Rosser property of the parallel reduction $\geqslant_{\mathrm{p}}$ is easily shown. The Theorem then follows from a simple induction. □

The next two Lemmas show that if the result N of a parallel reduction of an application M is the constant C, a variable or an abstraction, then the same result can be arrived at by performing lazy reduction on M; except in the case of the result being an abstraction, the "normal form" of the lazy reduction differs from the result in that all redexes "under" the abstraction are not contracted. The proofs are straightforward inductions on the size of proofs of $M \geqslant_{\mathrm{p}} N$ and are dependent on the preceding Substitution Lemma.

LEMMA **4.4.3.9** *If* $M \equiv UV \geqslant_{\mathrm{p}} N$ *where* $N \equiv \mathsf{C}$ *or* x, *then* $M \twoheadrightarrow_{\mathrm{l}} N$. □

LEMMA **4.4.3.10**

$$M \equiv UV \geqslant_{\mathrm{p}} \lambda x.P \propto p_M \Rightarrow \exists P'.M \twoheadrightarrow_{\mathrm{l}} \lambda x.P' \geqslant_{\mathrm{p}} \lambda x.P \propto p \text{ where } p < p_M.$$

□

The next two Lemmas lead up to the proof of the Standardization Theorem. Their proofs employ ***lexicographic induction*** on a pair (or more generally, an n-tuple) of integers $\langle m, n\rangle$. The ordering, $\leqslant$, used is as follows:

$$\langle m, n\rangle \leqslant \langle m', n'\rangle \stackrel{\text{def}}{=} m < m' \text{ or else } m = m' \,\&\, n \leqslant n'.$$

Observe that lexicographic induction is just a labour-saving device; any such induction can always be replaced by nested (ordinary) inductions.

The length, $|M|$, of a $\lambda\mathsf{C}$-term M is defined inductively as follows:

$$\begin{aligned} |x|, |\mathsf{C}| &\stackrel{\text{def}}{=} 1, \\ |\lambda x.M| &\stackrel{\text{def}}{=} |M| + 1, \\ |MN| &\stackrel{\text{def}}{=} |M| + |N|. \end{aligned}$$

LEMMA **4.4.3.11 (Commutativity)**

$$M \geqslant_{\mathrm{p}} M' \rightarrow_{\mathrm{l}} M'' \Rightarrow \exists K.M \twoheadrightarrow_{\mathrm{l}} K \geqslant_{\mathrm{p}} M''.$$

PROOF The proof, which follows that of Lemma 8 in [Plo75, page 140], is by lexicographic induction on $\langle p_M, |M|\rangle$ where p_M is the size of proof of $M \geqslant_{\mathrm{p}} M'$. □

LEMMA **4.4.3.12**

$$\langle M_1, \cdots, M_j\rangle \,\&\, M \geqslant_{\mathrm{p}} M_1 \Rightarrow \exists \langle N_1, \cdots, N_n\rangle.N_1 \equiv M \,\&\, N_n \equiv M_j.$$

PROOF The proof is by lexicographic induction on $\langle j, p_M, |M|\rangle$ where p_M is the size of proof of $M \geqslant_{\mathrm{p}} M_1$ and by cases on the last rule used in the proof. The proof follows that of Lemma 9 in [Plo75, page 141]. □

Now, we are in a position to prove the Standardization Theorem.

PROOF OF THEOREM 4.4.3.4 (Standardization)
The direction "$\Leftarrow$" is obvious. Suppose $\lambda\beta C \vdash M \geqslant N$. By Lemma 4.4.3.7, $\exists N_1, \cdots, N_l$ such that $M \equiv N_1 \geqslant_p \cdots \geqslant_p N_l = N$. We proceed by induction on l. The base case is obvious. Suppose true for $l-1$. Then $\exists K_1, \cdots, K_k.K_1 \equiv N_2 \,\&\, K_k = N \,\&\, \langle K_1, \cdots, K_k\rangle$. As $M \geqslant_p K_1$, the result follows at once from Lemma 4.4.3.12. □

Operational Chracterization of Strong Unsolvability

DEFINITION **4.4.3.13** Let $M \in \Lambda(C)$. An infinite βC-reduction starting from M

$$M \equiv M_0 \to_{\beta c}^{\Delta_1} M_1 \to_{\beta c}^{\Delta_2} \cdots \to_{\beta c}^{\Delta_i} M_i \to_{\beta c}^{\Delta_{i+1}} \cdots$$

is *quasi-lazy* if $\exists \langle n_i : i \in \omega\rangle . \forall i \in \omega . n_i < n_{i+1} \,\&\, M_{n_i - 1} \to_{\beta c}^{\Delta_{n_i}} M_{n_i}$ is a one-step lazy reduction.

Next, we prove the following result:

PROPOSITION **4.4.3.14** *Let $M \in \Lambda(C)$. If M has an infinite quasi-lazy reduction, then M has an infinite lazy reduction.*

LEMMA **4.4.3.15** (Substitutivity of βC) *The reduction $\to_{\beta c}$ is substitutive:*

$$\forall M, N \in \Lambda(C).M \to_{\beta c} N \Rightarrow \forall P \in \Lambda(C).M[x := P] \to_{\beta c} N[x := P].$$

PROOF The proof follows from the substitutivity of β-reduction which is an easy exercise. Substitutivity of the C-reduction is immediate. □

The next Lemma says that lazy one-step reductions in a βC-reduction can always be advanced.

LEMMA **4.4.3.16** (Advancement) *Let $M, M', N \in \Lambda(C)$. Then*

(i) $M \twoheadrightarrow_{\neq 1} N \to_1 N' \,\&\, M \to_1 M' \Rightarrow M' \twoheadrightarrow_{\beta c} N'$.

(ii) $M \twoheadrightarrow_{\neq 1} N \overset{+}{\to}_1 N' \Rightarrow \exists M'.M \overset{+}{\to}_1 M' \twoheadrightarrow_{\neq 1} N'$.

PROOF (i) Let $C[\,], C'[\,]$ range over *lazy* contexts. We consider the hardest of the three cases in which $M \to_1 M'$, *i.e.* case c. with reference to Remark 4.4.2.4(i):

$$M \equiv C[(\lambda x.P)Q] \to_1 C[P[x := Q]] \equiv M' \text{ and}$$

$$M \equiv C[(\lambda x.P)Q] \twoheadrightarrow_{\neq 1} C'[(\lambda x.P')Q'] \equiv N;$$

where $P \twoheadrightarrow_{\beta c} P', Q \twoheadrightarrow_{\beta c} Q'$ and $C[\,] \twoheadrightarrow_{\beta c} C'[\,]$ (regarding the "hole" $[\,]$ as a free variable). Hence, by the substitutivity of $\beta\mathsf{C}$,

$$C[P[x := Q]] \twoheadrightarrow_{\beta c} C[P'[x := Q]] \twoheadrightarrow_{\beta c} C'[P'[x := Q]] \twoheadrightarrow_{\beta c} C'[P'[x := Q']].$$

Then, note that $C'[(\lambda x.P')Q'] \rightarrow_1 C'[P'[x := Q']]$, and so, $N' \equiv C'[P'[x := Q']]$, then the result follows immediately. The proof of (ii) follows the that of Lemma 13.2.5(ii) in [Bar84, page 328] with $\twoheadrightarrow_\beta$ and the β-Standardization Theorem replaced by $\twoheadrightarrow_{\beta c}$ the $\beta\mathsf{C}$-Standardization Theorem. □

The proof of Proposition 4.4.3.14 then follows from the preceding Lemma (Advancement)(ii) by a "diagram chase" as depicted comprehensively in Figure 13.4 in [Bar84, page 329].

DEFINITION **4.4.3.17** $M \in \Lambda(\mathsf{C})$ is *strongly unsolvable*[5], denoted $M \in \mathbf{PO}_0$, if

$$\neg[\exists N.\lambda\beta\mathsf{C} \vdash M = \lambda x.N] \;\&\; \neg[\lambda\beta\mathsf{C} \vdash M = \mathsf{C}]\&$$

$$\neg[\exists \text{ lazy } C[\,] \in \Lambda(\mathsf{C}), \exists x \in \mathsf{Var}.\lambda\beta\mathsf{C} \vdash M = C[x]].$$

We prove an *operational characterization* result for strong unsolvability in $\beta\mathsf{C}$.

PROPOSITION **4.4.3.18 (Operational Characterization)** *Let $M \in \Lambda(\mathsf{C})$. Then, the following are equivalent:*

(i) M is strongly unsolvable, i.e. $M \in \mathbf{PO}_0$,

(ii) M has an infinite quasi-lazy reduction,

(iii) $M\uparrow_1$,

PROOF (ii) and (iii) are equivalent by Proposition 4.4.3.14. The *normal forms* of the lazy $\beta\mathsf{C}$-reduction, *i.e.* all those $M \in \Lambda(\mathsf{C})$ such that $M \not\rightarrow_1$ are of the following syntactic shapes: C, or $\lambda x.P$ or $C[x]$ for some lazy context. Hence, if $M \downarrow_1 \lambda x.P$, say, then $\lambda\beta\mathsf{C} \vdash M = \lambda x.P$; *i.e.* we have shown (i) $\Rightarrow$ (iii). The converse follows from the Standardization Theorem. □

LEMMA **4.4.3.19** *$\lambda\ell_{\mathsf{C}}$ is a $\lambda\beta\mathsf{C}$-theory, i.e. $\lambda\ell_{\mathsf{C}}$ is a consistent extension of $\lambda\beta\mathsf{C}$.*

PROOF Straightforward. □

[5]This notion is a natural and *consistent* extension of *strong unsolvability* as introduced in Definition 2.1.1.3 which applies only to λ-terms (without any constants).

4.4.4 Simulation of Call-by-Value Evaluation in $\lambda\ell_C$

In this subsection, we show that *convergence* (and dually, divergence) in the call-by-value language can be *simulated* in the pure lazy language augmented with a convergence testing constant via a *syntactic translation* of terms.

DEFINITION **4.4.4.1** We define a translation $\overline{(\ )} : \Lambda \to \Lambda(C)$ by structural induction as follows:

$$\begin{aligned} \overline{x} &\stackrel{\text{def}}{=} x, \\ \overline{\lambda x.M} &\stackrel{\text{def}}{=} \lambda x.\overline{M}, \\ \overline{MN} &\stackrel{\text{def}}{=} C\overline{N}((\overline{M})(\overline{N})). \end{aligned}$$

THEOREM **4.4.4.2** (Simulation) *Let $M \in \Lambda^o$. Then,*

$$M\Downarrow_v \iff \overline{M}\Downarrow_C .$$

REMARK **4.4.4.3** The translation $\overline{(\ )}$ does not in general preserve the *result* of convergent call-by-value reduction. Just consider $M \equiv (\lambda xy.x)(\mathbf{II})$. Observe that $M \Downarrow_v \lambda y.\mathbf{I}$; whereas $\overline{M} \Downarrow_C \lambda y.\overline{\mathbf{II}} =_\alpha \overline{\lambda y.\mathbf{I}}$. In the sequel, we show that there is, however, a *parallel* βC-reduction strategy, $\to_o$, which simulates call-by-value reduction via the same translation in a *step-wise* fashion (see Proposition 4.4.4.9); whence, the result of convergent call-by-value reduction is preserved.

Proof of the Simulation Theorem

The rest of this subsection will be devoted to the proof of the Simulation Theorem.

DEFINITION **4.4.4.4** We define *one-step call-by-value reduction* $\to_v \subseteq \Lambda^o \times \Lambda^o$ as follows:

$$(1)\ \frac{}{(\lambda x.P)(\lambda y.Q) \to_v P[x := \lambda y.Q]}$$

$$(2)\ \frac{M \to_v M'}{(\lambda x.P)M \to_v (\lambda x.P)M'} \qquad (3)\ \frac{M \to_v M'}{MN \to_v M'N}.$$

As usual, we define $M \twoheadrightarrow_v N$, $M \downarrow_v N$ and $M\uparrow_v$ accordingly.

LEMMA **4.4.4.5** *Let $M \in \Lambda^o$. Then, $M \Downarrow_v \lambda x.P \iff M \downarrow_v \lambda x.P$.*

PROOF Straightforward induction on the number of reduction steps in $\Downarrow_v$ in exactly the same way as the proof of Lemma 2.1.1.2. □

In the following, we define a "one-step" *parallel* βC-reduction strategy in $\Lambda(\mathsf{C})$ which simulates call-by-value reduction in a *step-wise* fashion.

DEFINITION **4.4.4.6** We define a notion of reduction, $\rightarrow_o \subseteq \Lambda(\mathsf{C}) \times \Lambda(\mathsf{C})$ as follows:

$$(1)\ \frac{}{\mathsf{C}(\lambda y.Q)((\lambda x.P)(\lambda y.Q)) \rightarrow_o P[x := \lambda y.Q]}$$

$$(2)\ \frac{P \rightarrow_o P'}{\mathsf{C}P((\lambda x.Q)P) \rightarrow_o \mathsf{C}P'((\lambda x.Q)P')} \qquad (3)\ \frac{P \rightarrow_o P'}{\mathsf{C}Q(PQ) \rightarrow_o \mathsf{C}Q(P'Q)}.$$

Similarly, we defined $\twoheadrightarrow_o$, $\overset{+}{\rightarrow}_o$ (transitive closure of $\rightarrow_o$), $M \downarrow_o N$ and $M\uparrow_o$ accordingly. We denote $M \twoheadrightarrow_o^p N \overset{\text{def}}{=} M \equiv M_0 \rightarrow_o M_1 \rightarrow_o \cdots \rightarrow_o M_p \equiv N$.

Note that rules (1), (2) and (3) in the definition of $\rightarrow_o$ are just the respective "translation" of the correspondingly numbered rules in the Definition of $\rightarrow_v$. This is made precise in Proposition 4.4.4.9.

REMARK **4.4.4.7** It is easy to see, by a straightforward inductive argument, that for $M, N \in \Lambda(\mathsf{C})$,

$$M \rightarrow_o N \Rightarrow M \twoheadrightarrow_{\beta\mathsf{c}} N.$$

In particular, if $M \rightarrow_o N$ according to rule (1), then $M \twoheadrightarrow_1^2 N$.

LEMMA **4.4.4.8** *Let* $P, Q \in \Lambda$. *Then,* $\overline{P}[x := \overline{Q}] = \overline{P[x := Q]}$.

PROOF By structural induction according to the syntactic categories of P. □

PROPOSITION **4.4.4.9 (1-Step Simulation)** *Let* $M, N \in \Lambda^o$. *Then,*

$$M \rightarrow_v N \iff \overline{M} \rightarrow_o \overline{N}.$$

PROOF Straightforward structural induction according to the rules of Definitions 4.4.4.4 and 4.4.4.6 and by an appeal to Lemma 4.4.4.8. □

An immediate corollary of the above Lemma is that the reduction $\rightarrow_o$ starting from any $\overline{M}$ where $M \in \Lambda^o$ is *deterministic* because $\rightarrow_v$ is.

COROLLARY **4.4.4.10** *Let* $M \in \Lambda^o$. *If* $\overline{M} \downarrow_o N$, *then* $N \equiv \lambda x_\iota P$. □

The following Proposition establishes the difficult direction *i.e.* "$\Leftarrow$" in the bi-implication of the Simulation Theorem.

PROPOSITION **4.4.4.11** *Let $M \equiv \overline{L}$ for some $L \in \Lambda^o$. If $\overline{M}\uparrow_o$, then $\overline{M}$ has an infinite quasi-lazy reduction. Hence, by the Operational Characterization Proposition 4.4.3.18, $\overline{M}\uparrow_1$.*

PROOF Let $M \in \overline{\Lambda^o}$. Suppose $M\uparrow_o$, *i.e.*

$$M \equiv M_0 \to_o^{\Delta_1} M_1 \to_o^{\Delta_2} \cdots \to_o^{\Delta_n} M_n \to_o^{\Delta_{n+1}} \cdots.$$

We assert that precisely one of the following three cases holds for M (with reference to Definition 4.4.4.6):

A. An unbounded number of reductions Δ_i use rule (1).

B. $\exists N \geqslant 1.\forall i \geqslant N.\Delta_i$ uses (2) as the last rule.

C. $\exists N \geqslant 1.\forall i \geqslant N.\Delta_i$ uses (3) as the last rule.

To see why this is so, suppose case A does not hold; and that N is the least n such that $\forall i \geqslant n.\Delta_i$ uses (2) or (3) as the last rule.

- If Δ_N uses (2) as the last rule, then M_N has syntactic shape $\mathsf{C}P((\lambda x.Q)P)$ where P is not an abstraction; for otherwise, Δ_{N+1} would use rule (1), contradicting the supposition. For Δ_{N+1}, the only possible candidate for the last rule applied is (2) with the premise $P \to_o P'$. P' clearly cannot be an abstraction for the same reason as before. Applying this argument inductively yields case B. Note that $P\uparrow_o$.
- Suppose Δ_N uses (3) as the last rule and M_N has syntactic shape $\mathsf{C}Q(PQ)$. If $P\uparrow_o$, then it should be clear that case C holds. If not, then by Corollary 4.4.4.10, $P \downarrow_o (\lambda x.P')$ (in which case Q is not an abstraction), and so case B holds.

For ease of reading, we introduce the following shorthand. Define two classes of contexts parametrized by their subscripts as follows:

$$D_{\lambda x.Q}[\,] \stackrel{\text{def}}{=} \mathsf{C}[\,]((\lambda x.Q)[\,]); \qquad E_Q[\,] \stackrel{\text{def}}{=} \mathsf{C}Q([\,]Q).$$

Now, if case A holds, then by Remark 4.4.4.7, M has an infinite quasi-lazy reduction. We consider cases B and C:
case B: $M \twoheadrightarrow_o D_{\lambda x.Q}[P_0] \to_o D_{\lambda x.Q}[P_1] \to_o \cdots$ where $P_0\uparrow_o$;
case C: $M \twoheadrightarrow_o E_Q[P_0] \to_o E_Q[P_1] \to_o \cdots$ where $P_0\uparrow_o$.

Let $F_i[\,]$ range over contexts $D_{\lambda x.Q}[\,]$ or $E_Q[\,]$, for $i \in \omega$ and $Q \in \Lambda(\mathsf{C})$. We claim that

- **either** I. Some finite N is the largest n such that $\forall i \leqslant n.\mathbf{L}_n$ holds, where

$$\mathbf{L}_n : M \twoheadrightarrow_o F_1[X_1] \twoheadrightarrow_o F_1[F_2[X_2]] \twoheadrightarrow_o \cdots \twoheadrightarrow_o F_1[\cdots[F_n[X_n]]\cdots] \,\&\, X_n\uparrow_o,$$

  in which case $X_N\uparrow_o$ according to A;

- **or** II. $\mathbf{L}_n$ is true for all $n \in \omega$.
  Equivalently, $\exists\{X_i : i \in \omega, X_0 \equiv M, X_i\uparrow_o\}$ such that

$$\begin{aligned} M \twoheadrightarrow_o^{p_1} \; & F_1[X_1] \twoheadrightarrow_o^{p_2} \cdots \twoheadrightarrow_o^{p_i} F_1[\cdots[F_i[X_i]]\cdots] \\ \twoheadrightarrow_o^{p_{i+1}} \; & F_1[\cdots[F_{i+1}[X_{i+1}]]\cdots]\cdots \end{aligned}$$

  We choose inductively a canonical set of X_i's as follows: for each $i \in \omega$, $X_{i+1} \stackrel{\text{def}}{=}$ that X which corresponds to the *least* p such that $X_i \twoheadrightarrow_o^p F[X]\&X\uparrow_o$ where F is some $D[\,]$ or $E[\,]$ (note that $F_{i+1} \stackrel{\text{def}}{=} F$). Observe that each $X_i\uparrow_o$ according to B or C and that each $p_i \geqslant 0$.

It should be clear that for case I, $M\uparrow_1$ has an infinite quasi-lazy reduction by induction on N.

For case II, we prescribe an infinite βC-reduction strategy starting from M which is *quasi-lazy*. First, for each X_i, we specify a reduction sequence composed of "one-step" *parallel* βC-reductions $\rightarrow_a$ (as usual, we define $\twoheadrightarrow_a$ to be the reflexive, transitive closure of $\rightarrow_a$) starting from X_i which

- *either* terminates, in which case,

$$X_i \equiv Z_0 \rightarrow_a^{\Delta_1} Z_1 \cdots \rightarrow_a^{\Delta_{q_i}} Z_{q_i} \equiv C_{i+1}[X_{i+1}]$$

  for some *lazy* context $C_{i+1}[\,]$;

- *or* X_i diverges under $\rightarrow_a$ and that the infinite βC-reduction (*i.e.* the associated *standard reduction*) is *quasi-lazy*.

$\rightarrow_a$ is defined by case analysis on the syntactic shape of X_i as follows. Given:

$$X_i \equiv Y_0 \rightarrow_o^{\Delta_1} Y_1 \cdots \rightarrow_o^{\Delta_{p_i}} Y_{p_i} \equiv F_{i+1}[X_{i+1}];\ \text{where } p_i \geqslant 0.$$

1. $X_i \equiv \mathsf{C}(\lambda x.Q)((\lambda y.P)(\lambda x.Q))$. Then, $p_i > 0$ and *at least* one $\rightarrow_o^{\Delta_i}$-reduction (*i.e.* $i = 1$) uses rule (1); hence, by Remark 4.4.4.7 the associated *standardized* βC-reduction from X_i to $F_{i+1}[X_{i+1}]$ contains at least two one-step lazy reductions.
   a. If $F_{i+1}[\,] \equiv D[\,] \equiv \mathsf{C}[\,]((\lambda y.S)[\,])$, then set $Z_i \stackrel{\text{def}}{=} Y_i, q_i \stackrel{\text{def}}{=} p_i, \rightarrow_a^{\Delta_i} \stackrel{\text{def}}{=} \rightarrow_o^{\Delta_i}$, $C_{i+1} \stackrel{\text{def}}{=} F_{i+1}[\,]$ which is a lazy context.

b. If $F_{i+1}[\,] \equiv E[\,] \equiv \mathsf{C}R([\,]R)$, then for all $1 \leqslant i \leqslant p_i$ set Z_i and $\rightarrow_{\mathrm{a}}^{\Delta_i}$ as in case 1a; for $i > p_i$, take $\rightarrow_{\mathrm{a}}^{\Delta_i}$ to be the lazy reduction starting from $\mathsf{C}R(X_{i+1}R)$, *i.e.*

$$\rightarrow_{\mathrm{a}}^{\Delta_{p_i+j}} \stackrel{\mathrm{def}}{=} \rightarrow_{\mathrm{l}}^{\Delta_j} : \mathsf{C}R(X_{i+1}R) \equiv W_0 \rightarrow_{\mathrm{l}}^{\Delta_1} W_1 \rightarrow_{\mathrm{l}}^{\Delta_2} \cdots \rightarrow_{\mathrm{l}}^{\Delta_i} W_i \cdots$$

Now, the lazy reduction of $\mathsf{C}R$ may or may not terminate. If it terminates, *i.e.* $\mathsf{C}R{\downarrow_{\mathrm{l}}}$ after, say, l 1-step lazy reductions, then

$$\begin{aligned} X_i \twoheadrightarrow_{\mathrm{a}}^{p_i} F_{i+1}[X_{i+1}] \equiv\ & \mathsf{C}R(X_{i+1}R) \twoheadrightarrow_{\mathrm{a}}^{l} (\stackrel{\mathrm{def}}{=} \twoheadrightarrow_{\mathrm{l}}^{l})\ \mathbf{I}(X_{i+1}R) \\ & \rightarrow_{\mathrm{a}}^{\Delta_{p_i+l+1}} C_{i+1}[X_{i+1}]; \end{aligned}$$

where $C_{i+1}[\,] \stackrel{\mathrm{def}}{=} ([\,]R)$; or else $\forall j \in \omega.\ \rightarrow_{\mathrm{a}}^{\Delta_{p_i+j}} \stackrel{\mathrm{def}}{=} \rightarrow_{\mathrm{l}}^{\Delta_j}$. Note that the associated standardized $\beta\mathsf{C}$-reduction of the latter infinite $\rightarrow_{\mathrm{a}}$-reduction is quasi-lazy.

2. $X_i \equiv \mathsf{C}P((\lambda x.Q)P)$ and P is not an abstraction.

   a. If $p_i = 0$, then $X_{i+1} = P$; $q_i \stackrel{\mathrm{def}}{=} 0, C_{i+1}[\,] \stackrel{\mathrm{def}}{=} \mathsf{C}[\,]((\lambda x.Q)P)$.

   b. If $p_i > 0$, then for some $j < p_i$ we must have

   $$X_i \twoheadrightarrow_{\mathrm{o}}^{j} \mathsf{C}(\lambda y.P')((\lambda x.Q)(\lambda y.P'))$$

   — the syntactic shape corresponding to case 1 ; for if not, then $P{\uparrow_{\mathrm{o}}}$ and so contradicting $p_i > 0$. Define the $\rightarrow_{\mathrm{a}}$-reduction from X_i to consist first of the finite sequence of the preceding $\rightarrow_{\mathrm{o}}$-reduction, then *followed* by the (possibly infinite) sequence of $\rightarrow_{\mathrm{a}}$-reduction starting from $\mathsf{C}(\lambda y.P')((\lambda x.Q)(\lambda y.P'))$ as specified in case 1.

3. $X_i \equiv \mathsf{C}Q(PQ)$ where P is not an abstraction.

   a. If $p_i = 0$, then $X_{i+1} = P$. Define the $\rightarrow_{\mathrm{a}}$-reduction starting from X_i as identical to the lazy reduction of $\mathsf{C}Q$ (as in case 1b) which may or may not terminate. If $\mathsf{C}Q{\downarrow_{\mathrm{l}}}$ does, then

   $$X_i \twoheadrightarrow_{\mathrm{a}} (\stackrel{\mathrm{def}}{=} \twoheadrightarrow_{\mathrm{l}})\ \mathbf{I}(PQ) \rightarrow_{\mathrm{a}} (PQ) \equiv C_{i+1}[X_{i+1}];$$

   where $C_{i+1} \stackrel{\mathrm{def}}{=} ([\,]Q)$ or else, X_i diverges under $\rightarrow_{\mathrm{a}}$. As in case 1b, the associated infinite standardized $\beta\mathsf{C}$-reduction is quasi-lazy.

   b. If $p_i > 0$, then we must have $X_i \twoheadrightarrow_{\mathrm{o}} \mathsf{C}Q((\lambda x.P')Q)$ — the syntactic shape corresponding to case 2 or 1 — for some P'. Define the $\rightarrow_{\mathrm{a}}$-reduction from X_i to be composed of the sequence of the preceding $\rightarrow_{\mathrm{o}}$-reduction followed by the $\rightarrow_{\mathrm{a}}$-reduction starting from $\mathsf{C}Q((\lambda x.P')Q)$ as prescribed in case 2 or 1 respectively.

Now, we are in a position to define an infinite reduction composed of $\rightarrow_a$-reductions from $X_0 \equiv M$ as follows. First, observe that

$$P \rightarrow_l P' \Rightarrow \forall \text{ lazy } C[\,] \in \Lambda(\mathsf{C}).C[P] \rightarrow_l C[Q].$$

We confuse $\rightarrow_a$ with its *lazy* compatible closure in the following way: If $X_i \twoheadrightarrow_a C_{i+1}[X_{i+1}]$ then for any *lazy context* $C[\,]$, we write $C[X_i] \twoheadrightarrow_a C[C_{i+1}[X_{i+1}]]$.

Let N be the least i such that $X_i\uparrow_a$; of course N may be ω, in which case, consider the following infinite reduction:

$$M \equiv X_0 \twoheadrightarrow_a^{p_1} C_1[X_1] \cdots \twoheadrightarrow_a^{p_i} C_1[\cdots[C_i[X_i]]\cdots]\cdots \twoheadrightarrow_a^{p_N} C_1[\cdots[C_N[X_N]]\cdots]\cdots$$

We claim that the above infinite βC-reduction (just "decompose" the $\rightarrow_o$-reduction into its associated standardized βC-reduction) is quasi-lazy.

To see this, we check through the above cases. Note that the $\rightarrow_a$-reduction segment from X_i to $C_{i+1}[X_{i+1}]$ corresponding to cases 1a, 1b, 2b and 3a all involve at least one lazy-reduction step. Hence, we only need to check the remaining case of an infinite reduction from M as defined above which use cases 1a,1b, 2b and 3a only *finitely* many times, say all before the $\rightarrow_a$-reduction of $X_{N'}$. In other words, all $\rightarrow_a$-reductions after $X_{N'}$ correspond to either cases 2a or 3b. This implies that the syntactically *finite* λC-term $X_{N'}$ has the syntactic shape of $\underbrace{\mathsf{C}(\cdots(\mathsf{C}\cdots)\cdots)}_{n}(PQ)$ such that n is *unbounded* which is absurd.

We therefore conclude the proof. □

PROOF OF THEOREM 4.4.4.2
Let $M \in \Lambda^o$. Suppose $M \Downarrow_v \lambda x.P$, *i.e.* $M \downarrow_v \lambda x.P$ by Lemma 4.4.4.5. By an appeal to the the 1-Step Simulation Proposition, we have $\overline{M} \twoheadrightarrow_o \lambda x.\overline{P}$. By the Standardization Theorem, we have $\overline{M}\Downarrow_c$. Now, suppose $M\Uparrow_v$. Then, $\overline{M}\uparrow_o$; and so, by Proposition 4.4.4.11, $\overline{M}\uparrow_l$, that is to say, $\overline{M}\Uparrow_c$ by Operational Characterization Proposition 4.4.3.18. □

4.5 Non Full Abstraction Results

We present two non full abstraction results in this section: namely, $\lambda\ell$ and $\lambda\ell_c$ are *not* fully abstract with respect to D. Any *lazy reflexive object* in the category of cpo's expands to a lazy λ-model which we call a *topological lazy λ-model.* A corollary of the first non full abstraction result is that *adequate* lazy topological λ-models are *incomplete* with respect to fully lazy λ-theories. The second non full abstraction result depends on the non definability of parallel convergence in $\lambda\ell_c$, which we prove by a syntactic case analysis.

4.5.1 $\lambda\ell$ is not Fully Abstract w.r.t. D

We state the first negative full-abstraction result.

THEOREM **4.5.1.1 (Non Full Abstraction 1)** $\exists M, N \in \Lambda. M \sim^B N \;\&\; D \not\models M = N$.

PROOF Let $M \equiv x(\lambda y.x\top\bot y)\top, N \equiv x(x\top\bot)\top$ where $\bot$ is any strongly unsolvable term, or equivalently a $\mathbf{PO}_0$-term and $\top$ is a $\mathbf{PO}_\infty$-term, for instance, $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ and $\mathbf{YK}$ respectively. We show:

I. $M \sim^B N$,

II. $D \not\models M = N$.

I. We claim that

$$\lambda\beta \vdash A{\downarrow_1} \Rightarrow A = \lambda y.Ay \text{ for } y \notin \mathsf{FV}(A).$$

(Recall form Chapter 2 that $A =_\beta \lambda x.P \iff A{\downarrow_1}$).

$$\begin{aligned} \lambda y.Ay &=_\beta \lambda y.(\lambda x.P)y && \beta\text{-conversion} \\ &=_\beta \lambda y.P[x := y] && \alpha\text{-conversion} \\ &=_\alpha \lambda x.P \\ &=_\beta A. \end{aligned}$$

Consequently, to show I, it suffices to show:

$$\forall\sigma : \mathsf{Var} \to \Lambda^o.\sigma(x) = L.L\top\bot\Uparrow \Rightarrow M_\sigma \sim^B N_\sigma,$$

since $\sim^B$ is a congruence relation. We consider the various cases for L such that $L\top\bot\Uparrow$. Now, it should be clear that $L \notin \mathcal{F}_3$. We remark the following:

- If a closed λ-term $P \in \mathbf{O}_2$[6], then, modulo β-convertibility, P must have one of the following syntactic shapes:
  (1) $\lambda x_1 x_2.\bot$,
  (2) $\lambda x_1 x_2.x_1\vec{P}$,
  (3) $\lambda x_1 x_2.x_2\vec{P}$.
- If a closed λ-term $P \in \mathbf{O}_1$, then, modulo β-convertibility, P must have the following:
  (1) $\lambda x.\bot$,

[6]Recall that for $n \in \omega$, $M \in \mathbf{O}_n$ if n is the *largest* i such that $\exists N.\lambda\beta \vdash M = \lambda x_1 \cdots x_i.N$.

(2) $\lambda x.x\vec{P}$.

By elimination, we arrive at the following cases for L such that $L\top\bot\Uparrow$.

(a) $L = \bot$,

(b) $L = \lambda x.\bot$,

(c) $L = \lambda x_1 x_2.\bot$,

(d) $L = \lambda x_1 x_2.x_2\vec{P}$.

We tabulate the simple calculation of each case as follows:

	M_σ	N_σ
(a),(b),(c)	$\bot$	$\bot$
(d)	$\top$	$\top$

Hence, $M \sim^B N$.

II. For any environment such that $\rho(x) = c$ where c is the convergence testing constant in D, we have

$$[\![M]\!]_\rho = c(\lambda y.c\top\bot y)\top = \top,$$

$$[\![N]\!]_\rho = c(c\top\bot)\top = c\bot\top = \bot.$$

Hence, $D \not\models M = N$. □

A fortriori, we have the following immediate corollary:

COROLLARY **4.5.1.2** *The inequational theory $\lambda\ell$ is not fully abstract with respect to D, i.e. it is* **not** *true that:*

$$\forall M, N \in \Lambda^o.M \precsim^B N \Rightarrow D \models M \sqsubseteq N.$$

□

Topological Incompleteness

It is well-known that any reflexive object in the Cartesian closed category of cpo's and Scott continuous functions is a λ-model which we refer to as a *topological λ-model*, following Honsell and Ronchi della Rocca [HdR88]. Let E be an object of the same category. We say that E is *lazy reflexive* if $[E \to E]_\bot \lhd E$; *i.e.* the *lifted* function space of E is a retract into E. It is straightforward to see, by essentially the same arguments as that for D, that a lazy reflexive object is a lazy λ-model. We call such a λ-model a *topological lazy λ-model.*

DEFINITION **4.5.1.3** A lazy λ-model $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!] \rangle$ is *fully lazy* if

$$\forall m, n \in \omega + 1. \forall M \in \mathbf{PO}_m. \forall N \in \mathbf{PO}_n. \mathcal{A} \models M = N \iff m = n.$$

OPEN QUESTION **4.5.1.4** It is obvious that a fully lazy topological λ-model is adequate. Is the converse true? That is to say, if all strongly unsolvable terms have the same denotation in a lazy topological λ-model, does it follow that all $\mathbf{PO}_\infty$-terms have the same denotation? (We can infer from the premise that for any fixed $n \in \omega$, all $\mathbf{PO}_n$-terms have the same denotation.)

It is natural to ask if all fully lazy λ-theories are induced by fully lazy topological λ-models, *i.e.* if fully lazy topological λ-models are *complete* with respect to fully lazy λ-theories.

Now, let E be a fully lazy topological λ-model. Then M and N in the proof of the Theorem have different denotations in E because convergence testing is definable in E, as is the case in D. We therefore have a simple proof of the following incompleteness result:

COROLLARY **4.5.1.5 (Topological Incompleteness)** *Fully lazy topological λ-models are incomplete with respect to fully lazy λ-theories.* □

Compare the Corollary with the topological incompleteness result in [HdR88].

Another Corollary of the Non Full Abstraction Theorem is the following:

COROLLARY **4.5.1.6 (Convergence testing is not λ-definable)** *Convergence testing is not definable in $\langle \Lambda^o, \Downarrow \rangle$.*

PROOF If convergence testing were definable in $\langle \Lambda^o, \Downarrow \rangle$ by C say, then with reference to the same M and N in the proof of Theorem 4.5.1.1, for σ such that $\sigma(x) = C$ we have $M_\sigma \Downarrow$ and $N_\sigma \Uparrow$; contradicting I in the proof of the Theorem. □

4.5.2 $\lambda\ell_C$ is not Fully Abstract w.r.t. D

Given that convergence testing is definable in D and that in the construction of our previous counter-example for the non full-abstraction of $\lambda\ell$ with respect to D, convergence testing features so pivotally; it is at least plausible that $\lambda\ell_C$ might be fully abstract with respect to D. This turns out *not* to be the case.

This result was first obtained by Samson Abramsky in [Abr87] as a corollary of his full abstraction result obtained by bringing domain logic to bear on the lazy λ-calculus. It was later independently obtained by the author by a direct, syntactic method. The following definitions are introduced to facilitate the case analysis in the proof of the non-full abstraction result.

DEFINITION **4.5.2.1** We define two binary reduction relations on $\Lambda(\mathsf{C})$: $\Downarrow_{\mathsf{x}}, \| \subseteq \Lambda(\mathsf{C}) \times \Lambda(\mathsf{C})$ by the following inference rules. Let P, Q, M, N range over (possibly open) $\lambda\mathsf{C}$-terms.

(abs-$\Downarrow_{\mathsf{x}}$) $$\frac{}{\lambda x.M \Downarrow_{\mathsf{x}} \lambda x.M}$$

(C-$\Downarrow_{\mathsf{x}}$) $$\frac{}{\mathsf{C} \Downarrow_{\mathsf{x}} \mathsf{C}}$$

(β_N-$\Downarrow_{\mathsf{x}}$) $$\frac{M \Downarrow_{\mathsf{x}} \lambda x.P \quad P[x := Q] \Downarrow_{\mathsf{x}} N}{MQ \Downarrow_{\mathsf{x}} N}$$

(Cλ-$\Downarrow_{\mathsf{x}}$) $$\frac{M \Downarrow_{\mathsf{x}}}{\mathsf{C}M \Downarrow_{\mathsf{x}} I}$$

(var-$\|$) $$\frac{}{x \| x}$$

(C-$\|$,$\vec{Q}$-$\|$) $$\frac{M \| N}{\mathsf{C}M \| \mathsf{C}N \quad MQ \| NQ}$$

(β_N-$\|$) $$\frac{M \Downarrow_{\mathsf{x}} \lambda x.P \quad P[x := Q] \| N}{MQ \| N}$$

As usual, we define convergence predicates $\Downarrow_{\mathsf{x}}$ and $\|$.

$$M\Downarrow_{\mathsf{x}} \stackrel{\text{def}}{=} \exists N.M \Downarrow_{\mathsf{x}} N.$$

Similarly, for $M\|$. Note that the divergence predicate is defined as

$$M\Uparrow_{\mathsf{x}} \stackrel{\text{def}}{=} \neg[M\Downarrow_{\mathsf{x}}] \,\&\, \neg[M\|].$$

REMARK **4.5.2.2** (i) The set of four inference rules post-fixed by $\Downarrow_{\mathsf{x}}$ are exactly the same as those for $\Downarrow_{\mathsf{c}}$ (except, of course, the meta-variables in the former range over possibly open $\lambda\mathsf{C}$-terms whereas those in the latter over closed terms). Hence, (regarding $\Downarrow_{\mathsf{c}}$ as a binary relation on $\Lambda(\mathsf{C})$), $\Downarrow_{\mathsf{x}}$ is a conservative extension of $\Downarrow_{\mathsf{c}}$, *i.e.* for $M \in \Lambda(\mathsf{C})^o$, we have $M\Downarrow_{\mathsf{c}} \iff M\Downarrow_{\mathsf{x}}$.

(ii) We read $M \| N$ as "M converges *partially* to N". "Partiality" arises expectedly from the *uninstantiated* free variables which might occur in head variable position in the $\lambda\mathsf{C}$-terms being considered for reduction.

The following Lemma asserts that $\Downarrow_x$ and $\|$ together specify a *deterministic* reduction strategy for λC-terms.

LEMMA **4.5.2.3** *Let $M \in \Lambda(\mathsf{C})$ and $\mathsf{FV}(M) = \{x_1, \cdots, x_n\}$ where $n \geqslant 0$. Then, precisely one of the following holds:*

(1) $M \Uparrow_x$,

(2) $M \Downarrow_x \lambda x.P$,

(3) $M \Downarrow_x \mathsf{C}$,

(4) $M \| C[x]$ *where* $C[\,]$ *is a lazy context.*

PROOF By a straightforward argument from the definition of $M \Uparrow_x$ and the observation that (2) and (3) describe the two syntactic classes of "normal forms" of $\Downarrow_x$ and that (4) captures the "normal forms" of $\|$. □

LEMMA **4.5.2.4** *Let $M \in \Lambda(\mathsf{C})$ and let $C[\,]$ range over lazy contexts. Then,*

(i) $M \downarrow_l \mathsf{C} \iff M \Downarrow_x \mathsf{C}$.

(ii) $M \downarrow_l \lambda x.P \iff M \Downarrow_x \lambda x.P$.

(iii) $M \downarrow_l C[x] \iff M \| C[x]$.

(iv) $M \uparrow_l \iff M \Uparrow_x$.

PROOF Straightforward induction on the number of one-step $\beta\mathsf{C}$-reductions in $M \Downarrow_x N$ and $M \| C[x]$. □

EXAMPLE **4.5.2.5** We present some examples of $\to_l$-divergent λC-terms:

(i) Let $\Delta_1 \equiv \lambda x.\mathsf{C}xxx$. Then, $\Delta_1\Delta_1 \to_l \mathsf{C}\Delta_1\Delta_1\Delta_1 \to_l \Delta_1\Delta_1 \to_l \cdots$.

(ii) Let $\Delta_2 \equiv \lambda x.\mathsf{C}(xx)$. Then, $\forall n \in \omega.\Delta_2\Delta_2 \twoheadrightarrow_l \underbrace{\mathsf{C}(\mathsf{C}(\cdots(\mathsf{C}}_{n}(\Delta_2\Delta_2))\cdots))$.

(iii) Let $\Delta_3 \equiv \lambda x.xx\mathsf{C}$. Then, $\Delta_3\Delta_3 \to_l \Delta_3\Delta_3\mathsf{C} \to_l \cdots \to_l \Delta_3\Delta_3\underbrace{\mathsf{C}\cdots\mathsf{C}}_{n} \to_l \cdots$.

The next Lemma makes precise the idea that $\Downarrow_x$ and $\Uparrow_x$ are natural extensions of $\Downarrow_c$ and $\Uparrow_c$ respectively to possibly open $\lambda\mathsf{C}$ terms in a way that respects closed substitutions.

LEMMA **4.5.2.6** *Let $M \in \Lambda(\mathsf{C})$ such that $\mathsf{FV}(M) = \{\vec{x}\}$. Then,*

(i) $M \Downarrow_x N \Rightarrow \forall \sigma : \mathsf{Var} \to \Lambda(\mathsf{C})^o.M_\sigma \Downarrow_c N_\sigma$.

(ii) $M \Uparrow_x \Rightarrow \forall \sigma : \mathsf{Var} \to \Lambda(\mathsf{C})^o.M_\sigma \Uparrow_c$.

(iii) $M\Uparrow_{\mathsf{x}} \Rightarrow \lambda\vec{x}.M \sim^{B} \lambda\vec{x}.\Theta$ *where* $\Theta \in \Lambda(\mathsf{C})^{o} \,\&\, \Theta\Uparrow_{\mathsf{C}}$. *See the previous Example for some such candidates.*

PROOF

(i) By Lemma 4.5.2.4 and the substitutivity of $\twoheadrightarrow_{\mathsf{I}}$.

(ii) Suppose $M\Uparrow_{\mathsf{x}}$. Then, by Lemma 4.5.2.4, $M\uparrow_{\mathsf{I}}$. By the substitutivity of $\twoheadrightarrow_{\mathsf{I}}$, we have $M_{\sigma}\uparrow_{\mathsf{I}}$ and so, by Lemma 4.5.2.4 again, $M_{\sigma}\Uparrow_{\mathsf{C}}$.

(iii) In view of (ii), it is easy to see that

$$\forall \vec{P} \subseteq \Lambda(\mathsf{C})^{o}.(\lambda\vec{x}.M)\vec{P}\Downarrow_{\mathsf{C}} \Longleftrightarrow (\lambda\vec{x}.\Omega)\vec{P}\Downarrow_{\mathsf{C}}.$$

□

DEFINITION **4.5.2.7** Let $\mathcal{K} = \langle K, \cdot, \Downarrow_{\mathcal{K}}\rangle$ be an aswd. We say that *parallel convergence is definable* in $\mathcal{K}$ if there exists $p \in K$ such that for $x, y \in K$, $\mathcal{K}$ satisfies the following:

- $p\Downarrow_{\mathcal{K}}$,
- $x\Downarrow_{\mathcal{K}} \Rightarrow pxy\Downarrow_{\mathcal{K}} \,\&\, pyx\Downarrow_{\mathcal{K}}$,
- $x\Uparrow_{\mathcal{K}} \,\&\, y\Uparrow_{\mathcal{K}} \Rightarrow pxy\Uparrow_{\mathcal{K}}$.

PROPOSITION **4.5.2.8** (Non λC-definability of P) *Let* $P \in \Lambda(\mathsf{C})^{o}$. *Then,*

$$* \qquad P\perp\perp\Uparrow_{\mathsf{C}} \Rightarrow P\perp_{1}\perp\Uparrow_{\mathsf{C}} \text{ or } P\perp\perp_{1}\Uparrow_{\mathsf{C}}$$

where $\perp$ *is any divergent* λC*-term and* $\perp_{1}$ *is any least convergent term, say* $\lambda x.\perp$. *Hence,* parallel convergence *is not definable in* $\langle\Lambda(\mathsf{C})^{o}, \Downarrow_{\mathsf{C}}\rangle$.

PROOF Wlog, we assume $P\Downarrow_{\mathsf{C}}$, $P\perp\Downarrow_{\mathsf{C}}$ and $P \notin \mathcal{F}_{3}$ which leaves us with the case of $P \Downarrow_{\mathsf{C}} \lambda x.M$ and that $\mathsf{FV}(M) \subseteq \{x\}$;

By applying Lemma 4.5.2.3 to M, we assert that P has one of the following syntactic shapes, modulo β-convertibility:

(1) $\lambda x.\perp$,

(2) $\lambda x.\mathsf{C}$,

(3) $\lambda x.C[x]$,

(4) $\lambda x_{1}x_{2}.\perp$,

(5) $\lambda x_{1}x_{2}.C[x_{i}]$ where $i = 1, 2$,

(6) $\lambda x_{1}x_{2}.\mathsf{C}$;

where $C[\,]$ ranges over lazy contexts and $\perp$ over $\rightarrow_{\mathsf{I}}$-divergent λC-terms. Observe that (6) satisfies (*) vacuously because the antecedent is not satisfied. It is straightforward to check that (1), (2), (3) and (4) satisfy (*). For (5), if $i = 1$, then $P\perp\perp_{1}\Uparrow_{\mathsf{C}}$, if $i = 2$ then $P\perp_{1}\perp\Uparrow_{\mathsf{C}}$. □

Now, we are in a position to prove the second non full abstraction result. (See [Abr87] for a different proof.)

THEOREM **4.5.2.9 (Non Full Abstraction 2)** $\exists M, N \in \Lambda(\mathsf{C}).M \sim^{\mathsf{c}} N \,\&\, D \not\models M = N$.

PROOF Let $\bot$ and $\bot_1$ be as before. Consider the following terms:

$$M \equiv \lambda x.\mathsf{C}(\mathsf{C}(x\bot_1\bot)A), \qquad N \equiv \lambda x.\mathsf{C}(\mathsf{C}(x\bot_1\bot)B)$$

where $A \equiv (x\bot\bot), B \equiv (x\bot\bot_1)$. We show:

I. $M \precsim^{\mathsf{c}} N \,\&\, N \precsim^{\mathsf{c}} M$.

II. $D \not\models M = N$.

I: Let $C[\,] \equiv \lambda x.\mathsf{C}(\mathsf{C}(x\bot_1\bot)(x\bot[\,]))$. Then, notice that $M \equiv C[\bot], N \equiv C[\bot_1]$. Since $\bot \precsim^{\mathsf{c}} \bot_1$, by the precongruence of $\precsim^{\mathsf{c}}$, we deduce that $M \precsim^{\mathsf{c}} N$. To show $N \precsim^{\mathsf{c}} M$, we only need to show

$$(*) \qquad \forall P \in \Lambda(\mathsf{C})^o.NP\Downarrow_{\mathsf{c}} \Rightarrow MP\Downarrow_{\mathsf{c}}.$$

This is because: first, we have $N\Downarrow_{\mathsf{c}}$ *and* $M\Downarrow_{\mathsf{c}}$. Next, assume $(*)$ is valid. Then, observe that,

$$\begin{aligned} NP\vec{Q}\Downarrow_{\mathsf{c}} &\Rightarrow NP\Downarrow_{\mathsf{c}} \mathbf{I} \,\&\, \vec{Q}\Downarrow_{\mathsf{c}} \\ &\Rightarrow MP\Downarrow_{\mathsf{c}} \mathbf{I} \,\&\, \vec{Q}\Downarrow_{\mathsf{c}} \\ &\Rightarrow MP\vec{Q}\Downarrow_{\mathsf{c}}. \end{aligned}$$

To verify $(*)$, we claim that it suffices to consider those P such that (let σ be the substitution mapping x to P) $B_\sigma\Downarrow_{\mathsf{c}} \,\&\, A_\sigma\Uparrow_{\mathsf{c}}$. This is because: if $B_\sigma\Uparrow_{\mathsf{c}}$, then $NP\Uparrow_{\mathsf{c}}$; if $A_\sigma\Downarrow_{\mathsf{c}}$, then (note that $A_\sigma \equiv P\bot\bot$, by $\bot \precsim^{\mathsf{c}} \bot_1$ and the precongruence of $\precsim^{\mathsf{c}}$) $P\bot_1\bot\Downarrow_{\mathsf{c}}$, whence $MP\Downarrow_{\mathsf{c}}$. Now, suppose $B_\sigma \equiv P\bot\bot_1\Downarrow_{\mathsf{c}}$ and $A_\sigma \equiv P\bot\bot\Uparrow_{\mathsf{c}}$. By the non $\lambda\mathsf{C}$-definability of P, we must have $P\bot_1\bot\Uparrow_{\mathsf{c}}$, and so, $NP\Uparrow_{\mathsf{c}}$, and we are done.

II: Let parallel convergence testing be definable in D by p. Then, $[\![M]\!]p\Uparrow$ and $[\![N]\!]p\Downarrow$ in D. □

4.6 Construction of Fully Abstract Models

4.6.1 Overview of the Fully Abstract Submodel Construction and its Proof

The Problem

Let $\mathcal{K} = \langle K, \cdot^{\mathcal{K}}, \Downarrow_{\mathcal{K}} \rangle$ be a *fully adequate* lts, i.e. there is a map $k : K \to D$ that preserves application (in $\mathcal{K}$) and λ-terms such that $\forall M \in \Lambda(K)^o . M\Downarrow_{\mathcal{K}} \Rightarrow D \models M\Downarrow$ where the interpretation of M in D is defined via the map k. We aim to construct $Q^{\mathcal{K}}$, a *retract* of D, which is *fully abstract* with respect to $\mathcal{K}$ by the *restrictive* approach. That is to say $Q^{\mathcal{K}} \xrightarrow{\phi^{\mathcal{K}}} D \xrightarrow{\psi^{\mathcal{K}}} Q^{\mathcal{K}}$ with $\psi^{\mathcal{K}} \circ \phi^{\mathcal{K}} = \mathrm{id}$ and

$$\forall M \in \Lambda(K)^o . [\![M]\!]^{\mathcal{K}} \stackrel{\text{def}}{=} \psi^{\mathcal{K}}([\![M]\!]);$$

satisfying $\forall M, N \in \Lambda(K)^o$,

- $[\![MN]\!]^{\mathcal{K}} = [\![M]\!]^{\mathcal{K}} \cdot^{\mathcal{K}} [\![N]\!]^{\mathcal{K}}$;
- $M \lesssim^{\mathcal{K}} N \iff Q^{\mathcal{K}} \models M \sqsubseteq N$.

In the following, we present a sketch of the general strategy we shall adopt to construct such $Q^{\mathcal{K}}$ for any fully-adequate lts $\mathcal{K}$. However, we are only able to *prove* that $Q^{\mathcal{K}}$ is fully abstract for $\mathcal{K}$ for a restricted class of lts's, which includes $\lambda\ell_c$ and $\lambda\ell_\omega$, a "labelled" version of $\lambda\ell$ (to be introduced later).

Construction of $Q^{\mathcal{K}}$

The construction relies on a *bisimulation logical relation*, $\lessdot^{\mathcal{K}}$, between D and $\mathcal{K}$ which captures the extent to which an element d of D *bisimulates* an element M of $\Lambda(K)^o$ with respect to a suite of *tests* consisting of elements of $\Lambda(K)^o$. $\lessdot^{\mathcal{K}} \subseteq D \times \Lambda(K)^o$ satisfies the following recursive specification: $d \lessdot^{\mathcal{K}} M$ iff

- $\forall \vec{P} \subseteq \Lambda(K)^o . D \models d\vec{P}\Downarrow \Rightarrow \mathcal{K} \models M\vec{P}\Downarrow_{\mathcal{K}}$ &
- $\forall e \in D . \forall N \in \Lambda(K)^o . [e \lessdot^{\mathcal{K}} N \Rightarrow de \lessdot^{\mathcal{K}} MN]$.

Thus, $\lessdot^{\mathcal{K}}$ may be seen as a natural extension of the by now familiar notion of bisimulation to one between two *different* lts's. Intuitively, $d \lessdot^{\mathcal{K}} M$ if "all that can be observed about d in D by applying it to terms in $\Lambda(K)^o$ can equally be observed about M in $\mathcal{K}$". $\lessdot^{\mathcal{K}}$ satisfies the property of *arbitrary join inclusiveness*, i.e. for any $X \subseteq D$

$$[\forall x \in X . x \lessdot^{\mathcal{K}} M] \Rightarrow (\bigsqcup X) \lessdot^{\mathcal{K}} M.$$

Define a preorder $\lesssim^{\mathcal{K}}$ on D as

$$d \lesssim^{\mathcal{K}} e \stackrel{\text{def}}{=} \forall M \in \Lambda(K)^o . e \lessdot^{\mathcal{K}} M \Rightarrow d \lessdot^{\mathcal{K}} M.$$

[7] Abramsky calls it *K-sensible*.

$\precsim^K$ compares the extent to which any two elements in D *bisimulate* elements of $\Lambda(K)^o$. Finally, Q^K is obtained by taking the respective supremums of the equivalence classes induced by the preoder $\precsim^K$.

Proof of Full Abstraction

This we secure by a technique first employed in [Mil77], see also [Mul86]. Construct for the model Q^K and the language K respectively a chain of *approximants* such that, roughly speaking, both the model Q^K and the language K are appropriate *completions* of their respective chains.

We assume that the *canonical projections* $\langle \psi_n \rangle_{n\in\omega}$ *are definable in the language* K by $\langle \Psi_n \rangle_{n\in\omega}$ in the sense that $\forall M, N \in \Lambda(K)^o$, and $n \in \omega$, K satisfies the following:

- $\Psi_0 \sim^K \lambda x.\bot$,
- $\Psi_{n+1} M \Downarrow_K \iff M \Downarrow_K$,
- $\Psi_{n+1} MN \sim^K \Psi_n(M(\Psi_n N))$,

where $\bot$ represents any divergent element in K, *i.e.* $\bot \Uparrow_K$; and that K is *reflective*, *i.e.*

$$\forall M \in \Lambda(K)^o . [\![M]\!] \lessdot^K M.$$

- For each $i \in \omega$, define $\Lambda(K)^o_i$ as the smallest subset of $\Lambda(K)^o$ containing

  $$\{ \Psi_i M : M \in \Lambda(K)^o \}$$

  closed under application and $\sim^K$. $K_i \stackrel{\text{def}}{=} \langle \Lambda(K)^o_i, \Downarrow_K \rangle$, the *i-th approximant of the language* K, is a well-defined q-aswd and denote the associated bisimulation ordering as $\sqsubseteq^K_i$.
- Define for each $i \in \omega$, Q^K_i, the *i-th approximant of the model*, consisting of the respective supremums of the intersection of D_i and the equivalence classes induced by $\precsim^K$.

Full abstraction of the *completion*, *i.e.* $\forall M, N \in \Lambda(K)^o$,

$$M \sqsubseteq^K N \iff Q^K \models M \sqsubseteq N;$$

then follows from the full abstraction of the approximants, *i.e.* $\forall M, N \in \Lambda(K)^o_i$

$$M \sqsubseteq^K_i N \iff Q^K_i \models M \sqsubseteq N$$

by a continuity argument. To summarize, we solve the above-mentioned problem for a class of lts as follows:

THEOREM **4.6.1.1 (Full Abstraction)** *Let K be a fully-adequate, reflexive lts in which the projection functions are internally definable. Then, $\forall M, N \in \Lambda(K)^o$*

$$M \sqsupseteq^K N \iff Q^K \models M \sqsubseteq N.$$

□

$\lambda\ell_C$ satisfies the premises of the Theorem, hence a fully abstract model which is a retract of D exists (and can be constructed) for it. Similarly for $\lambda\ell$. The same method reduces the full abstraction problem for $\lambda\ell$ to an open question about *conservativity* of $\lambda\ell_\omega$ over $\lambda\ell$.

Complementarity of C in the Lazy Regime

The convergence testing constant C introduced to $\lambda\ell_C$ enables the projection functions $\langle\psi_n\rangle_{n\in\omega}$ to be *internally definable* thereby making it possible to enunciate finite domain-theoretic information within the language $\lambda\ell_C$. The domain-theoretic role C plays is clear: the *lifted* space $D_\perp$ is just unitary *separated sum* and C constitutes the corresponding *discriminatory function* [Plo81] i.e. the "elimination" operation concomitant to the "introduction" operation up_D.

That convergence testing *complements* lazy λ-calculus is reinforced further from a category-theoretic perspective. In Chapter 5, we introduce a formal proof system λ_L based on Scott's logic of existence [Sco79] which is *correct* (see [Plo75] for definition) w.r.t. $\lambda\ell$ and may be given a sound interpretation in partial categories. The interpretation is *complete* only for the subclass of λ_L in which convergence testing is definable. These results lead us to conclude that a foundational treatment of lazy functional programming in the framework of the pure untyped λ-calculus should include as fundamental a device for testing convergence. We propose $\lambda\ell_C$ as such a framework.

4.6.2 Approximants of the Language: $\mathcal{K}_i = \langle \Lambda(K)^o_i, \Downarrow_K \rangle$

In this section, we make precise the notion of the *approximants* of languages (which are lts's) given the assumption that *projection maps* are internally definable. The projection maps in question are essentially appropriate *syntactic* representations of the canonical projection maps form D to D_n. For this class of languages which includes $\lambda\ell_C$ and $\lambda\ell_\omega$ (to be introduced), we prove an *Operational*

Approximation Lemma which relates bisimulation preorders of the *sub-languages* to the *full* language.

Let $\mathcal{K} = \langle K, \Downarrow_K \rangle$ be a lts. Denote the associated bisimulation preorder and equivalence by $\sqsubseteq^{\mathcal{K}}$ and $\sim^{\mathcal{K}}$ respectively.

DEFINITION **4.6.2.1** (i) We say that *projection maps are definable* in $\mathcal{K}$ if there exists $\langle \Psi_n \in \Lambda(K)^o : n \in \omega \rangle$ such that for all $M, N \in \Lambda(K)^o$ and $n \in \omega$, $\mathcal{K}$ satisfies:

- $\Psi_0 \sim^{\mathcal{K}} \lambda x.\bot$,
- $\Psi_{n+1} M \Downarrow_K \iff M \Downarrow_K$,
- $\Psi_{n+1} MN \sim^{\mathcal{K}} \Psi_n(M(\Psi_n N))$;

where $\bot$ represents any divergent element in $\mathcal{K}$, *i.e.* $\bot \Uparrow_K$. Note that it follows, by an inductive argument, that $\mathcal{K}$ satisfies

$$\Psi_m(\Psi_n M) \sim^{\mathcal{K}} \Psi_{\min(m,n)} M;$$

where $\min(m,n)$ is the minimum of m and n.

(ii) For $i \in \omega$, define $\Lambda(K)^o_i$ inductively as follows:

$$\frac{M \in \Lambda(K)^o}{\Psi_i M \in \Lambda(K)^o_i} \qquad \frac{M_1, \cdots, M_m \in \Lambda(K)^o_i \quad m \geqslant 1}{(M_1 \cdots M_m) \in \Lambda(K)^o_i}$$

$$\frac{M \in \Lambda(K)^o_i \quad M \sim^{\mathcal{K}} N}{N \in \Lambda(K)^o_i}.$$

In other words, $\Lambda(K)^o_i$ is the least subset of $\Lambda(K)^o$ containing

$$\{ \Psi_i M : M \in \Lambda(K)^o \}$$

which is closed under application and $\sim^{\mathcal{K}}$.

NOTATION **4.6.2.2** We abbreviate $\Psi_i M$ as M^i.

Clearly, $\mathcal{K}_i \stackrel{\text{def}}{=} \langle \Lambda(K)^o_i, \Downarrow_K \rangle$, the *i-th approximant of* $\mathcal{K}$ with $\Downarrow_K$ subject to the obvious restriction is an aswd. Denote the associated bisimulation preorder and equivalence by $\sqsubseteq^{\mathcal{K}}_i$ and $\sim^{\mathcal{K}}_i$ respectively.

REMARK **4.6.2.3** In the above Definition, we have assumed that $\mathcal{K}$ is a lts. However, for the notions just introduced (*i.e.* definability of projection maps and approximants of the language) to make sense, it suffices for $\mathcal{K}$ to be a q-aswd with an interpretation of λ-terms. Specifically, the satisfaction of the axiom (aswd) is *not* required.

REMARK **4.6.2.4** (i) By definition,

$$\forall M \in \Lambda(K)^o_i.\exists M_1,\cdots,M_m \in \Lambda(K)^o, m \geqslant 1.M \sim^K M^i_1 \cdots M^i_m.$$

(ii) $\forall M \in \Lambda(K)^o_i.M^i \sim^K M$. This is because,

$$\begin{array}{lll} M^i & \sim^K & (M^i_1 \cdots M^i_m)^i \qquad\qquad \text{By (i)} \\ & \sim^K & ((M_1 M_2^{i-1} \cdots M_m^{i-m+1})^{i-m+1})^i \quad \text{By Definition 4.6.2.1(i)} \\ & \sim^K & (M_1 M_2^{i-1} \cdots M_m^{i-m+1})^{i-m+1} \\ & \sim^K & M. \end{array}$$

(iii) For the same reason as (ii), we have $\forall M \in \Lambda(K)^o_i.M \sim^K M^{i+1}$ which implies that $\forall i \in \omega.\Lambda(K)^o_i \subseteq \Lambda(K)^o_{i+1}$.

LEMMA **4.6.2.5 (Operational Approximation)** *Let $K = \langle K, \Downarrow_K \rangle$ be an lts in which projection maps are definable and let $i \in \omega$.*

(i) Let $M, N \in \Lambda(K)^o$. Then, $M \sqsubseteq^K N \Rightarrow \forall i \in \omega.M^i \sqsubseteq^K_i N^i$.

(ii) Let $M, N \in \Lambda(K)^o_i$. Then, $M \sqsubseteq^K_i N \Rightarrow M \sqsubseteq^K N$.

PROOF (i) By definition of $\sqsubseteq^K$. For (ii), let $M, N \in \Lambda(K)^o_i$ such that $M \sqsubseteq^K_i N$. Let $X_1,\cdots,X_l \subseteq \Lambda(K)^o$ for $l \in \omega$. Then, by (ii) of Remark 4.6.2.4,

$$\begin{array}{lll} MX_1 \cdots X_l & \sim^K & M^i X_1 \cdots X_l \qquad\qquad \text{Definition 4.6.2.1} \\ & \sim^K & (MX_1^{i-1} \cdots X_l^{i-l})^{i-l}. \end{array}$$

Observe that $X_j^{i-j} \in \Lambda(K)^o_i$. Now, by Definition 4.6.2.1

$$\begin{array}{lll} MX_1 \cdots X_l \Downarrow_K & \Rightarrow & MX_1^{i-1} \cdots X_l^{i-l} \Downarrow_K \quad \text{by premise} \\ & \Rightarrow & NX_1^{i-1} \cdots X_l^{i-l} \Downarrow_K \\ & \Rightarrow & NX_1 \cdots X_l \Downarrow_K . \end{array}$$

We have thus shown $M \sqsubseteq^K N$. □

COROLLARY **4.6.2.6** Let $M, N \in \Lambda(K)^o_i$. Then,

$$M \sim^K N \iff M \sim^K_i N.$$

Hence, for $M \in \Lambda(K)^o_i$, $M \sim^K_i M^i$. □

Projection Maps are definable in $\lambda\ell_{\mathsf{C}}$

Recall the sequence of λC-terms $\langle \Psi_n : n \in \omega \rangle$ defining the respective canonical projections of D to D_n.

LEMMA **4.6.2.7** *Let $M, N, P \in \Lambda(\mathsf{C})$ and $n, m \in \omega$.*

(i) $M\Downarrow_{\mathsf{x}} \iff \Psi_{n+1}M\Downarrow_{\mathsf{x}}$.

(ii) $\Psi_n(\Psi_m P) \sim^{\mathsf{c}} \Psi_{\min(m,n)}P$.

(iii) $(\Psi_{n+1}M)N \sim^{\mathsf{c}} \Psi_n(M(\Psi_n N)) \sim^{\mathsf{c}} (\Psi_{n+1}M)(\Psi_n N)$.

Hence, projection maps are definable in $\lambda\ell_{\mathsf{C}}$.

PROOF (i) is immediate from the definition of Ψ_{n+1}. We prove the first equivalence of (iii) by induction on n. The base case of $n = 0$ is trivial.

$$\begin{aligned}(\Psi_{n+1}M)N &\sim^{\mathsf{c}} \mathsf{C}M(\lambda y.\Psi_n(M(\Psi_n y)))N \\ &\sim^{\mathsf{c}} \mathsf{C}M(\Psi_n(M(\Psi_n N))).\end{aligned}$$

The last step is justified because $\mathsf{C}MPN \sim^{\mathsf{c}} \mathsf{C}M(PN)$ which may be seen to hold by a simple case analysis according to whether $M\Downarrow_{\mathsf{x}}$ or $M\Uparrow_{\mathsf{x}}$. Similarly, by definition of Ψ_n, we have $\mathsf{C}M(\Psi_n(M(\Psi_n N)) \sim^{\mathsf{c}} (\Psi_n(M(\Psi_n N))$. Hence, $(\Psi_{n+1}M)N \sim^{\mathsf{c}} \Psi_n(M(\Psi_n N))$.

Next, we prove (ii) by induction on $n + m = k$. The base cases of $k = 0$ and $n, m \leqslant 1$ are trivial. Suppose true for $n + m = k$. Now, let $m + n = k + 1$. Wlog, assume $n, m \geqslant 1$.

$$\begin{aligned}\Psi_n(\Psi_m P) &\sim^{\mathsf{c}} \mathsf{C}(\Psi_m P)(\lambda y.\Psi_{n-1}((\Psi_m P)(\Psi_{n-1}y))) && \text{(i) and (iii)} \\ &\sim^{\mathsf{c}} \mathsf{C}P(\lambda y.\Psi_{n-1}(\Psi_{m-1}(P(\Psi_{m-1}(\Psi_{n-1}y))))) && \text{ind. hyp.} \\ &\sim^{\mathsf{c}} \mathsf{C}P(\lambda y.\Psi_{\min(n-1,m-1)}(P(\Psi_{\min(n-1,m-1)}))) \\ &\sim^{\mathsf{c}} \Psi_{\min(n,m)}P.\end{aligned}$$

Finally, we apply (ii) and the first equivalence of (iii) to prove the second equivalence of (iii).

$$\begin{aligned}(\Psi_{n+1}M)(\Psi_n N) &\sim^{\mathsf{c}} \Psi_n(M(\Psi_n(\Psi_n N))) && \text{(ii)} \\ &\sim^{\mathsf{c}} \Psi_n(M(\Psi_n N)) \\ &\sim^{\mathsf{c}} (\Psi_{n+1}M)N.\end{aligned}$$

□

The Lambda Transition System $\lambda\ell_\omega = \langle(\Lambda^\omega)^o, \Downarrow_\omega\rangle$

Observe that projection maps are not definable in $\lambda\ell$. This is because the *only* λ-term X that *discriminates* between convergent and divergent elements in the following elementary way

$$\forall M \in \Lambda^o. \begin{cases} XM\Downarrow & \text{if } M\Downarrow, \\ XM\uparrow\uparrow & \text{if } M\uparrow\uparrow; \end{cases}$$

is the identity **I**.

We therefore define a new q-aswd $\lambda\ell_\omega = \langle(\Lambda^\omega)^o, \Downarrow_\omega\rangle$ with $\Lambda^\omega \stackrel{\text{def}}{=} \Lambda(\langle \Psi_n : n \in \omega\rangle)$ which is essentially $\lambda\ell$ augmented with the projection *constants.* Note that the formal constants Ψ_n here are not to be confused with the λC-terms called by the same name which were introduced previously: the context in which they occur should clarify which of the two is meant.

DEFINITION **4.6.2.8** We write $\Lambda^\omega \stackrel{\text{def}}{=} \Lambda(\langle \Psi_n : n \in \omega\rangle)$. The binary reduction relation $\Downarrow_\omega \subseteq (\Lambda^\omega)^o \times (\Lambda^\omega)^o$ is defined inductively as follows:

$$\frac{}{\Psi_n \Downarrow_\omega \Psi_n} \qquad \frac{}{\lambda x.P \Downarrow_\omega \lambda x.P}$$

$$\frac{M \Downarrow_\omega \Psi_{n+1} \quad N\Downarrow_\omega}{MN \Downarrow_\omega \lambda y.\Psi_n(N(\Psi_n y))} \qquad \frac{M \Downarrow_\omega \lambda x.P \quad P[x := Q] \Downarrow_\omega N}{MQ \Downarrow_\omega N}.$$

Define the bisimulation preorder $\lesssim^\omega$ and equivalence $\sim^\omega$ accordingly.

We provide a *relative interpretation* of $\Downarrow_\omega$ in $\Downarrow_{\mathsf{C}}$ via a syntactic translation

$$(-)' : \Lambda^\omega \to \Lambda(\mathsf{C})$$

where $M' \stackrel{\text{def}}{=}$ "M with all occurrences of the *formal constants* Ψ_i replaced by the λC*-terms* Ψ_i for each respective $i \in \omega$."

LEMMA **4.6.2.9** *Let* $M \in \Lambda^\omega$. $M \Downarrow_\omega N \iff M' \Downarrow_{\mathsf{C}} N'$.

PROOF Straightforward induction according to the rules in Definitions 4.6.2.8 and 4.4.1.2 respectively. □

As a helpful consequence, we have the following:

COROLLARY **4.6.2.10** *Let* $M, N \in (\Lambda^\omega)^o$. *Then,* $M' \lesssim^{\mathsf{C}} N' \Rightarrow M \lesssim^\omega N$.
The converse is not true — just consider the pair of terms: $M \equiv x(x(\mathbf{K}\Omega)\Omega)(\mathbf{K}\Omega)$ and $N \equiv x(\lambda y.x(\mathbf{K}\Omega)\Omega y)(\mathbf{K}\Omega)$. The proof that $M \not\sim^\omega N$ requires a careful syntactic case analysis.

Since projection maps are definable in $\lambda\ell_c$, by an appeal to the preceding Corollary, we conclude that

LEMMA **4.6.2.11** *Projection maps are definable in* $\lambda\ell_\omega$. □

PROPOSITION **4.6.2.12** **(Contextual Equivalence)** $\forall M, N \in (\Lambda^\omega)^o$:

$$M \precsim^\omega N \iff \forall C[\,] \in (\Lambda^\omega)^o . C[M]{\Downarrow_\omega} \Rightarrow C[N]{\Downarrow_\omega}.$$

Hence, $\lambda\ell_\omega$ *is an aswd, i.e.* $\precsim^\omega$ *is a logical relation.*

PROOF The proof uses some domain logic notions of Abramsky. We will assume the reader's familiarity with [Abr87, Chap 6] and use his notations.

Define for $n \in \omega$, $F_n : \mathcal{L} \to \mathcal{L}$ where $\mathcal{L}$, the domain logic, is defined on page 188 of *op. cit.*

$$\begin{aligned} F_0(\phi) &\stackrel{\text{def}}{=} t, \\ F_{n+1}(t) &\stackrel{\text{def}}{=} t, \\ F_{n+1}(\phi \wedge \psi) &\stackrel{\text{def}}{=} F_{n+1}(\phi) \wedge F_{n+1}(\psi), \\ F_{n+1}((\phi \to \psi)_\perp) &\stackrel{\text{def}}{=} (F_n(\phi) \to F_n(\psi))_\perp. \end{aligned}$$

Note that $F_n(F_n\phi) = F_n(\phi)$.

Define $\mathbf{P}_n \stackrel{\text{def}}{=} \{\, (F_n(\phi) \to F_n(\phi))_\perp : \phi \in \mathcal{L} \,\}^\dagger \in \mathsf{Filt}\mathcal{L}$. We claim

$$\forall n \in \omega . \mathbf{P}_n \subseteq t_{\lambda\ell_\omega}(\mathbf{\Psi}_n)$$

where $t_{\lambda\ell_\omega}$ is the unique "logic"-preserving morphism (of the category **LTS**) from $\lambda\ell_\omega$ to D as defined on page 197 in *op. cit.*

The argument then follows that which Abramsky developed to prove Theorem 6.6.11 (of which this Proposition is an analogue) found on page 210 of *op. cit.*

We prove the claim by induction on n. The base case is trivial. Suppose the claim is true for some $n \geqslant 0$. Let $M \vDash_{\mathcal{L}} F_{n+1}(\phi) \equiv (\psi \to (\psi' \to \psi'')_\perp)_\perp$ for some $\phi \in \mathcal{L}$. Note that $(\psi' \to \psi'')_\perp \equiv F_n(\theta)$ and $\psi \equiv F_n(\theta')$ for some $\theta, \theta' \in \mathcal{L}$.

Then, it suffices to prove

$$(\dagger) \qquad \mathbf{\Psi}_{n+1} M \vDash_{\mathcal{L}} (\psi \to (\psi' \to \psi'')_\perp)_\perp.$$

Clearly, $\mathbf{\Psi}_{n+1} M {\Downarrow_\omega}$, and so, to prove (†), it suffices to prove:

$$(\ddagger) \qquad N \vDash_{\mathcal{L}} \psi \Rightarrow \mathbf{\Psi}_{n+1} MN \sim^\omega \mathbf{\Psi}_n(M(\mathbf{\Psi}_n N)) \vDash_{\mathcal{L}} (\psi' \to \psi'')_\perp;$$

the preceding equivalence $\sim^\omega$ is justified because projection maps are definable in $\lambda\ell_\omega$.

By induction hypothesis, $\forall \phi \in \mathcal{L} . \mathbf{\Psi}_n \vDash_{\mathcal{L}} (F_n(\phi) \to F_n(\phi))_\perp$. It follows that $\mathbf{\Psi}_n N \vDash_{\mathcal{L}} \psi \equiv F_n(\theta')$, and so $M(\mathbf{\Psi}_n N) \vDash_{\mathcal{L}} (\psi' \to \psi'')_\perp \equiv F_n(\theta)$. (‡) then follows from another appeal to the induction hypothesis. □

An important corollary of the Proposition is the satisfaction of the axiom (aswd); and so, it is easy to see that $\lambda\ell_\omega$ (*i.e.* the associated quotient structure $\langle(\Lambda^\omega)^o/\sim^\omega, \Downarrow_\omega\rangle$) is a lts.

An Open Question

Λ^ω is reminiscent of the classical labelled λ-terms (see Chapter 3): the only difference is that the *labels* themselves *i.e.* Ψ_i, which belong to Λ^ω, are not labelled terms. Λ^o may be seen as the collection of *limits* of all "partial" terms (*i.e.* those with labels) in $(\Lambda^\omega)^o$ with respect to the obvious ordering induced by the labels. In other words, given any term $M \in (\Lambda^\omega)^o$, there is a term $|M| \in \Lambda^o$ defined by replacing all Ψ_i with $\mathbf{I}$, which M approximates. Now, it is obvious that for $M, N \in \Lambda^o$,

$$M \sqsubseteq^\omega N \Rightarrow M \sqsubseteq^B N,$$

because $\Downarrow_\omega$ extends $\Downarrow$ conservatively. However, is the following true?

OPEN QUESTION **4.6.2.13** *Let* $M, N \in \Lambda^o$.

(i) $M \sqsubseteq^B N \Rightarrow M \sqsubseteq^\omega N$?

(ii) $M \sim^B N \Rightarrow M \sim^\omega N$?

The fact that $(\Lambda^\omega)^o$ is a superset of Λ^o gives us some grounds to suspect the *falsity* of the above implications because in general, with an enlarged suite of tests available in $(\Lambda^\omega)^o$, one expects to differentiate more finely between terms. However, the new tests consist in *approximants* of terms which *already exist* in Λ^o, from which one might perhaps surmise that nothing *really* new or of much consequence is introduced in $(\Lambda^\omega)^o$. We shall see that if Open Question 4.6.2.13(i) is true, then our argument for the construction of a fully abstract model for $\lambda\ell$ is valid.

4.6.3 Bisimulation Logical Relations and their Properties

In this section, we study *bisimulation logical relations.* Hitherto, we have considered only how an element of a lts might *bisimulate* another element of the *same* lts, and this with respect to a suite of tests (which are again elements of the same lts) consisting of "inputs arguments" to be applied. Bisimulation logical relations may be regarded as the natural extension of the notion of bisimulation to one between elements of *different* lts's. Given any lts $\mathcal{K}$, $\lessdot^{\mathcal{K}}$, the bisimulation logical relation between D and $\mathcal{K}$ is by construction a *logical relation.* Further, $\lessdot^{\mathcal{K}}$ is *arbitrary join inclusive.* We consider specifically $\lessdot$, $\lessdot^\omega$ and $\lessdot^c$, bisimulation

logical relations between D and $\lambda\ell$, between D and $\lambda\ell_\omega$ and between D and $\lambda\ell_c$ respectively. We show that all three are *reflexive i.e.* $\forall M \in \Lambda(K)^o.[\![M]\!] \lessdot^K M$.

Introduction

In this section, we introduce a class of hereditarily defined relations known as *logical relations*. Logical relations were used by Gordon Plotkin in [Plo73] to tackle definability problems in lambda calculus and domain theory. We refer the reader to [Sta85] for further historical remarks and a survey of some powerful applications of logical relations. In our work, given a lts K, logical relations are employed to construct retracts of D with accompanying *homomorphic* semantic functions with respect to the interpretation of λK-terms. This is needed specifically for the construction of fully abstract models for $\lambda\ell_c$ and $\lambda\ell_\omega$. The connection between logical relations and the existence of surjective homomorphic functions is crucial to our construction, as is the case in most applications — a point emphasized in [AL86].

Following [Plo73], we introduce the notion of a logical relation:

DEFINITION **4.6.3.1** A relation $R \subseteq \Pi_{i\in I} K_i$, where each $K_i \stackrel{\text{def}}{=} \langle K_i, \Downarrow_i\rangle$ is a lts, is a *logical relation* if $\vec{d} \in R \Rightarrow \forall \vec{e} \in R.\vec{d}\cdot\vec{e} \in R$ where $\vec{d}\cdot\vec{e} \stackrel{\text{def}}{=} \langle d_i e_i : i \in I\rangle$.

NOTATION **4.6.3.2** (i) In the following, let $K \stackrel{\text{def}}{=} \langle K, \Downarrow_K\rangle$ range over lambda transition systems with a prescribed interpretation in D. The associated bisimulation preorder and equivalence are denoted $\precsim^K$ and $\sim^K$ respectively.

(i) For simplicity, we will omit the decorated brackets of the semantic function $[\![-]\!]$ in the following context: for $d \in D$ and $\vec{P} \subseteq \Lambda(K)^o$,

$$d\vec{P} \equiv dP_1\cdots P_n \stackrel{\text{def}}{=} d[\![P_1]\!]\cdots[\![P_n]\!];$$

where $[\![P]\!]$ is the denotation in D obtained by mapping each $k \in K$ to D via the prescribed interpretation.

Specification

We prescribe a recursive specification of the *bisimulation logical relations* we seek: $\lessdot^K \subseteq D \times \Lambda(K)^o$ where $d \lessdot^K M$ iff

$$\forall \vec{P} \subseteq \Lambda(K)^o.[D \vDash d\vec{P}\Downarrow \Rightarrow K \vDash M\vec{P}\Downarrow_K] \quad \&$$

$$\forall e \in D.\forall N \in \Lambda(K)^o.[e \lessdot^K N \Rightarrow de \lessdot^K MN].$$

REMARK **4.6.3.3** Let $a, b \in K$. Recall that $a \sqsubseteq^K b \iff \forall \vec{d} \subseteq K.a\vec{d}\Downarrow_K \Rightarrow b\vec{d}\Downarrow_K$. The intended reading of the bisimulation ordering is "for all sequences of *tests*, $\vec{d}$, to which a and b are subjected, all that can be observed about a can equally be observed about b". Note that the tests are themselves elements of the *same* lts and that only convergence is observable.

The logical relation $d \lessdot^K M$ may be seen as a generalization of the notion of bisimulation ordering to one between elements of *possibly different* lts's. We may read $d \lessdot^K M$ as "for all sequences of elements of K (or equivalently, λK-definable elements of D) as tests, all that can be observed about d can equally be observed about M". In other words, $d \lessdot^K M$ means that with respect to the elements of K as tests, M contains as least as much operational information as d.

The second implication in the above recursive specification (which is what qualifies $\lessdot^K$ as a *logical* relation) may be seen as a generalized notion of precongruence between elements of two (or more) different lts's.

DIRECTION **4.6.3.4** This suggests a theory of bisimulation between two or more different lts's parametrized over the same suite of tests.

Existence of the Bisimulation Logical Relation $\lessdot^K$

Following [MP87], we show that a non-trivial relation satisfying the above recursive specification does indeed exist and prove some salient properties satisfied by the relation. Note that the existence proof is general enough to apply to bisimulation logical relations between any two lts's, provided the first lts has *lazy approximable application.*

We begin by defining inductively a family of relations $\lessdot^K_n$ on $D_n \times \Lambda(K)^o$ which are intended to be successive approximations to the targeted logical relation $\lessdot^K$. Recall the following embedding-projection pairs:

$$D_n \overset{\phi_n}{\hookrightarrow} D \overset{\psi_n}{\twoheadrightarrow} D_n, \qquad D_n \overset{i_n}{\hookrightarrow} D_{n+1} \overset{j_n}{\twoheadrightarrow} D_n.$$

DEFINITION **4.6.3.5** For $n \in \omega$, define inductively $\lessdot^K_n \subseteq D_n \times \Lambda(K)^o$ as follows:

$$\begin{aligned} d \lessdot^K_0 M &\overset{\text{def}}{=} \forall \vec{P} \subseteq \Lambda(K)^o.\phi_0(d)\vec{P}\Downarrow \Rightarrow M\vec{P}\Downarrow_K; \\ d \lessdot^K_{n+1} M &\overset{\text{def}}{=} \forall \vec{P} \subseteq \Lambda(K)^o.\phi_{n+1}(d)\vec{P}\Downarrow \Rightarrow M\vec{P}\Downarrow_K \quad \& \\ &\qquad \forall e \in D_n.\forall N \in \Lambda(K)^o.[e \lessdot^K_n N \Rightarrow de \lessdot^K_n MN]. \end{aligned}$$

Note that since D_0 is the singleton set $\{\perp\}$ and $\phi_0(\perp) = \perp$, the antecedent in the r.h.s. of the definition of $\perp \lessdot_0^K M$ is never satisfied, hence vacuously,

$$\forall M \in \Lambda(K)^o.\perp \lessdot_0^K M.$$

D_1 is a two-element set with the top element equal to $\perp_1$ — the least convergent element of D. It is easy to see that $\forall M \Downarrow_K . \perp_1 \lessdot_1^K M$.

The targeted relation, $\lessdot^K$, is obtained as the "conjunction" of all $\lessdot_n^K$, as n ranges over ω, which we will make precise later in the subsection.

We remark the following useful fact.

LEMMA **4.6.3.6** *Let $D \overset{F}{\hookrightarrow} E \overset{G}{\twoheadrightarrow} D$ be an embedding of complete lattices D into E; i.e. F, G are continuous maps satisfying $F \circ G \sqsubseteq \mathrm{id}$ and $G \circ F = \mathrm{id}$. Then, F preserves arbitrary joins.*

PROOF By the Adjoint Functor Theorem. □

Applying the above Lemma, we observe that the embeddings $\phi_n : D_n \hookrightarrow D$ preserve arbitrary joins. To all intents and purposes, we may regard the embeddings, ϕ_n, simply as set-theoretic inclusions.

Let us establish some properties satisfied by the family of relations $\lessdot_n^K$. Note that all the results for the rest of this subsection culminating in Lemma 4.6.3.13 apply, in particular, to $\lessdot_n^c$, $\lessdot_n$ and $\lessdot_n^\omega$.

LEMMA **4.6.3.7** $\forall n \in \omega . \forall d, d' \in D_n . \forall M \in \Lambda(K)^o . d' \sqsubseteq d \lessdot_n^K M \Rightarrow d' \lessdot_n^K M.$

PROOF We prove by induction on n. The base case is trivial. Suppose true for n. Let $d' \sqsubseteq d \lessdot_{n+1}^K M$. Applying the monotonicity of "$\cdot$" repeatedly,

$$\phi_{n+1}(d')\vec{P}\Downarrow \Rightarrow \phi_{n+1}(d)\vec{P}\Downarrow \Rightarrow M\vec{P}\Downarrow_K .$$

Suppose $e \lessdot_n^K N$. Then, $d'e \sqsubseteq de \lessdot_n^K MN$. By induction hypothesis, $d'e \lessdot_n^K MN$ and we are done. □

LEMMA **4.6.3.8** *For each $n \in \omega$, $\lessdot_n^K$ is arbitrary join inclusive in its first argument; i.e. $\forall X \subseteq D_n . [\forall x \in X . x \lessdot_n^K M \Rightarrow \bigsqcup X \lessdot_n^K M]$.*

PROOF By induction on n. Again, the base case is trivial. Suppose true for n. Let $X \subseteq D_{n+1}$ and $\forall x \in X . x \lessdot_{n+1}^K M$. We will show $\bigsqcup X \lessdot_{n+1}^K M$. Suppose $\phi_{n+1}(\bigsqcup X)\vec{P}\Downarrow$. Recall that the projection ϕ_{n+1} preserves arbitrary joins. Also, application left-preserves arbitrary joins Lemma 4.3.5.2 (and applying it repeatedly), we deduce that $\phi_{n+1}(x)\vec{P}\Downarrow$ for some x. By supposition, $M\vec{P}\Downarrow_K$.

It remains to show $e \lessdot_n^K N \Rightarrow (\bigsqcup X) \cdot e \lessdot_n^K MN$. Since application (left-) preserves arbitrary joins, $(\bigsqcup X) \cdot e = \bigsqcup_{x \in X} x \cdot e$. Observe that $\forall x \in X . xe \lessdot_n^K MN$ by supposition. Invoking the induction hypothesis, we have $\bigsqcup_{x \in X} x \cdot e \lessdot_n^K MN$ and we are done. □

For any M, we have already observed that $\bot \lessdot^K_n M$. The preceding Lemma establishes that $\{ d \in D_n : d \lessdot^K_n M \} \subseteq D_n$ is a downward-closed set which is also closed under *arbitrary* (not just directed) joins. It is therefore, a *fortiori*, Scott-closed.

LEMMA **4.6.3.9** *(i) If $d \lessdot^K_n M$ then $i_n(d) \lessdot^K_{n+1} M$.*

(ii) If $d \lessdot^K_{n+1} M$ then $j_n(d) \lessdot^K_n M$.

PROOF We prove by simultaneous induction on n. The base cases for (i) and (ii) are trivial. Suppose assertions (i) and (ii) are true for n. For (i): Suppose $d \lessdot^K_{n+1} M$, we want to prove $i_{n+1}(d) \lessdot^K_{n+2} M$. Observe that $\phi_{n+2}(i_{n+1}(d)) = \phi_{n+1}(d)$. Hence,

$$\phi_{n+2}(i_{n+1}(d))\vec{P}\Downarrow \Rightarrow M\vec{P}\Downarrow_K.$$

Ir remains to prove: $e \lessdot^K_{n+1} N \Rightarrow i_{n+1}(d) \cdot e \lessdot^K_{n+1} MN$. By induction hypothesis in (ii), $j_n(e) \lessdot^K_n N$. Hence, $d(j_n(e)) \lessdot^K_n MN$. Now, by induction hypothesis in (i), $i_n(d(j_n(e)) \lessdot^K_{n+1} MN$ which is just $i_{n+1}(d) \cdot e \lessdot^K_{n+1} MN$. The proof for (ii) is similar and we omit it. □

We are now in a position to define the relation $\lessdot^K$ for which we have hitherto been labouring in this subsection.

DEFINITION **4.6.3.10** Define $\lessdot^K \subseteq D \times \Lambda(K)^o$ as follows:

$$d \lessdot^K M \stackrel{\text{def}}{=} \forall n \in \omega . d_n \lessdot^K_n M.$$

Defined in this way, $\lessdot^K$ inherits properties satisfied by its "approximants", $\lessdot^K_n$. A first such inherited property is what we loosely term as the Scott-closedness (in the first argument) of $\lessdot^K$. In fact, $\lessdot^K$ satisfies a stronger property: it is *arbitrary join inclusive* which we will show later.

LEMMA **4.6.3.11** (**Scott-Closedness**) *For each $M \in \Lambda(K)^o$, the set*

$$\lessdot^{K^{-1}}(M) \stackrel{\text{def}}{=} \{ d \in D : d \lessdot^K M \} \text{ is Scott-closed.}$$

PROOF First, we have already seen that $\forall M . \bot \lessdot^K M$. Let $d' \sqsubseteq d \lessdot^K M$. By definition, $\forall n \in \omega . d_n \lessdot^K_n M$. But we have $\forall n . d'_n \sqsubseteq d_n$, by the monotonicity of the projection ψ_n. Hence, $d' \lessdot^K M$, *i.e.* $\lessdot^{K^{-1}}(M)$ is left-closed. It remains to show that $\lessdot^{K^{-1}}(M)$ is closed under directed joins. Suppose for some directed $X \subseteq D$, $\forall x \in X . x \lessdot^K D$. If $\phi_n((\bigsqcup X)_n)\vec{P}\Downarrow$, then continuity of "$\cdot$" and ϕ_n imply that for some $x \in X$, $\phi_n(x)\vec{P}\Downarrow$; which implies, by supposition, that $M\vec{P}\Downarrow_K$. Let $e \lessdot^K_n N$, for some $e \in D_n$. Then,

$$\phi_{n+1}((\bigsqcup X)_{n+1}) \cdot e = \bigsqcup_{x \in X} \phi_{n+1}(x_{n+1}) \cdot e.$$

Since, by supposition, $\forall x \in X . x_{n+1} \lessdot^K_{n+1} M$, we have $x_{n+1} \cdot e \lessdot^K_n MN$. Finally, inclusiveness of $\lessdot^K_n$ implies that $\bigsqcup_{x \in X} x_{n+1} \cdot e \lessdot^K_n MN$. □

REMARK **4.6.3.12** For each $d \in D$, there is a canonical choice of $\langle d_n \rangle_{n\in\omega}$ where each $d_n \in D$ such that $d = \bigsqcup \phi_n(d_n)$; each d_n is uniquely determined by the n-projection function $\psi_n : D \to D_n$. However, for any sequence $\langle e_n \rangle_{n\in\omega}$ where $e_n \in D_n$ such that

$$d = \bigsqcup_{n\in\omega} \phi_n(e_n),$$

we claim that: $d \lessdot^K M$ iff $\forall n \in \omega . e_n \lessdot^K_n M$.

"$\Rightarrow$" We first note that by Lemma 4.6.3.9, for $d \in D_n$

$$d \lessdot^K_n M \Longleftrightarrow \phi_n(d) \lessdot^K M.$$

Now, for each n, $\phi_n(e_n) \sqsubseteq d \lessdot^K M$. By $\lessdot^K$ being left-closed in the first argument, we have $\phi_n(e_n) \lessdot^K M$. Hence, $e_n \lessdot^K_n M$.

"$\Leftarrow$" $e_n \lessdot^K_n M \Rightarrow \phi_n(e_n) \lessdot^K M$. Then, by directed-join inclusiveness of $\lessdot^K$.

We are now in a position to show that the relation $\lessdot^K$ thus defined does indeed satisfy the recursive specification we stated at the outset.

LEMMA **4.6.3.13** (Recursive Specification) $d \lessdot^K M$ *iff*

(A) $\forall \vec{P} \subseteq \Lambda(K)^\circ . [D \models d\vec{P}\Downarrow \Rightarrow K \models M\vec{P}\Downarrow_K]$ *and*

(B) $\forall e \in D . \forall N \in \Lambda(K)^\circ . [e \lessdot^K N \Rightarrow de \lessdot^K MN]$.

PROOF "$\Rightarrow$": Suppose $d\vec{P}\Downarrow$. By the continuity of "$\cdot$", we have $d\vec{P} = \bigsqcup(d_n\vec{P})$. Hence, for some n, $d_n\vec{P}\Downarrow$, and so by $d_n \lessdot^K_n M$, we have $M\vec{P}\Downarrow_K$. Let $e \lessdot^K N$. Then $e_n \lessdot^K_n N$ and so, as $d_{n+1} \lessdot^K_{n+1} M$, we have $d_{n+1}e_n \lessdot^K_n MN$ and so $\phi_n(d_{n+1}e_n) \lessdot^K MN$. Now, since $de = \bigsqcup_{n\in\omega} \phi_n(d_{n+1} \cdot e_n)$, by Remark 4.6.3.12, we conclude that $de \lessdot^K MN$.
"$\Leftarrow$": We show by induction $\forall n \in \omega . d_n \lessdot^K_n M$. The base case is trivial. Suppose $d_n \lessdot^K_n M$ and we want to establish $d_{n+1} \lessdot^K_{n+1} M$. $\phi_{n+1}(d_{n+1})\vec{P}\Downarrow$ clearly implies, by monotonicity of "$\cdot$", that $d\vec{P}\Downarrow$. $M\vec{P}\Downarrow_K$ then follows by supposition. Let $e \lessdot^K_n N$. By hypothesis, $d\phi_n(e) \lessdot^K MN$. But, $\phi_n(d_{n+1}e) \sqsubseteq d\phi_n(e)$. Since $\lessdot^K$ is left-closed (in the first argument), we have $\phi_n(d_{n+1}e) \lessdot^K MN$ which is equivalent to $d_{n+1}e \lessdot^K_n MN$. □

Bisimulation Logical Relations are Arbitrary Join Inclusive

We can now prove an important property satisfied by the logical relation $\lessdot^K$.

LEMMA **4.6.3.14 (Arbitrary Join Inclusiveness)** $\lessdot^K$ *is arbitrary join inclusive in the first argument, i.e. for any* $X \subseteq D$

$$[\forall x \in X. x \lessdot^K M] \Rightarrow (\bigsqcup X) \lessdot^K M.$$

PROOF By Lemma 4.3.5.1, we have

$$\bigsqcup X = \bigsqcup_{x\in X}\bigsqcup_{n\in\omega} \phi_n(x_n) = \bigsqcup_{n\in\omega}\bigsqcup_{x\in X} \phi_n(x_n) = \bigsqcup_{n\in\omega} \phi_n(\bigsqcup_{x\in X} x_n);$$

the last step is justified because the embedding ϕ_n preserves arbitrary joins. By Remark 4.6.3.12, it suffices to show inductively:

$$\forall n \in \omega. \bigsqcup_{x\in X} x_n \lessdot^K_n M.$$

Base case is again trivial. Suppose $\bigsqcup_{x\in X} x_n \lessdot^K_n M$. Since

$$(\bigsqcup_{x\in X} \phi_{n+1}(x_{n+1}))\vec{P} = \bigsqcup_{x\in X} (\phi_{n+1}(x_{n+1})\vec{P}),$$

$\phi_{n+1}(\bigsqcup_{x\in X} x_{n+1})\vec{P}\Downarrow$ then implies that for some $x \in X$, $\phi_{n+1}(x_{n+1})\vec{P}\Downarrow$. Then, by definition of $x \lessdot^K M$, we deduce that $M\vec{P}\Downarrow_K$.

Suppose $e \lessdot^K_n N$. By definition of $x \lessdot^K M$, $\forall x \in X. x_{n+1} \cdot e \lessdot^K_n MN$. Since $\lessdot^K_n$ is arbitrary join inclusive, we have

$$(\bigsqcup_{x\in X} x_{n+1}) \cdot e = \bigsqcup_{x\in X} (x_{n+1} \cdot e) \lessdot^K_n MN,$$

and we are done. □

LEMMA **4.6.3.15** *For* $d \in D$ *and* $M, N \in \Lambda(K)^o$, $d \lessdot^K M \sqsubseteq^K N \Rightarrow d \lessdot^K N$.

PROOF The lemma may be established by proving $\forall n \in \omega. L_n$ inductively where:

$$\mathrm{L}_n: \quad \forall d \in D_n. \forall M, N \in \Lambda(K)^o. d \lessdot^K_n M \sqsubseteq^K N \Rightarrow d \lessdot^K_n N.$$

The details should be routine by now and we omit it. □

As examples of $\lessdot^K$, we will consider $\lessdot$, $\lessdot^\omega$ and $\lessdot^c$. In the case of $\lambda\ell_\omega$, the formal constants Ψ_i are given the natural interpretations in D, *i.e.* $\mathsf{Gr}(\mathsf{up}\psi_i)$); and for $\lambda\ell_c$, C is interpreted as $\mathsf{Gr}(\mathsf{up}(f_{\perp_1,i}))$ as before.

Reflexive Bisimulation Logical Relations

DEFINITION **4.6.3.16** Let K be a lts with a prescribed interpretation in D. The bisimulation logical relation $\lessdot^K$ is *reflexive* if $\forall M \in \Lambda(K)^o.[\![M]\!] \lessdot^K M$. We say that K is reflexive if $\lessdot^K$ is.

OPEN QUESTION **4.6.3.17** Compare the above notion with the Fundamental Theorem of Logical Relation (see e.g. [Sta85]). Can a similar general result be proved? More specifically, characterize the class of lts $K = \langle K, \Downarrow_K \rangle$ such that $\forall M \in \Lambda(K)^o.[\![M]\!] \lessdot^K M$.

For the rest of this subsection, we will show that $\lessdot$, $\lessdot^{\mathsf{c}}$ and $\lessdot^{\omega}$ are reflexive.

PROPOSITION **4.6.3.18** *For $K = \lambda\ell, \lambda\ell_{\mathsf{c}}, \lambda\ell_{\omega}$, we have $\forall M \in \Lambda(K)^o.[\![M]\!] \lessdot^K M$.*

PROOF We will only prove the case of $\lambda\ell_{\mathsf{c}}$. The rest is similar. Owing to the well-known translation between $\Lambda^o(C)$ and $\mathbf{CL}(C)$ where C is a set of constants and $\mathbf{CL}(C)$ the class of **SK**-combinators over C (see e.g. [Bar84]), and because $\lessdot^{\mathsf{c}}$ is a logical relation, it suffices to prove:

(a) $c \stackrel{\text{def}}{=} [\![\mathsf{C}]\!] \lessdot^{\mathsf{c}} \mathsf{C}$

(b) $k \stackrel{\text{def}}{=} [\![\mathbf{K}]\!] \lessdot^{\mathsf{c}} \mathbf{K}$

(c) $s \stackrel{\text{def}}{=} [\![\mathbf{S}]\!] \lessdot^{\mathsf{c}} \mathbf{S}$.

We omit the proof of (b) since it is similar to and easier than (c). With reference to the Recursive Specification Lemma, observe that in each case, condition (A) is trivially satisfied.

(a) We show: $e \lessdot^{\mathsf{c}} N \Rightarrow ce \lessdot^{\mathsf{c}} \mathsf{C}N$. Suppose the antecedent. Firstly, we establish $\forall \vec{P}.ce\vec{P}\Downarrow \Rightarrow \mathsf{C}N\vec{P}\Downarrow_{\mathsf{c}}$. Wlog, we may assume $e\Downarrow$, hence $N\Downarrow_{\mathsf{c}}$; then the implication is trivially true since $ce = [\![\mathbf{I}]\!]$ and $\mathsf{C}N \Downarrow_{\mathsf{c}} \mathbf{I}$. Now, suppose $f \lessdot^{\mathsf{c}} P$, then $cef = f \lessdot^{\mathsf{c}} P \sim^{\mathsf{c}} \mathsf{C}NP$, and so, we have $cef \lessdot^{\mathsf{c}} \mathsf{C}NP$.

(c) We show $e \lessdot^{\mathsf{c}} N \Rightarrow se \lessdot^{\mathsf{c}} \mathbf{S}N$. We first show

$$(*) \qquad \forall P_1, \cdots, P_n.seP_1 \cdots P_n\Downarrow \Rightarrow \mathbf{S}NP_1 \cdots P_n\Downarrow_{\mathsf{c}}$$

(*) holds trivially for $n \leqslant 1$. For $n \geqslant 2$,

$$seP_1 \cdots P_n = eP_2(P_1P_2)P_3 \cdots P_n;$$

similarly for $\mathbf{S}N\vec{P}$. (*) is then seen to hold because $e \lessdot^{\mathsf{c}} N$. Next, we show $f \lessdot^{\mathsf{c}} P \Rightarrow sef \lessdot^{\mathsf{c}} \mathbf{S}NP$. Suppose the antecedent. By similar argument, we see that $\forall \vec{Q}.sef\vec{Q}\Downarrow_{\mathsf{c}} \Rightarrow \mathbf{S}NP\vec{Q}\Downarrow_{\mathsf{c}}$. Suppose $g \lessdot^{\mathsf{c}} Q$. Then $sefg = eg(fg) \lessdot^{\mathsf{c}} NQ(PQ) \sim^{\mathsf{c}} \mathbf{S}NPQ$ and so, we are done.

□

Operationally-Tailored Preorders on D

Given a lts $K = \langle K, \Downarrow_K \rangle$, we define a preorder $\lesssim^K$ on D which, roughly speaking, compares the extent to which any two elements in D bisimulate elements of K with respect to tests consisting of elements of K. The quotient of D by $\simeq^K$ identifies any two elements of D which are *not* observably distinguishable with respect to elements of K as tests. In certain cases to be specified in the sequel, this quotient structure turns out to be a fully abstract model for the language K.

DEFINITION **4.6.3.19** For $d, e \in D$,

$$d \lesssim^K e \stackrel{\text{def}}{=} \forall M \in \Lambda(K)^o.[e \lessdot^K M \Rightarrow d \lessdot^K M].$$

We abbreviate $d \lesssim^K e \,\&\, e \lesssim^K d$ as $d \simeq^K e$. Denote the $\simeq^K$-equivalence class of d as $[d]^K$.

We will show that $\lesssim^K$ simulates in D the bisimulation relation $\trianglelefteq^K$ of K, *i.e.*

$$\forall M, N \in \Lambda(K)^o.M \trianglelefteq^K N \iff [\![M]\!] \lesssim^K [\![N]\!].$$

Since $\lessdot^K$ is left-closed in its first argument, it follows that $\sqsubseteq$, the domain order of D, is a *refinement* of $\lesssim^K$, *i.e.* $\sqsubseteq \subseteq \lesssim^K$. The following result says that the relation $\lesssim^K$ is component-wise *arbitrary join inclusive.*

LEMMA **4.6.3.20** *Let $X \subseteq D$. Then,*

(i) $[\forall x \in X.x \lesssim^K d] \Rightarrow \bigsqcup X \lesssim^K d.$

(ii) $[\forall x \in X.d \lesssim^K x] \Rightarrow d \lesssim^K \bigsqcup X.$

(iii) $X \subseteq [d]^K \Rightarrow \bigsqcup X \in [d]^K.$

PROOF (i) is a straightforward corollary of the join inclusiveness of $\lessdot^K$. (ii) follows directly from $\sqsubseteq \subseteq \lesssim^K$ and the transitivity of $\lesssim^K$. (iii) follows from (i) and (ii). □

COROLLARY **4.6.3.21** *Let $d, e \in D$. Then, $d \lesssim^K e \Rightarrow \bigsqcup [d]^K \sqsubseteq \bigsqcup [e]^K$.*

PROOF Suppose $d \lesssim^K e$. By (i) of Lemma, $\bigsqcup [d]^K \lesssim^K e$; and so, by (i) again, $\bigsqcup [d]^K \sqcup e \lesssim^K e$. The other direction follows from $\sqsubseteq \subseteq \lesssim^K$. Hence, $\bigsqcup [d]^K \sqcup e \simeq^K e$. It then follows that $\bigsqcup [d]^K \sqsubseteq \bigsqcup [d]^K \sqcup e \sqsubseteq \bigsqcup [e]^K$. □

Next, we establish a crucial relationship between $\precsim^K$ and $\sqsubseteq^K$. First, a Lemma.

LEMMA **4.6.3.22** *Let $M, N \in \Lambda(K)^o$ and $d \in D$. Then,*

$$M \sqsubseteq^K N \Rightarrow \forall d \in D.[d \lessdot^K M \Rightarrow d \lessdot^K N].$$

PROOF We prove $\forall n \in \omega.\mathbf{L}_n$, where $\mathbf{L}_n$ is the following statement:

$$\forall M, N \in \Lambda(K)^o.M \sqsubseteq^K N \Rightarrow \forall d \in D.\forall m \leqslant n.[d_m \lessdot^K_m M \Rightarrow d_m \lessdot^K_m N].$$

The routine details are omitted. □

COROLLARY **4.6.3.23** *Let K be reflexive and fully adequate and let $M, N \in \Lambda(K)^o$. Then, $M \sqsubseteq^K N \iff [\![M]\!] \lessdot^K N$.*

PROOF "$\Rightarrow$": By reflexivity of K *i.e.* $[\![M]\!] \lessdot^K M$ and an appeal to the Lemma. "$\Leftarrow$": Immediate from full adequacy of K and condition (A) of the Recursive Specification Lemma. □

LEMMA **4.6.3.24** *Let K be reflexive and fully adequate, $d \in D$ and $M \in \Lambda(K)^o$. Then, $d \lessdot^K M \iff d \precsim^K [\![M]\!]$.*

PROOF "$\Rightarrow$": Let $d \lessdot^K M$. Suppose $[\![M]\!] \lessdot^K N$ and so, by the above Corollary, $M \sqsubseteq^K N$ which by Lemma 4.6.3.22, implies $d \lessdot^K N$. Hence, $d \precsim^K [\![M]\!]$.
"$\Leftarrow$": Suppose $d \precsim^K [\![M]\!]$. Reflexivity of $\lessdot^K$ and full adequacy then imply $d \lessdot^K M$. □

PROPOSITION **4.6.3.25** (Crucial) *Let K be reflexive and fully adequate and let $M, N \in \Lambda(K)^o$. Then,*

$$[\![M]\!] \precsim^K [\![N]\!] \iff M \sqsubseteq^K N.$$

PROOF

$$\begin{array}{lll} & M \sqsubseteq^K N & \text{Corollary 4.6.3.23} \\ \iff & [\![M]\!] \lessdot^K N & \text{Lemma 4.6.3.24} \\ \iff & [\![M]\!] \precsim^K [\![N]\!]. & \end{array}$$

□

As usual, we define $[d]^K \lesssim^K [e]^K$ if $d \lesssim^K e$, turning $\langle D/\tilde{\sim}^K, \lesssim^K/\tilde{\sim}^K\rangle$ into a partial order. In fact, we can say rather more than that:

LEMMA **4.6.3.26** *$D_{\lesssim^K} = \langle D/\tilde{\sim}^K, \lesssim^K/\tilde{\sim}^K\rangle$ is a complete lattice.*

PROOF It is instructive, at this juncture, to identify the $\lesssim^K$-least element. Clearly, $\forall d.\bot \lesssim^K d$. We claim:

$$d \lesssim^K \bot \Rightarrow d = \bot,$$

which implies that $[\bot]^K = \{\bot\}$. To see this, recall that $\forall M.\bot \lessdot^K M$; and so, for any $d \lesssim^K \bot$ we have:

$$\forall M \in \Lambda(K)^o.\forall \vec{P} \in \Lambda(K)^o.d\vec{P}\Downarrow \Rightarrow M\vec{P}\Downarrow_K.$$

In particular, $\forall M.d\Downarrow \Rightarrow M\Downarrow_K$. Now, $d \neq \bot$ *i.e.* $d\Downarrow$ would lead to a contradiction.

The $\lesssim^K$-largest element is $[\top]^K$.

It remains (and suffices) to show that every subset has a lub. Let $X = \{[d_i]^K : i \in I\} \subseteq D_{\lesssim^K}$. Observe that $\overline{d} \stackrel{\text{def}}{=} \bigsqcup\{d'_i\}$ where $\forall i \in I.d'_i \in [d_i]^K$ exists because D is a complete lattice. We immediately have $\forall i.[d_i]^K \lesssim^K [\overline{d}]^K$ because $d_i \lesssim^K d'_i \sqsubseteq \overline{d}$. Hence, $[\overline{d}]^K$ is an $\lesssim^K$-upper bound of X. Now suppose for some d, $\forall i \in I.[d_i]^K \lesssim^K [d]^K$. Suppose $d \lessdot^K M$. The previous supposition then implies

$$\forall i \in I.d'_i \lessdot^K M.$$

By arbitrary join inclusiveness of $\lessdot^K$, $\overline{d} \lessdot^K M$. Hence, $[\overline{d}]^K \lesssim^K [d]^K$. □

Note that the $\lesssim^K$-lub, $[\overline{d}]^K$ where $\overline{d} \stackrel{\text{def}}{=} \bigsqcup d'_i$ is independent of the choice of representatives form the respective $\tilde{\sim}^K$-equivalence classes $[d_i]^K$.

LEMMA **4.6.3.27** *(i) For $d \in D$, $\bigsqcup_{\sqsubseteq}[d]^K \tilde{\sim}^K d$; equivalently, $\bigsqcup_{\sqsubseteq}[d]^K \in [d]^K$.*

(ii) The map $[-]^K : D \to D_{\lesssim^K}$ defined by $d \mapsto [d]^K$ is continuous, i.e. for $X \equiv \{d_i : i \in I\} \subseteq D$, a $\tilde{\sqsubseteq}$-directed set,

$$\bigsqcup_{\lesssim^K}\{[d_i]^K : i \in I\} = [\bigsqcup_{\sqsubseteq}\{d_i : i \in I\}]^K.$$

PROOF

(i) Direct from Lemma 4.6.3.20(iii).

(ii) $[-]$ is monotonic because $\sqsubseteq \subseteq \lesssim^K$. For the same reason, if $\{d_i : i \in I\}$ is a $\sqsubseteq$-directed subset of D, then, $\{[d_i] : i \in I\}$ is a $\lesssim^K$-directed set. Since $D_{\lesssim^K}$ is a complete lattice, $\bigsqcup_{\lesssim^K}[d_i]$ exists and by the preceding remark,

$$\bigsqcup_{\lesssim^K}\{[d_i] : i \in I\} = [\bigsqcup_{\sqsubseteq}\{d_i : i \in I\}]$$

which is precisely what the continuity of $[-]$ entails.

□

Definition of the Model Q^K

We are now in a position to define a substructure Q^K of D which will turn out to be fully abstract with respect to K subject to certain conditions which will be specified in the sequel.

DEFINITION **4.6.3.28**

Define a map $Q^K : D \to D$ by

$$Q^K(d) \stackrel{\text{def}}{=} \bigsqcup_{\sqsubseteq} [d]^K.$$

Observe that for $d \in D$, $\bigsqcup_{\sqsubseteq} [d]^K \in [d]^K$ implies $[\bigsqcup_{\sqsubseteq} [d]^K]^K = [d]^K$. Hence,

$$Q^K(Q^K(d)) = \bigsqcup [\bigsqcup [d]^K]^K = \bigsqcup [d]^K = Q^K(d).$$

Thus, we have $Q^K \circ Q^K = Q^K$. Note that Q^K so defined is continuous by Corollary 4.6.3.21 and the fact that $\sqsubseteq \subseteq \lesssim^K$; and that $\mathrm{id}_D \sqsubseteq Q^K$. We write the image of the map Q^K as Q^K — no confusion need arise. We remark that Q^K is a retract of D.

4.6.4 Approximants of the Model: Q_i^K

In this subsection, we define Q_i^K, the i-th approximant of Q^K, which we will show, is *fully abstract* with respect to K_i, the i-th approximant of the language $K = \langle K, \Downarrow_K \rangle$. This subsection is devoted to the proof of the "approximant" full abstraction results at each finite level $i \in \omega$. Roughly speaking, the strategy that we shall adopt is to prove the main full abstraction result by establishing it as the "limit" of the approximant full abstraction results at each finite level. Recall that $D_i \stackrel{\text{def}}{=} \psi_i D$ and $[\![\Psi_i]\!] = \psi_i$. For each $d \in D_i$, define $[d]_i^K \stackrel{\text{def}}{=} D_i \cap [d]^K$.

LEMMA **4.6.4.1** *(i) For $i \in \omega$, $d \in D_i$, $\bigsqcup_{\sqsubseteq} [d]_i^K = \psi_i(\bigsqcup_{\sqsubseteq} [d]^K) \in [d]_i^K$.*

(ii) Define $\max_i : D_i \to D_i$ by $\max_i(d) = \bigsqcup [d]_i^K$. Then, $\max_i$ is monotonic, hence continuous since D_i is finite.

(iii) For $d, e \in D. d \lesssim^K e \Rightarrow \bigsqcup [d]_i^K \sqsubseteq \bigsqcup [e]_i^K$.

PROOF

(i) Let $\overline{d} \stackrel{\text{def}}{=} \bigsqcup [d]_i^K$. We first observe that $\overline{d} \in D_i$, because D_i is a complete lattice. Next, we show $\overline{d} \simeq^K d$, equivalently, $\overline{d} \in [d]^K$. Now, $d \lesssim^K \overline{d}$ follows immediately from $\sqsubseteq \subseteq \lesssim^K$. To show $\overline{d} \lesssim^K d$, suppose $d \prec^K M$. By definition of $[d]_i^K$

$$\forall e \in [d]_i^K . e \prec^K M.$$

By arbitrary join inclusiveness of $\prec^K$, $\overline{d} = \bigsqcup [d]_i^K \prec^K M$.
Since $[d]_i^K = [d]^K \cap D_i$, we have $\bigsqcup [d]_i^K \sqsubseteq \bigsqcup [d]^K$. Hence,

$$\bigsqcup [d]_i^K = \psi_i(\bigsqcup [d]_i^K) \sqsubseteq \psi_i(\bigsqcup [d]^K)$$

because $\bigsqcup [d]_i^K \in D_i$. To show $\psi_i(\bigsqcup [d]^K) \sqsubseteq \bigsqcup [d]_i^K$, it suffices to show

$$\psi_i(\bigsqcup [d]^K) \stackrel{-}{\sim}^K d.$$

$d \sqsubseteq \bigsqcup [d]^K$ implies $d = \psi_i(d) \sqsubseteq \psi_i(\bigsqcup [d]^K)$ which implies $d \lesssim^K \psi_i(\bigsqcup [d]^K)$. Now, suppose $d \prec^K M$. Then, by inclusiveness of $\prec^K$, $\bigsqcup [d]^K \prec^K M$. By definition of $\prec^K$, $\forall i \in \omega . \psi_i(\bigsqcup [d]^K) \prec_i^K M$ which is equivalent to $\psi_i(\bigsqcup [d]^K) \prec^K M$ and we are done.

(ii) Let $d, e \in D_i$ such that $d \sqsubseteq e$. Then, we have $d \lesssim^K e$ and so, by Corollary 4.6.3.21, $\bigsqcup [d]^K \sqsubseteq \bigsqcup [e]^K$. Monotonicity of ψ_i and (i) together imply that $\max_i(d) \sqsubseteq \max_i(e)$.

(iii) Similar to (ii).

□

DEFINITION **4.6.4.2** We define $Q_i^K : D \to D_i$ as $Q_i^K(d) = \max_i \circ \psi_i(d) = \bigsqcup [d_i]_i^K$.

Then, for $d \in D_i$,

$$\begin{aligned} Q_i^K \circ Q_i^K(d) &= Q_i^K(\bigsqcup [d]_i^K) \\ &= \bigsqcup [\bigsqcup [d]_i^K]_i^K \\ &= \bigsqcup [d]_i^K \\ &= Q_i^K(d). \end{aligned}$$

The penultimate step is justified because

$$\bigsqcup [d]_i^K \in [d]^K \cap D_i \Rightarrow [\bigsqcup [d]_i^K]_i^K = [d]_i^K.$$

Hence, Q_i^K is a retract of D_i; and hence, of D.

DEFINITION **4.6.4.3** We define, for each $i \in \omega$, a structure $\langle Q_i^K, \cdot, [\![-]\!]^{Q_i^K} \rangle$ where

- "$\cdot$" is just the restriction of "$\cdot$":$D \times D \to D$ to $Q_i^K \times Q_i^K$.
- $[\![-]\!]^{Q_i^K} : \Lambda(K)^o \to Q_i^K$ is defined by $[\![M]\!]^{Q_i^K} \stackrel{\text{def}}{=} Q_i^K \circ ([\![M]\!])$.

Clearly, we need to show that application is well-defined, i.e. $\forall d, e \in Q_i^K . d \cdot e \in Q_i^K$. To do this, we first prove a useful Lemma. Let $x, y \in D$ and x is compact. Define $(x \Rightarrow y)$ to be the element $\mathsf{Gr}(\mathsf{up}(f_{x,y})) \in D$. Note that for $d \in D$,

$$(x \Rightarrow y) \cdot d = \begin{cases} y & \text{if } x \sqsubseteq d, \\ \bot & \text{else.} \end{cases}$$

LEMMA **4.6.4.4** *Let K be reflexive. Suppose $d, c \in D$ such that $dc{\Downarrow}$. Then,*

$$b \lesssim^K dc \Rightarrow (c \Rightarrow b) \lesssim^K d.$$

PROOF Suppose $d \lessdot^K M$. We will establish $(c \Rightarrow b) \lessdot^K M$ according to the Recursive Specification Lemma. Let $P_1, \cdots, P_n \in \Lambda(K)^o$ such that $(c \Rightarrow b)\vec{P}{\Downarrow}_K$. We consider three cases:

- $n = 0$: Since by premise, $dc{\Downarrow}$, and so, $d{\Downarrow}$ which implies $M{\Downarrow}_K$ by supposition.
- $n = 1$: Then $(c \Rightarrow b)P_1 = b$ and $c \sqsubseteq [\![P_1]\!]$, the latter implies that $c \lessdot^K P$ by reflexivity of $\lessdot^K$ and left-closedness of $\lessdot^K$. Since $\lessdot^K$ is a logical relation, $b \lesssim^K dc \lessdot^K MP_1$. Hence, $(c \Rightarrow b)P_1 (= b){\Downarrow} \Rightarrow MP_1{\Downarrow}_K$.
- $n \geqslant 2$: It should be clear that $(c \Rightarrow b)\vec{P}{\Downarrow} \Rightarrow (c \Rightarrow b)\vec{P} = bP_2 \cdots P_n$ and $c \sqsubseteq P_1$. By the same argument as before $b \lessdot^K MP_1$; then $bP_2 \cdots P_n{\Downarrow} \Rightarrow MP_1 \cdots P_n{\Downarrow}_K$.

Next, we show: $e \lessdot^K N \Rightarrow (c \Rightarrow b)e \lessdot^K MN$. Wlog, suppose $(c \Rightarrow b)e{\Downarrow}$, *i.e.* $(c \Rightarrow b)e = b$ and $c \sqsubseteq e$; the latter implies $c \lessdot^K N$ by left-closedness of $\lessdot^K$. Hence, $dc \lessdot^K MN$, and so, by assumption, $b \lessdot^K MN$. □

REMARK **4.6.4.5** If $c, b \in D_{i-1}$, then observe that $\psi_i(c \Rightarrow b) = (c \Rightarrow b)$; and so $(c \Rightarrow b) \in D_i$.

COROLLARY **4.6.4.6** *The application in $Q_i^K = \langle Q_i^K, \cdot, [\![-]\!]^{Q_i^K} \rangle$ is well-defined, i.e. for $d, e \in Q_i^K, d \cdot e \in Q_i^K$.*

PROOF Let $f \stackrel{\text{def}}{=} \bigsqcup [de]_{i-1}^K (= \bigsqcup [de]_i^K)$, then $f \in D_{i-1} \subseteq D_i$. To show $d \cdot e \in Q_i^K$, it suffices to show $f = de$. Because $de \in [de]_{i-1}^K$, we clearly have $de \sqsubseteq \bigsqcup [de]_{i-1}^K = f$. It remains to show $f \sqsubseteq de$. Now $f \lesssim^K \psi_{i-1}(de) = de_{i-1}$, by Lemma 4.6.4.1(i). Wlog, assume $de_{i-1}{\Downarrow}$; otherwise $f = \bot$ since $[\bot]^K$ is a singleton set. By the Lemma, we have $(e_{i-1} \Rightarrow f) \lesssim^K d$. By Lemma 4.6.4.1 and the above Remark,

$$(e_{i-1} \Rightarrow f) \sqsubseteq \bigsqcup [(e_{i-1} \Rightarrow f)]_i^K \sqsubseteq \bigsqcup [d]_i^K = d.$$

Hence, $f = (e_{i-1} \Rightarrow f)e \sqsubseteq de$. □

$Q_i^K = \langle Q_i^K, \cdot, \Uparrow \rangle$ is a q-aswd where $\cdot$ is restricted to Q_i^K.

LEMMA **4.6.4.7** *Let K be reflexive and fully adequate.*

(i) For $d \in D_i$ and $M \in \Lambda(K)_i^o$, then $d \lessdot^K M \Rightarrow Q_i^K(d) = \bigsqcup [d]_i^K \sqsubseteq [\![M]\!]^{Q_i^K}$.

(ii) $\forall M \in \Lambda(K)_i^o . [\![M]\!]^{Q_i^K} \lessdot^K M$.

PROOF (i): Suppose $d \lessdot^K M$, then $d \lesssim^K [\![M]\!]$ by Lemma 4.6.3.24. This implies $Q_i^K(d) \sqsubseteq [\![M]\!]^{Q_i^K}$ by Lemma 4.6.4.1(iii).
(ii): Let $M \in \Lambda(K)_i^o$. Now $[\![M]\!]^{Q_i^K} \simeq^K [\![M]\!]_i = [\![M]\!] \lessdot^K M$, because $\lessdot^K$ is reflexive. Hence, $[\![M]\!]^{Q_i^K} \lessdot^K M$. □

PROPOSITION **4.6.4.8** *Let K be reflexive and fully adequate. Then, $[\![-]\!]^{Q_i^K}$ preserve application, i.e. $\forall M, N \in \Lambda(K)_i^o . [\![MN]\!]^{Q_i^K} = [\![M]\!]^{Q_i^K} \cdot [\![N]\!]^{Q_i^K}$.*

PROOF Let $M, N \in \Lambda(K)_i^o$. Then,

$$\begin{aligned} [\![MN]\!]^{Q_i^K} &= Q_i^K([\![M^i N^i]\!]_i) \\ &\sqsubseteq Q_i^K([\![M]\!]_i \cdot [\![N]\!]_i) \\ &\sqsubseteq Q_i^K([\![M]\!]^{Q_i^K} \cdot [\![N]\!]^{Q_i^K}). \end{aligned}$$

Since, $[\![M]\!]^{Q_i^K}, [\![N]\!]^{Q_i^K} \in Q_i^K$ and that application in Q_i^K is well defined, $[\![M]\!]^{Q_i^K} \cdot [\![N]\!]^{Q_i^K} \in Q_i^K$. Hence, $[\![MN]\!]^{Q_i^K} \sqsubseteq Q_i^K([\![M]\!]^{Q_i^K} [\![N]\!]^{Q_i^K}) = [\![M]\!]^{Q_i^K} \cdot [\![N]\!]^{Q_i^K}$.

To prove the other inequality, observe that since $\lessdot^K$ is a logical relation (this property is crucial at this point) and that by Lemma 4.6.4.7(ii), $[\![M]\!]^{Q_i^K} \cdot [\![N]\!]^{Q_i^K} \lessdot^K MN$. Then, by Lemma 4.6.4.7(i),

$$[\![M]\!]^{Q_i^K} [\![N]\!]^{Q_i^K} = Q_i([\![M]\!]^{Q_i^K} [\![N]\!]^{Q_i^K}) \sqsubseteq [\![MN]\!]^{Q_i^K}.$$

□

4.6.5 Proof of Full Abstraction

PROPOSITION **4.6.5.1 (Approximant Full Abstraction)**
Let K be reflexive, fully adequate and that projection maps are definable in it. Then, $\forall i \in \omega . K_i$ is fully abstract with respect to Q_i^K, i.e.

$$\forall M, N \in \Lambda(K)_i^o . M \trianglelefteq_i^K N \iff [\![M]\!]^{Q_i^K} \sqsubseteq [\![N]\!]^{Q_i^K}.$$

PROOF "⇒": Let $M \trianglelefteq_i^K N$. By Lemma 4.6.2.5(ii), we have $M \trianglelefteq^K N$. Hence, by Proposition 4.6.3.25, $[\![M]\!] \lesssim^K [\![N]\!]$. Then, by Lemma 4.6.4.1(iii),

$$[\![M]\!]^{Q_i^K} = \bigsqcup [[\![M]\!]]_i^K \sqsubseteq \bigsqcup [[\![N]\!]]_i^K = [\![N]\!]^{Q_i^K}.$$

Thus, indeed $[\![M]\!]^{Q_i^K} \sqsubseteq [\![N]\!]^{Q_i^K}$.
"$\Leftarrow$": Conversely, suppose $[\![M]\!]^{Q_i^K} \sqsubseteq [\![N]\!]^{Q_i^K}$. Then

$$[\![M]\!] = ([\![M]\!])_i \simeq^K [\![M]\!]^{Q_i^K} \sqsubseteq [\![N]\!]^{Q_i^K} \simeq^K ([\![N]\!])_i = [\![N]\!];$$

i.e. $[\![M]\!] \lesssim^K [\![N]\!]$. By Proposition 4.6.3.25, $M \sqsubseteq^K N$ and so, by Lemma 4.6.2.5(i), $M^i \sqsubseteq_i^K N^i$. Since $M \sim^{K_i} M^i$ and $N \sim^{K_i} N^i$ by Remark 4.6.2.6, $M \sqsubseteq_i^K N$. □

LEMMA **4.6.5.2** *(i) For $i \leqslant j$, $Q_i^K \overset{Q_j^K}{\hookrightarrow} Q_j^K \overset{Q_i^K}{\twoheadrightarrow} Q_i^K$ is an embedding. Hence, we have $Q_0^K \trianglelefteq Q_1^K \trianglelefteq \cdots Q_i^K \trianglelefteq \cdots \trianglelefteq Q_j^K \cdots$.*

(ii) $Q^K = \bigsqcup_{i\in\omega} Q_i^K$.

PROOF

(i) Let $d \in Q_i^K \subseteq D_i \subseteq D_j$. Then,

$$\begin{aligned} Q_i^K \circ Q_j^K(d) &= Q_i^K(\bigsqcup [d]_j^K) \\ &= \bigsqcup [\bigsqcup [d]_j^K]_i^K \quad \because \bigsqcup [d]_j^K \in [d]_j^K \\ &= \bigsqcup [d_j]_i^K \\ &= \bigsqcup [d]_i^K \\ &= Q_i^K(d) = d. \end{aligned}$$

Let $e \in Q_j^K$. Then, $[e]_i^K \subseteq [e]_j^K \Rightarrow \bigsqcup [e]_i^K \sqsubseteq \bigsqcup [e]_j^K \Rightarrow \bigsqcup [e]_i^K \lesssim \bigsqcup [e]_j^K$. Hence, by Lemma 4.6.4.1(iii), $Q_i^K \circ Q_i^K(e) = \bigsqcup [\bigsqcup [e]_i^K]_j^K \sqsubseteq \bigsqcup [\bigsqcup [e]_j^K]_j^K = e$.

(ii) For $d \in D_i$,

$$Q^K(d) = Q^K(\bigsqcup_{i\in\omega} d_i) = \bigsqcup_{i\in\omega} Q^K(d_i) = \bigsqcup_{i\in\omega} \bigsqcup_{\sqsubseteq} [d_i]^K.$$

Observe that $\bigsqcup_{\sqsubseteq} [d_i]^K = \bigsqcup_{\sqsubseteq} [d]_i^K$. Hence, $Q^K(d) = \bigsqcup_{i\in\omega} Q_i^K(d)$.

LEMMA **4.6.5.3** *The application in $\langle Q^K, \cdot \rangle$ is well-defined.*

PROOF Let $d, e \in Q^K$. Then,

$$\begin{aligned} d \cdot e &= Q^K(d) \cdot Q^K(e) \\ &= \bigsqcup_{i\in\omega} Q_i^K(d) \cdot \bigsqcup_{i\in\omega} Q_i^K(e) \\ &= \bigsqcup_{i\in\omega}(Q_i^K(d) \cdot Q_i^K(e)) \\ &= \bigsqcup_{i\in\omega} Q_i^K(d \cdot e) \\ &= Q^K(d \cdot e). \end{aligned}$$

□

Let $M \in \Lambda(K)^o$. Then

$$\begin{aligned} [\![M]\!]^Q &= Q^K([\![M]\!]) \\ &= (\bigsqcup_i Q_i^K)(\bigsqcup_i [\![M]\!]_i) \\ &= \bigsqcup_i Q_i^K([\![M]\!]_i) \\ &= \bigsqcup_i Q_i^K([\![M^i]\!]_i) \\ &= \bigsqcup_i [\![M^i]\!]^{Q_i^K}. \end{aligned}$$

Also, $[\![M]\!] = \bigsqcup [\![M^i]\!]_i \sqsubseteq \bigsqcup [\![M^i]\!]^{Q_i^K} = [\![M]\!]^Q$. Hence, $\forall M \in \Lambda(K)^o.[\![M]\!] \sqsubseteq [\![M]\!]^Q$.

PROPOSITION **4.6.5.4** $[\![-]\!]^Q$ *preserves application, i.e.*

$$\forall M, N \in \Lambda(K)^o.[\![MN]\!]^Q = [\![M]\!]^Q [\![N]\!]^Q.$$

PROOF

$$\begin{aligned} [\![MN]\!]^Q &= Q^K([\![MN]\!]) \\ &= (\bigsqcup_i Q_i^K)(\bigsqcup_i [\![M^i N^i]\!]_i) \\ &= \bigsqcup_i Q_i^K([\![M^i N^i]\!]_i) \\ &= \bigsqcup_i [\![M^i N^i]\!]^{Q_i^K} && [\![-]\!]^{Q_i^K} \text{ preserves application} \\ &= \bigsqcup_i [\![M^i]\!]^{Q_i^K} [\![N^i]\!]^{Q_i^K} \\ &= (\bigsqcup_i [\![M^i]\!]^{Q_i^K})(\bigsqcup_i [\![N^i]\!]^{Q_i^K}) \\ &= [\![M]\!]^Q [\![N]\!]^Q. \end{aligned}$$

□

We are now able to prove the main Theorem (4.6.1.1) of this Section and solve the full abstraction problem which we posed in the beginning of the Section for a class of lts.

PROOF OF THEOREM 4.6.1.1
"$\Rightarrow$": Suppose $M \precsim^{\mathcal{K}} N$. Then, by Lemma 4.6.2.5, $\forall i \in \omega. M^i \precsim_i^{\mathcal{K}} N^i$. Full abstraction of $\mathcal{K}_i$ w.r.t. $Q_i^{\mathcal{K}}$ implies $\forall i \in \omega. [\![M^i]\!]^{Q_i^{\mathcal{K}}} \sqsubseteq [\![N^i]\!]^{Q_i^{\mathcal{K}}}$. Hence,

$$[\![M]\!]^{Q} = \bigsqcup_{i\in\omega} [\![M^i]\!]^{Q_i^{\mathcal{K}}} \sqsubseteq \bigsqcup_{i\in\omega} [\![N^i]\!]^{Q_i^{\mathcal{K}}} = [\![N]\!]^{Q}.$$

"$\Leftarrow$": Suppose $[\![M]\!]^{Q} \sqsubseteq [\![N]\!]^{Q}$. Then $[\![M]\!] \sqsubseteq [\![M]\!]^{Q} \sqsubseteq [\![N]\!]^{Q} = Q^{\mathcal{K}}([\![N]\!]) \simeq^{\mathcal{K}} [\![N]\!]$. Thus, $[\![M]\!] \lesssim^{\mathcal{K}} [\![N]\!]$, and so, by Proposition 4.6.3.25, we have $M \precsim^{\mathcal{K}} N$. □

COROLLARY **4.6.5.5** *Since $\lambda\ell_{\mathsf{C}}$ and $\lambda\ell_{\omega}$ are reflexive, fully adequate and in which projection maps are definable, fully abstract models exist and can be constructed for them.* □

COROLLARY **4.6.5.6 (Full Abstraction Conjecture)**
If Open Question 4.6.2.13(i) could be resolved in the affirmative, then $\lambda\ell$ is fully abstract with respect to $Q^{\lambda\ell_{\omega}}$. □

OPEN QUESTION **4.6.5.7** Can our approach yield a proof for the full abstraction of $\lambda\ell_{\mathsf{P}}$ with respect to D (a result due to Abramsky in [Abr87, Chapter 6]) by showing that the fully abstract submodel Q obtained from D by quotienting it out with the appropriate operational equivalence is isomorphic to D?

4.6.6 Relationships between Lazy Operational Preorders

In this last subsection, we investigate the relationships between the various operational preorders introduced in this thesis, namely, $\precsim_B$, $\precsim_w$, and $\precsim_L$ which are introduced in Chapter 2, $\lesssim$, the *PSE lazy ordering*, introduced in Chapter 3 and $\precsim^B$, the *Abramsky bisimulation ordering.*

First, note an important example.

EXAMPLE **4.6.6.1** Let $M \equiv x(\lambda x.\Omega)$ and $N \equiv xx$. Then, $M \precsim^B N$. To see this, observe that if any closed substitution σ maps x to a strongly unsolvable term, then $M_\sigma \sim^B N_\sigma$. If not, say $\sigma(x) =_\beta \lambda x.P$, then, since $\precsim^B$ is a precongruence, we have $M_\sigma \equiv C[\Omega] \precsim^B C[P] \equiv N_\sigma$ where $C[\,] \stackrel{\text{def}}{=} \lambda x.P(\lambda x.[\,])$.

PROPOSITION **4.6.6.2** $\precsim_L \subset \precsim^B$.

PROOF Take M and N as in the previous Example. It is easy to see that $M \not\simeq_L N$. Hence, $\precsim^B \not\subseteq \precsim_L$. Suppose $M \precsim_L N$. It is not difficult to see that

$$\exists C[\,] \in \Lambda.M =_\beta C[\lambda x_1 \cdots x_{n_1}.O_1, \cdots, \lambda x_1 \cdots x_{n_m}.O_m] \&$$

$N =_\beta C[O'_1, \cdots, O'_m]$ satisfying

(1) $\forall 1 \leqslant i \leqslant m.O_i \in \mathbf{PO}_0$ and

(2) $n_i > 0 \Rightarrow O'_i \in \mathbf{PO}_\infty$.

Then, observe that $\forall 1 \leqslant i \leqslant m.\lambda x_1 \cdots x_{n_i}.O_i \precsim^B O'_i$ since $\lambda\ell$ is an fully lazy λ-theory, and that $\mathbf{PO}_0$ and $\mathbf{PO}_\infty$ are the least and the greatest elements with respect to $\precsim^B$ respectively. Then, by appealing to the fact that $\precsim^B$ is a precongruence, we have $M \precsim^B N$. □

REMARK **4.6.6.3** It is instructive to see why $\precsim^B \not\subseteq \precsim_L$.

- $\mathbf{PO}_\infty$-elements are the "top" elements with respect to the preorder $\precsim^B$; whereas they are just a maximal element with respect to $\precsim_L$.
- Variables, either by themselves, or as subterms of a term are given meaning with respect to the $\precsim^B$-ordering by considering their respective closures, *i.e.* universal quantification over all closed terms. In this way *uniformity* of interpretation is forced upon free variables in a way which is incompatible with $\precsim_L$, as is illustrated by the previous Example.

PROPOSITION **4.6.6.4** $\lesssim \subset \precsim^B$.

PROOF Let M and N be defined according to the previous Example. It is easy to see that $M \not\lesssim N$. Hence, $\precsim^B \not\subseteq \lesssim$. Let D_A be the free lazy PSE-model generated from A (see Chapter 3). Let $M \lesssim N$, we show that $\forall C[\,] \in \Lambda^o.C[M]\Downarrow \Rightarrow C[N]\Downarrow$ which is so if and only if $M \precsim^B N$ by Proposition 4.1.3.5. By the Soundness Theorem, we have $\forall \rho.[\![M]\!]_\rho \subseteq [\![N]\!]_\rho$. Let $C[\,] \in \Lambda^o$. Then $\forall \rho.[\![C[M]]\!]_\rho \subseteq [\![C[N]]\!]_\rho$, because application and abstractions are monotonic operations. This means that

$$\forall \rho.A \subseteq [\![C[M]]\!]_\rho \Rightarrow A \subseteq [\![C[N]]\!]_\rho;$$

in particular, this is true for ρ ranging over those that map free variables of M and N to λ-definable substitutions. We therefore have

$$\forall \sigma : \mathsf{Var} \to \Lambda^o.A \subseteq [\![C[M_\sigma]]\!] \Rightarrow A \subseteq [\![C[N_\sigma]]\!].$$

But note that in D_A, $\forall P \in \Lambda^o.P\Downarrow \iff A \subseteq [\![P]\!]$. Whence, $C[M_\sigma]\Downarrow \Rightarrow C[N_\sigma]\Downarrow$, and this is true for all closed substitutions σ. We therefore conclude that $M \precsim^B N$. □

We summarize the results in this subsection in the following theorem:

THEOREM **4.6.6.5**

$$
\begin{array}{ccccccc}
 & & \cong_B & & & & \\
 & & \cup & & & & \\
\cong_L & \subset & \lesssim & \subset & \cong^B & = & \cong^C \\
 & & \cap & & & & \\
 & & \cong_\omega & & & &
\end{array}
$$

□

Chapter 5

Category-Theoretic Characterization of the Lazy Lambda Calculus

Synopsis of the Chapter[1]

This Chapter divides naturally into three sections. Section 1 introduces the various category-theoretic approaches to partiality, surveying notions like Rosolini's *p-category*, diPaola and Hellers' *dominical category* and various concreteness criteria. These lead up to the notions of *partial Cartesian closed category* pCCC and *partial Cartesian closed dominical category* pCCDC which may be regarded as *partial* counterparts of Cartesian closed category. Two main applications of partial categories then follow. A formal proof system, called λ_L-calculus, consisting of a logic for partial functions and inference rules pertaining to constructors and operations in the lazy regime is the subject matter of section 2. λ_L is shown to be *correct* with respect to $\lambda\ell$, *Abramsky lazy λ-theory* (see Chapter 4). λ_L augmented with *convergence testing* has a *sound* and *complete* interpretation in categories of partial morphisms. The second application of partial categories, contained in section 3, is a category-theoretic semantics of *lazy λC-models*, which are lazy λ-models (see Chapter 3) in which *convergence testing* is definable. We prove that *lazy reflexive objects* in a pCCDC give rise to lazy λC-models.

[1]This Chapter would not have been written without the inspiration and helpful guidance provided so cheerfully by Eugenio Moggi. The generosity of his friendship is very much appreciated.

5.1 Categories of Partial Morphisms

This section introduces various category-theoretic frameworks for partiality which take expression in *partial morphisms*. Analogy with well-known concepts in set theory will be used as motivation.

The references are [Mog88a,RR88,Ros86]; see also [CO88,AL86]. In particular, [RR88] is a delightfully concise and comprehensive survey of the various category-theoretic approaches to partiality on which much of the material of this introductory section is based. Chapters 2, 3 and 4 of Rosolini's PhD thesis are a careful and detailed exposition of many of the concepts that will be introduced in this section.

5.1.1 Domain Structures and Categories with Domains

Let A be a category. Let $a, b, c, \cdots$ range over objects of the category (denoted $a, b, c \in \mathrm{Obj}(\mathsf{A})$) and that $f, g, h, \cdots$ range over morphisms. We call the collection of morphisms form a to b the *hom-set* from a to b and denote it as $\mathsf{A}(a, b)$.

As is well-known in the set-theoretic framework, any subset X of a set A can be characterized by an injective map into A — just take an injective map i into A whose image is X. Clearly, this representation is not unique, for any map j isomorphic to i will do just as well. We therefore define a *subobject* of an object $a \in \mathrm{Obj}(\mathsf{A})$ as an *equivalence class* of monos $i : d \hookrightarrow a$ into a; a morphism $i' : d' \hookrightarrow a$ is equivalent to i if there is an isomorphism $h : d \to d'$ such that $i = i' \circ h$. It follows that subobjects can only be determined up to isomorphism.

We review some basic category-theoretic notions and fix notation in the process.

NOTATION **5.1.1.1**

- A mono f from a to b is denoted as $f : a \hookrightarrow b$.
- $[i : d \hookrightarrow a]$ is the *subobject of a corresponding to i*, i.e. the *equivalence class* of monos isomorphic to i. Denote $\mathrm{SubObj}(a)$ as the class of subobjects of a.
- Suppose $f : a \to b$ and that $[i : d \hookrightarrow b]$ is a subobject of b. Then $f^{-1}([i])$, the *inverse image of* $[i]$ *along* f is the subobject $[i']$ of a such that

$$\begin{array}{ccc} d' & \xrightarrow{f' = i^{-1}(f)} & d \\ {\scriptstyle i' = f^{-1}(i)}\downarrow & & \downarrow{\scriptstyle i} \\ a & \xrightarrow[f]{} & b \end{array}$$

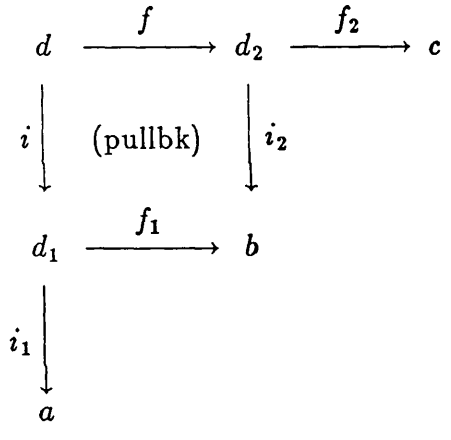
$a \overset{i'}{\hookleftarrow} d' \overset{f'}{\rightarrow} d$ is a pullback of $a \overset{f}{\rightarrow} b \overset{i}{\hookleftarrow} d$ for some f'. We shall use the fact that *the inverse image of a mono is a mono.*

In set-theoretic terms, a partial map f from A to B is essentially a *total* map from X — the domain of definition of f which is a subset of A — to B. We capture this perspective of partial maps in category-theoretic terms as follows:

DEFINITION **5.1.1.2** A *witness* for a *partial morphism* from a to b is a pair $[i, f]$ such that $i : d \hookrightarrow a$ and $f : d \rightarrow b$. We say that two witnesses $[i, f]$ and $[i', f']$ are *equivalent* if there exists an isomorphism h such that $i' = i \circ h$ and $f' = f \circ h$. A *partial morphism* from a to b is the equivalence class of isomorphic witnesses for a partial morphism from a to b.

We present some basic notions pertaining to partial morphisms.

- A partial morphism g from a to b is denoted as $g : a \rightharpoonup b$.
- The *definedness* partial order $\leqslant$ on the class of partial morphisms from a to b is defined as follows: $[i, f] \leqslant [i', f'] \overset{\text{def}}{=} \exists j. i = i' \circ j \;\&\; f = f' \circ j$.
- The *domain* of a partial morphism $[i, f]$ is the equivalence class $[i]$.
- The *composition of partial morphisms* $[i_1, f_1] : a \rightharpoonup b$ and $[i_2, f_2] : b \rightharpoonup c$ is the partial morphism $[i_1 \circ i, f_2 \circ f] : a \rightharpoonup c$



where $d_1 \overset{i}{\hookleftarrow} d \overset{f}{\rightarrow} d_2$ is a pullback of $d_1 \overset{f_1}{\rightarrow} b \overset{i_2}{\hookleftarrow} d_2$. Composition is therefore well-defined iff such a pullback exists.

Clearly, given a category A, the objects of A together with partial morphisms do not in general form a category because composition as prescribed above may well be undefined. Fortunately, this may not be entirely undesirable after all since not all the definable partial morphisms are the *admissible* or meaningful

ones. This is evident, for example, in the category of topological spaces. In this case, monos are not even in general subspace inclusions and that one is typically interested in continuous maps defined on an *open* subspace. In the case of partial continuous functions on chain-complete posets, the partial morphisms of interest are those functions defined on *upward-closed* subsets. What is required is therefore a way of specifying the admissible partial morphisms. This is achieved by imposing some constraints on the class of *admissible* subobjects. For these reasons, categories of partial morphisms will be defined by specifying a category A and a *domain structure* $\mathcal{M}$ consisting in a collection of *admissible subobjects* and sufficient (and necessary, as it turns out) closure properties to qualify the collection of subobjects and partial morphisms as a category.

DEFINITION **5.1.1.3** $\mathcal{M} \stackrel{\text{def}}{=} \{ \mathcal{M}(a) : a \in \mathsf{Obj}(\mathsf{C}) \}$ is a *domain structure*[2] on A if

(1) $\forall a \in \mathsf{Obj}(\mathsf{A}).\mathcal{M}(a) \subseteq \mathsf{SubObj}(a)$,

(2) $\forall a \in \mathsf{Obj}(\mathsf{A}).[\mathsf{id}_a] \in \mathcal{M}(a)$,

(3) $[i' : c \hookrightarrow b] \in \mathcal{M}(b), [i : b \hookrightarrow a] \in \mathcal{M}(a) \Rightarrow [i \circ i' : c \hookrightarrow a] \in \mathcal{M}(a)$,

(4) $f : a \to b, m \in \mathcal{M}(b) \Rightarrow f^{-1}(m)$ exists, and $\in \mathcal{M}(a)$.

$\mathcal{M}$-Ptl(A) is the *category of partial morphisms in* A *with domains in* $\mathcal{M}$.

REMARK **5.1.1.4** Conditions (3) and (4) together ensure that the composition of partial morphisms is well-defined. (2) stipulates that the identities $[\mathsf{id}_a, \mathsf{id}_a]$ are in $\mathcal{M}$-Ptl(A). It is easy to check that the properties of a domain structure are (necessary and) sufficient to ensure that $\mathcal{M}$-Ptl(A) is a category.

There is a *canonical embedding* of A into $\mathcal{M}$-Ptl(A). Define $F : \mathsf{A} \to \mathcal{M}$-Ptl($\mathsf{A}$) to be the identity on objects and that it takes a map $f : a \to b$ to $[\mathsf{id}_a, f] : a \rightharpoonup b$. We call a map in the image of F *total*; and confuse A with the subcategory of $\mathcal{M}$-Ptl(A) consisting of total maps. Given any category A, there is a trivial domain structure, namely $\mathcal{M}(a) = \{ [\mathsf{id}_a] \}$. Clearly, $\mathcal{M}$-Ptl(A) is equivalent to A.

5.1.2 P-Categories

Let $\mathcal{M}$-Ptl(A) be a category of partial morphisms with domains in $\mathcal{M}$. Suppose A is endowed with certain structure. How much of this structure is inherited by $\mathcal{M}$-Ptl(A)? We shall seek to answer the question by concentrating on one such structure: the categorical product. This is partly because categorical product is such a ubiquitous feature in categories that characterize many theoretical computer science and recursion-theoretic concepts. More importantly, any category

[2]This notion is thus christened by Eugenio Moggi. Domain in this context is suggestive of the "domain of definition" of admissible partial morphisms

of partial morphisms can be fully embedded in a category of partial morphisms in a category which has categorical products.

Let us first remark the following:

LEMMA **5.1.2.1** *Suppose* A *has categorical products and* $\mathcal{M}$ *is a domain structure. Then the product bifunctor* $(-) \times (-)$ *on* A *can be extended to the whole of* $\mathcal{M}$-Ptl(A).

PROOF Easy. Define $[i, f] \times [j, g] \stackrel{\text{def}}{=} [i \times j, f \times g]$. □

However, $(-) \times (-)$ ceases to be a categorical product in $\mathcal{M}$-Ptl(A). It is instructive to see why this is so. Consider the category of sets and partial functions. Let $f : A \rightharpoonup B$ and $g : A \rightharpoonup C$ be partial maps. Extending the definition of pairing in the obvious way, we have $\langle f, g\rangle : A \rightharpoonup B \times C$ where

$$\langle f, g\rangle(x) \stackrel{\text{def}}{=} \begin{cases} \langle f(x), g(x)\rangle & \text{if both} f(x) \text{ and } g(x) \text{ are defined;} \\ \text{undefined} & \text{else.} \end{cases}$$

It should be clear that $\mathsf{dom}(\langle f, g\rangle) = \mathsf{dom}(f) \cap \mathsf{dom}(g)$. Now, take f to be a total map and g the everywhere undefined map, then $\mathsf{dom}(\langle f, g\rangle) = \emptyset$. Hence, we do *not* have

$$p_{B,C} \circ \langle f, g\rangle = f.$$

Crucially, the reason why the bifunctor $(-) \times (-)$ fails to be the categorical product on $\mathcal{M}$-Ptl(A) (given that A has categorical products) is precisely because the projections are not natural in *both* arguments. To see this, consider a genuinely partial map $f : b \rightharpoonup c$, *i.e.* the domain of f is not isomorphic to id_b. The following square

$$\begin{array}{ccc} a \times b & \xrightarrow{p_{a,b}} & a \\ {\scriptstyle \mathsf{id}_a \times f}\downarrow & & \downarrow{\scriptstyle \mathsf{id}_a} \\ a \times c & \xrightarrow[p_{a,c}]{} & a \end{array}$$

does *not* commute because $p_{a,c} \circ (\mathsf{id}_a \times f)$ is clearly genuinely partial, and hence cannot be equal to $\mathsf{id}_a \circ p_{a,b} = p_{a,b}$. However, it is easy (but tedious) to check that

- the diagonal: $\Delta : (-) \to (-) \times (-)$,
- commutativity isomorphism:

$$\tau_{a,b} \stackrel{\text{def}}{=} (q_{a,b} \times p_{a,b}) \circ \Delta_{a \times b} : a \times b \to b \times a;$$

- associativity isomorphism:

$$\alpha_{a,b,c} \stackrel{\text{def}}{=} ((\mathrm{id}_a \times p_{b,c}) \times (q_{b,c} \circ q_{a,b\times c})) \circ \Delta_{a\times(b\times c)} : a \times (b \times c) \to (a \times b) \times c;$$

are natural in *all* arguments, although the components from which they are constructed are not.

In what follows, we shall give an algebraic or equational description of the situation above.

DEFINITION **5.1.2.2 (Rosolini)** A *p-category* is a category C endowed with

- a bifunctor $\times : (-) \times (-) \to (-)$ which is called *product*,
- a natural transformation $\Delta : (-) \to (-) \times (-)$ which is call *diagonal* and
- two families of natural transformations

$$\{ p_{\_\times b} : (-) \times b \dot{\to} (-), b \in \mathrm{Obj}(\mathbb{C}) \} \qquad \text{and}$$

$$\{ q_{a\times\_} : a \times (-) \dot{\to} (-), a \in \mathrm{Obj}(\mathbb{C}) \}$$

which are called *projections* satisfying the following

- axioms:

$$p_{a,a} \circ \Delta_a = \mathrm{id}_a = q_{a,a} \circ \Delta_a \quad p_{a,b} \times q_{a,b} \circ \Delta_{a\times b} = \mathrm{id}_{a\times b}$$
$$p_{a,b} \circ (\mathrm{id}_a \times p_{b,c}) = p_{a,b\times c} \quad p_{a,c} \circ (\mathrm{id}_a \times q_{b,c}) = p_{a,b\times c}$$
$$q_{a,c} \circ (p_{a,b} \times \mathrm{id}_c) = q_{a\times b,c} \quad q_{b,c} \circ (q_{a,b} \times \mathrm{id}_c) = q_{a\times b,c}.$$

- commutativity and associativity isomorphisms $\tau_{a,b}$ and $\alpha_{a,b,c}$ respectively are natural in all arguments.

NOTATION **5.1.2.3** In contexts where confusion is unlikely to arise, subscripts in morphisms like $p_{a,b}, q_{a,b}, \Delta_a, \alpha_{a,b,c}, \tau_{a,b}$ and id_a will be omitted.

REMARK **5.1.2.4** (i) As remarked earlier, p-categories are designed to be algebraic characterizations of categories of the form $\mathcal{M}$-Ptl(A) such that A has categorical products. It is an easy exercise to check that

a. If A has categorical products, then the extension of the product from A to $\mathcal{M}$-Ptl(A) induces canonically a structure of p-category on $\mathcal{M}$-Ptl(A). In particular, A is itself a p-category.

b. Conversely, if $\mathcal{M}$-Ptl(A) is a p-category, then the bifunctor $(-) \times (-)$ is categorical product on A. To see why this is so (see Corollary 5.1.3.5); and to perceive A as a subcategory of (the p-category) $\mathcal{M}$-Ptl(A) is to anticipate the notion of the subcategory of "total morphisms" of a p-category, a formulation of which will follow.

The above observation may be summarized as:

$$\text{A has categorical products} \iff \mathcal{M}\text{-Ptl(A) is a p-category.}$$

It seems reasonable, therefore, to suggest that the prefix "p" in p-category is mnemonic of the keywords: *partial* morphisms and partially defined *products.*

(ii) A few comments about the various equational axioms in the definition. The first two are clearly necessary if $\times$ is to behave like the categorical product. The other four are somewhat redundant vis-à-vis the commutativity isomorphism. All four are needed to prove that α and τ are isomorphisms; but the fact that α is a natural isomorphism yields one pair of identities from the other.

5.1.3 The Category of Domains of a P-Category

At this juncture, there are pertinent questions that remain as yet unanswered:

(1) Since any category of partial morphisms $\mathcal{M}$-Ptl(A) is a p-category — indeed, p-categories are deliberately algebraic abstractions of $\mathcal{M}$-Ptl(A), given any p-category, how can one enunciate internally the *extent* of partiality of a morphism? More specifically, is there a way to differentiate a "partial" morphism from a "total" one?

(2) Is an arbitrary p-category of the form $\mathcal{M}$-Ptl(A)? If not, can it be fully embedded in one?

The above questions will be answered in what follows.

diPaola and Heller [dH88] introduced the notion of the *domain* of a morphism $f : a \to b$ in a p-category C. Recall that in a category of partial morphisms $\mathcal{M}$-Ptl(A), the "domain of definition" of a partial morphism $[i, f] : a \rightharpoonup b$ is represented by the subobject $[i]$. diPaola and Hellers' approach is to replace the subobject on which a partial morphism is defined by "that sub-function of the identity which is defined precisely on the domain of definition of the partial morphism".

DEFINITION **5.1.3.1 (diPaola & Heller)** Given a morphism $f : a \to b$ in a p-category C, the *domain* of f, $\mathrm{dom}(f) : a \to a$, is the composite map

$p_{a,b} \circ (\mathrm{id}_a \times f) \circ \Delta_a : a \rightharpoonup a.$

REMARK **5.1.3.2** (i) It is easy to check that in the p-category $\mathcal{M}$-Ptl(A), dom($[i, f]$), the domain (according to diPaola and Heller) of a partial morphism, $[i, f] : a \rightharpoonup b$, is $[i, i] : a \rightharpoonup a$. Clearly, such domains are in 1-1 correspondence with the subobjects in $\mathcal{M}$. To call $\mathcal{M}$ a domain structure is therefore consistent with DiPaolaformal and Hellers' notion of domain.

(ii) Given a p-category C, a morphism $f : a \rightharpoonup b$ is *total* if $\mathrm{dom}(f) = \mathrm{id}_a$. Define $\mathbf{C}_t$ to be the subcategory of C consisting of total morphisms. It is easy to see that:

$$\mathbf{A} \cong (\mathcal{M}\text{-Ptl}(\mathbf{A}))_t;$$

that is to say, the notion of totality (or partiality) in the p-category setting coincides with that in $\mathcal{M}$-Ptl(A) — the category of partial morphisms in a "total" category A. In p-categories, partiality is the prior notion and total morphisms are a derived notion; whereas in $\mathcal{M}$-Ptl(A), all maps in A are a *priori* total, and partial morphisms $[i, f]$ are defined in terms of total maps i and f.

The following properties are satisfied by domains.

PROPOSITION **5.1.3.3** *(i)* $\mathrm{dom}(\mathrm{id}) = \mathrm{id}, \mathrm{dom}(p) = \mathrm{id}, \mathrm{dom}(q) = \mathrm{id}, \mathrm{dom}(\Delta) = \mathrm{id}$,

(ii) $p \circ (\mathrm{id} \times f) = p \circ (\mathrm{id} \times \mathrm{dom}(f)), \mathrm{id} \times \mathrm{dom}(f) \circ \Delta = \Delta \circ \mathrm{dom}(f)$,

(iii) $\mathrm{dom}(f \circ g) = \mathrm{dom}(\mathrm{dom}(f) \circ g), \mathrm{dom}(f) \circ g = g \circ \mathrm{dom}(f \circ g)$,

(iv) $f \circ \mathrm{dom}(f) = f$,

(v) $\mathrm{dom}(f \times g) = \mathrm{dom}(f) \times \mathrm{dom}(g)$,

(vi) $\mathrm{dom}(f) \circ \mathrm{dom}(g) = \mathrm{dom}(g) \circ \mathrm{dom}(f) = \mathrm{dom}(\mathrm{dom}(f) \circ \mathrm{dom}(g))$,

(vii) $\mathrm{dom}(f) \circ \mathrm{dom}(f) = \mathrm{dom}(f)$.

PROOF Easy exercise in equational reasoning with the axioms of p-category. □

REMARK **5.1.3.4** (i) With reference to the above Proposition:

- (i) says that $\mathrm{id}, p, q, \Delta$ are total.
- The first equation in (ii) may be interpreted as showing the extent to which p falls short of naturality in the second argument. The second substantiates in category-theoretic terms our earlier informal statement about domains being the "sub-function of the identity defined on the domain of definition."

- (iii) implies that the composition of total maps is total.
- (vi) and (vii) say that composition is a binary operation on the set $\mathrm{Dom}(a)$ of domains on a, which is commutative, associative and idempotent with a unit. (In the category of sets and partial functions, composition behaves exactly as set-intersection in $\mathrm{Dom}(a) = \wp a$.) Hence, it induces a partial order on $\mathrm{Dom}(a)$:

$$\mathrm{dom}(f) \leqslant \mathrm{dom}(g) \iff \mathrm{dom}(f) = \mathrm{dom}(g) \circ \mathrm{dom}(f);$$

which coincides with the definedness partial order in p-categories of the form $\mathcal{M}$-Ptl(A).

(ii) A useful corollary of the Proposition is that

$$f \text{ is a domain } (\textit{i.e. } \exists g.f = \mathrm{dom}(g)) \text{ iff } \mathrm{dom}(f) = f \text{ iff } f \leqslant \mathrm{id}.$$

(iii) Let f and g be morphisms in a p-category and that $p \circ f = p \circ g$ and $q \circ f = q \circ g$. Then, $f = g$. To see this, observe that $(p \times q) \circ \Delta$ is the identity. By naturality of Δ, $f = (p \times q) \circ \Delta \circ f = (p \circ f \times q \circ f) \circ \Delta$; also $(p \circ g \times q \circ g) \circ \Delta = g$. Whence, the assertion follows.

COROLLARY **5.1.3.5** *If* $\mathbb{C}$ *is a p-category, then* $\mathbb{C}_t$ *has binary products.*

PROOF By the Proposition, $\mathbb{C}_t$ is closed under the product functor. Projection maps form $a \times b$ are exactly $p_{a,b}$ and $q_{a,b}$. For any pair of total maps $f : c \to a$ and $g : c \to b$. Let $h \stackrel{\text{def}}{=} (f \times g) \circ \Delta_c : c \to a \times b$. It is total since it is a composition of total maps. Now, $p \circ h = p \circ (f \times g) \circ \Delta = p \circ (f \times \mathrm{id}) \circ (\mathrm{id} \times g) \circ \Delta = f \circ p \circ (\mathrm{id} \times g) \circ \Delta = f \circ \mathrm{dom}(g) = f$, since g is total. Similarly, $q \circ h = g$. Uniqueness follows form Remark 5.1.3.4(iii). □

The discussion hitherto may be seen as providing an answer to question (1) which was posed at the beginning of the section. The answer to question (2) may be stated in the form of a representation theorem:

THEOREM **5.1.3.6 (Rosolini)** *Suppose* $\mathbb{C}$ *is a p-category. Then, there is a full embedding* $E : \mathbb{C} \to \mathcal{D}\text{-}Ptl(\mathbb{D}om_{\mathbb{C}})$ *which preserves the p-category structure.* □

$\mathbb{D}om_{\mathbb{C}}$ is the category of domains of the p-category $\mathbb{C}$ defined as follows:

DEFINITION **5.1.3.7** $\mathbb{D}om_{\mathbb{C}}$ has as objects all domains of $\mathbb{C}$, *i.e.* idempotent morphisms $\mathrm{dom}(f) : a \to a$. A morphism $g : \mathrm{dom}(f) \to \mathrm{dom}(h)$ in $\mathbb{D}om_{\mathbb{C}}$ is defined to be a morphism $g : a \to b$ in $\mathbb{C}$ such that

$$(\dagger) \qquad \mathrm{dom}(h) \circ g \circ \mathrm{dom}(f) = g.$$

The identity of an object $\mathrm{dom}(f)$ is itself and the composition of morphisms f and g is just $f \circ g$.

It is easy to check that $\mathbf{Dom}_{\mathbf{C}}$ is indeed a category. Note that (†) is equivalent to the conjunction of the following equations:

$$\mathsf{dom}(g) = \mathsf{dom}(f) \text{ and } g = \mathsf{dom}(h) \circ g;$$

which formalize the stipulation that g is "defined" on $\mathsf{dom}(h)$ and g "takes values" in $\mathsf{dom}(h)$ respectively. Significantly, $\mathbf{Dom}_{\mathbf{C}}$ has categorical products.

$\mathbf{C}$ also induces a canonical domain structure $\mathcal{D}$ on $\mathbf{Dom}_{\mathbf{C}}$. Before $\mathcal{D}$ is defined, the following preliminary Lemma is in order.

LEMMA **5.1.3.8** *Let* $f = \mathsf{dom}(f), g = \mathsf{dom}(g)$ *and* $h = \mathsf{dom}(h)$ *be domains on a. Then* $f : \mathsf{dom}(g) \to \mathsf{dom}(h)$ *is a map in* $\mathbf{Dom}_{\mathbf{C}}$ *iff*

$$f = g \text{ and } g \leqslant h.$$

In particular, if $f \leqslant h$, *then the map* $\mathsf{dom}(f) : \mathsf{dom}(f) \to \mathsf{dom}(h)$ *is a mono in* $\mathbf{Dom}_{\mathbf{C}}$.

PROOF Sufficiency is straightforward to check. To see the necessity: we have

$$f = \mathsf{dom}(f) = \mathsf{dom}(g) = g.$$

Also, $f = \mathsf{dom}(h) \circ f$ implies that $g = f = h \circ g = h \circ \mathsf{dom}(g)$. Hence, $g \leqslant h$. For the final assertion, suppose $l, l' : \mathsf{dom}(g) \to \mathsf{dom}(f)$ and

$$\mathsf{dom}(f) \circ l = \mathsf{dom}(f) \circ l'.$$

Then, by definition of morphism in $\mathbf{Dom}_{\mathbf{C}}$, since $\mathsf{dom}(h) \circ \mathsf{dom}(f) = \mathsf{dom}(f)$, we have

$$\begin{aligned} \mathsf{dom}(h) \circ (\mathsf{dom}(f) \circ l) \circ \mathsf{dom}(g) &= \mathsf{dom}(f) \circ l \circ \mathsf{dom}(g) \\ &= l. \end{aligned}$$

Similarly, $\mathsf{dom}(h) \circ (\mathsf{dom}(f) \circ l') \circ \mathsf{dom}(g) = l'$; and we are done. □

LEMMA **5.1.3.9** $\mathcal{M}$, *the collection of monos in* $\mathbf{Dom}_{\mathbf{C}}$ *of the form* .

$$\mathsf{dom}(f) : \mathsf{dom}(f) \to \mathsf{dom}(h)$$

with $\mathsf{dom}(f) \leqslant \mathsf{dom}(h)$ *is a domain structure.*

PROOF The class $\mathcal{M}$ is clearly closed under identity and composition. We claim that the pullback of $\mathsf{dom}(f) : \mathsf{dom}(f) \hookrightarrow \mathsf{dom}(h)$ along $l : \mathsf{dom}(g) \to \mathsf{dom}(h)$ is

$$\mathsf{dom}(\mathsf{dom}(f) \circ l) \hookrightarrow \mathsf{dom}(g).$$

To see this, by Proposition 5.1.3.3(iii)a, we have $\mathsf{dom}(f \circ l) = \mathsf{dom}(\mathsf{dom}(f) \circ l)$. By definition of morphism l,

$$\begin{aligned}
& \mathsf{dom}(g) \circ \mathsf{dom}(f \circ l) \\
= \ & \mathsf{dom}(l) \circ \mathsf{dom}(f \circ l) && \text{Proposition 5.1.3.3(iii)b} \\
= \ & \mathsf{dom}(f \circ l) \circ \mathsf{dom}(l \circ \mathsf{dom}(f \circ l)) && \text{Proposition 5.1.3.3(iii)b} \\
= \ & \mathsf{dom}(f \circ l) \circ \mathsf{dom}(\mathsf{dom}(f) \circ l) && \text{Proposition 5.1.3.3(iii)a} \\
= \ & \mathsf{dom}(f \circ l) \circ \mathsf{dom}(f \circ l) \\
= \ & \mathsf{dom}(f \circ l).
\end{aligned}$$

Hence $\mathsf{dom}(f \circ l) \leqslant \mathsf{dom}(g)$.
Consider the following diagram:

$$\begin{array}{ccc}
\mathsf{dom}(f \circ l) & \xrightarrow{\mathsf{dom}(f) \circ l} & \mathsf{dom}(f) \\
{\scriptstyle \mathsf{dom}(f \circ l)}\downarrow & (\dagger) & \downarrow{\scriptstyle \mathsf{dom}(f)} \\
\mathsf{dom}(g) & \xrightarrow[l]{} & \mathsf{dom}(h)
\end{array}$$

$\mathsf{dom}(f) \circ l : \mathsf{dom}(f \circ l) \to \mathsf{dom}(f)$ is a $\mathbb{D}\mathrm{om}_{\mathbb{C}}$ morphism, for

$$\begin{aligned}
\mathsf{dom}(f) \circ (\mathsf{dom}(f) \circ l) \circ \mathsf{dom}(f \circ l) &= (\mathsf{dom}(f) \circ l) \circ \mathsf{dom}(\mathsf{dom}(f) \circ l) \\
&= \mathsf{dom}(f) \circ l.
\end{aligned}$$

The last step is justified by Proposition 5.1.3.3(iv).
The above square (†) is a pullback square:

$$\begin{aligned}
l \circ \mathsf{dom}(f \circ l) &= \mathsf{dom}(f) \circ l && \text{Proposition 5.1.3.3(iii)b} \\
&= \mathsf{dom}(f) \circ (\mathsf{dom}(f) \circ l).
\end{aligned}$$

Universality: Suppose

$$\begin{array}{ccc}
\mathsf{dom}(k) & \xrightarrow{j} & \mathsf{dom}(f) \\
{\scriptstyle i}\downarrow & & \downarrow{\scriptstyle \mathsf{dom}(f)} \\
\mathsf{dom}(g) & \xrightarrow[l]{} & \mathsf{dom}(h)
\end{array}$$

commute, *i.e.* $l \circ i = \mathsf{dom}(f) \circ j = j$. Then, by definition of morphism,

$$\begin{aligned} j &= \mathsf{dom}(f) \circ j \\ &= \mathsf{dom}(f) \circ l \circ i \\ &= (\mathsf{dom}(f) \circ l) \circ i. \end{aligned}$$

Again, by definition of morphism,

$$\begin{aligned} i &= \mathsf{dom}(g) \circ i \\ &= \mathsf{dom}(l) \circ i && \text{Proposition 5.1.3.3(iii)b} \\ &= i \circ \mathsf{dom}(l \circ i) \\ &= i \circ \mathsf{dom}(\mathsf{dom}(f) \circ j) \\ &= i \circ \mathsf{dom}(\mathsf{dom}(f) \circ l \circ i) && \text{Proposition 5.1.3.3(iii)b} \\ &= \mathsf{dom}(\mathsf{dom}(f) \circ l) \circ i \\ &= \mathsf{dom}(f \circ l) \circ i. \end{aligned}$$

Uniqueness then follows from the injectiveness of $\mathsf{dom}(f \circ l)$. □

We are now in a position to prove the representation theorem.

PROOF Let $\mathbf{C}$ be a p-category and $f : a \to b$ a morphism in it. Define $E(f : a \to b)$ to be

$$[\mathsf{dom}(f), f] : \mathsf{id}_a \rightharpoonup \mathsf{id}_b.$$

Clearly $\mathsf{dom}(f) : \mathsf{dom}(f) \hookrightarrow \mathsf{id}_a$ and $f : \mathsf{dom}(f) \to \mathsf{id}_b$ are morphisms in $\mathbf{Dom_C}$. To see that E is a full embedding, consider any morphism:

$$\begin{array}{ccc} \mathsf{dom}(g) & \xrightarrow{\;h\;} & \mathsf{id}_b \\ {\scriptstyle \mathsf{dom}(g)}\big\downarrow & & \\ \mathsf{id}_a & & \end{array}$$

We immediately have $\mathsf{dom}(g) = \mathsf{dom}(h)$. Hence, the partial morphism is witnessed by $[\mathsf{dom}(h), h]$ and $h : a \to b$ is a morphism in $\mathbf{C}$. □

DEFINITION **5.1.3.10** Let $\mathbf{C}$ be a category and I a class of idempotents in $\mathbf{C}$. Then $Split(I)$ is the category whose objects are the elements of $I \cup \{\, \mathsf{id}_a : a \in \mathrm{Obj}(\mathbf{C}) \,\}$ and has morphisms $f : a \to b$ such that f is a morphism in $\mathbf{C}$ satisfying

$$b \circ f \circ a = f.$$

PROPOSITION **5.1.3.11** *The category of partial morphisms $\mathcal{D}$-Ptl($\mathbb{D}om_{\mathbb{C}}$) is precisely Split($\mathbb{D}om_{\mathbb{C}}$), the free completion of $\mathbb{C}$ where all domain idempotents split.*

PROOF A morphism in $\mathcal{D}$-Ptl($\mathbb{D}om_{\mathbb{C}}$) is of the form:

$$[\mathrm{dom}(f), f] : \mathrm{dom}(g) \rightharpoonup domh$$

such that $\mathrm{dom}(f) \leqslant \mathrm{dom}(g)$ and $\mathrm{dom}(h) \circ f = f$; or equivalently, $\mathrm{dom}(h) \circ f \circ \mathrm{dom}(g) = f$ which is precisely the criterion for $f : \mathrm{dom}(g) \rightharpoonup \mathrm{dom}(h)$ to be a morphism in $Split(\mathbb{D}om_{\mathbb{C}})$. □

COROLLARY **5.1.3.12** *The image of the embedding $E : \mathbb{C} \to \mathcal{D}$-Ptl($\mathbb{D}om_{\mathbb{C}}$) is the full subcategory of total objects. Hence, the embedding is an equivalence if and only if all domains split in $\mathbb{C}$.* □

5.1.4 Dominical Categories

Dominical categories were introduced in [dH88] as an abstract category-theoretic presentation for recursion theory. More generally, dominical categories constitute an approach to enunciate partiality in the category-theoretic framework which axiomatically "objectifies" undefinedness or meaninglessness as a family of "zero morphisms". At it turns out, a dominical category does not differ very much from a p-category; in point of fact, a dominical category is a slight specialization of the notion of p-category.

DEFINITION **5.1.4.1** (i) A *pointed category* is a category $\mathbb{C}$ with a family of distinguished morphisms $0_{a,b} : a \to b$ stable under composition where a, b range over Obj($\mathbb{C}$), *i.e.*

$$0_{a,b} \circ f = 0_{c,b} \text{ and } g \circ 0_{a,b} = 0_{a,c}$$

for all morphisms $f : c \to a$ and $g : b \to c$.

(ii) A morphism $f : a \to b$ in the pointed category $\mathbb{C}$ is *weakly total* if

$$\forall h : c \to a.[f \circ h = 0_{c,b} \Rightarrow h = 0_{c,a}].$$

Observe that identity morphisms are weakly total and that composition of weakly total morphisms is weakly total.

(iii) Let $\mathbb{C}'$ stand for the subcategory of $\mathbb{C}$ consisting of weakly total morphisms.

Now, we are in a position to define dominical categories.

DEFINITION **5.1.4.2 (DiPaola & Heller)** A pointed category $\mathbb{C}$ with a bifunctor $(-) \times (-) \to (-)$ is *dominical* if

- $\times$ maps $\mathbb{C}' \times \mathbb{C}'$ into $\mathbb{C}'$, defining a Cartesian product on $\mathbb{C}'$, in such a way that
- the induced commutativity and associativity isomorphisms are natural with respect to morphisms in $\mathbb{C}$.
- Moreover, for all morphisms f, g in $\mathbb{C}$:

(Z) $$f \times g = 0 \iff f = 0 \text{ or } g = 0$$

(Nat) $$p \circ f \times \mathsf{id} = f \circ p \qquad f \times f \circ \Delta = \Delta \circ f.$$

PROPOSITION **5.1.4.3** *A dominical category $\mathbb{C}$ is a p-category. Moreover, the categories $\mathbb{C}_t$ and $\mathbb{C}'$ coincide.*

PROOF The axioms (Nat) ensure that Δ is a natural transformation and that p is natural in its first argument. The naturality of q in the second then follows. Since $\times$ is Cartesian on $\mathbb{C}'$ and that for all $a, b \in \mathsf{Obj}(\mathbb{C})$, $p_{a,b}, q_{a,b}$ and Δ_a are all weakly total, hence the six identities in the definition of p-category hold. Thus, a dominical category is a p-category.

Let $f : a \to b$ be a morphism in $\mathbb{C}$. Suppose $\mathsf{dom}(f) = \mathsf{id}_a$. Then, given $f \circ h = 0$, we have

$$\begin{aligned} h &= \mathsf{dom}(f) \circ h \\ &= p \circ (\mathsf{id}_a \times f) \circ \Delta \circ h \quad \text{(Nat)} \\ &= p \circ (h \times (f \circ h)) \circ \Delta \quad \text{(Z)} \\ &= 0 \end{aligned}$$

Now, suppose $f : a \to b$ is weakly total, then

$$\begin{aligned} \mathsf{dom}(f) &= p_{a,b} \circ (\mathsf{id}_a \times f) \circ \Delta_a \quad \times \text{ is Cartesian on } \mathbb{C}' \\ &= \mathsf{id}_a \circ p_{a,a} \circ \Delta_a \\ &= \mathsf{id}_a. \end{aligned}$$

Hence, f is total. □

It is instructive to examine the two implications in the axiom (Z) separately.

DEFINITION **5.1.4.4** Let $\mathbb{C}$ be a non-empty pointed p-category. We say that $\mathbb{C}$ has *zero morphisms* if

$$f = 0 \text{ or } g = 0 \Rightarrow f \times g = 0.$$

If the reverse is true, i.e.

$$f \times g = 0 \Rightarrow f = 0 \text{ or } g = 0,$$

then, we say that $\mathbf{C}$ satisfies the *cancellation property on morphisms.*

As we alluded to earlier, the introduction of "zero morphisms" as in dominical categories is tantamount to the objectification of undefinedness. This can be seen convincingly when one examines $\mathbb{D}\mathrm{om}_{\mathbf{C}}$, the associated category of domains of a pointed p-category $\mathbf{C}$ that has zero morphisms. Intuitively, one would expect the domains of zero morphisms to be isomorphic (in $\mathbb{D}\mathrm{om}_{\mathbf{C}}$) to the "least defined" object, which may be translated into category-theoretic terms as a *strict initial object*[3], 0. Further, given an object $\mathrm{dom}(f)$ in $\mathbb{D}\mathrm{om}_{\mathbf{C}}$, one would also expect the existence of the "least" subobject of $\mathrm{dom}(f)$, which can be none other than

$$[0 \hookrightarrow \mathrm{dom}(f)].$$

More formally, we prove

LEMMA **5.1.4.5** *Let* $\mathbf{C}$ *be a non-empty pointed p-category* $\mathbf{C}$. *Then*[4],

(i) $\mathbf{C}$ *has zero morphisms,*

(ii) $\mathrm{dom}(f) = 0 \iff f = 0.$

(iii) Furthermore,

(1) $\mathbb{D}\mathrm{om}_{\mathbf{C}}$ *has a strict initial object 0,*

(2) all minimal subobjects $[0 \hookrightarrow \mathrm{dom}(f)]$ *are in* $\mathbb{D}\mathrm{om}_{\mathbf{C}}$.

PROOF

(i) By naturality of p in the first argument, we have

$$p_{a,a} \circ (0_{a,a} \times \mathrm{id}_b) = 0_{a,a} \circ p_{a,b} = 0_{a\times b,a}.$$

Since p is weakly total, we have $0_{a,a} \times \mathrm{id}_b = 0_{a\times b,a\times b}$. For any $f : a \to b$ in $\mathbf{C}$, we have,

$$\begin{aligned} 0_{c,d} \times f &= (0_{d,d} \times \mathrm{id}_b) \circ (0_{c,d} \times f) \quad \times \text{ is a bifunctor} \\ &= 0_{c\times a,d\times b}. \end{aligned}$$

[3]An object 0 is *strict* if it is the codomain of precisely one arrow, *i.e.* id_0

[4]We show that (i) necessarily follows from $\mathbf{C}$ being a pointed p-category, so does (ii). The corresponding Lemma 2.2 in [RR88] asserts that (i) $\iff$ (ii) which is inaccurate.

(ii) "$\Leftarrow$" follows immediately form (i).
"$\Rightarrow$": By Proposition 5.1.3.3(iv), we have $f = f \circ \mathrm{dom}(f) = 0$.

(iii) Let $a \in \mathrm{Obj}(\mathbb{C})$ and consider $0_{a,a} : a \to a$. $0_{a,a}$ is a domain because $0_{a,a} = p_{a,a} \circ (\mathrm{id}_a \times 0_a) \circ \Delta_a = \mathrm{dom}(0_{a,a})$. $\mathbb{D}\mathrm{om}_{\mathbb{C}}$ has a strict initial object 0; for if $f : 0 \to \mathrm{dom}(g)$ is a morphism in $\mathbb{D}\mathrm{om}_{\mathbb{C}}$, then $f = \mathrm{dom}(g) \circ f \circ 0 = 0$. Similarly, if $f : \mathrm{dom}(g) \to 0$, then $f = 0 \circ f \circ \mathrm{dom}(g) = 0$. For any object $\mathrm{dom}(f)$ in $\mathbb{D}\mathrm{om}_{\mathbb{C}}$, say, $\mathrm{dom}(f) : b \to b$ in $\mathbb{C}$, the map $0_{a,b} : a \to b$ gives rise to a map

$$0_{a,b} : 0_{a,a} \to \mathrm{dom}(f)$$

in $\mathbb{D}\mathrm{om}_{\mathbb{C}}$, for $0_{a,b} = \mathrm{dom}(f) \circ 0_{a,b} \circ 0_{a,a}$. For $b = a$, $0_{a,a} \to \mathrm{dom}(f)$ is a mono because $0 \leqslant \mathrm{dom}(f)$. Note that in $\mathbb{D}\mathrm{om}_{\mathbb{C}}$, all zero morphisms $0_{a,b}$ are isomorphic to the strict initial object 0.

□

5.1.5 Concreteness Conditions

DEFINITION **5.1.5.1** A p-category $\mathbb{C}$ has a *1-element object*, 1, if 1 is terminal in $\mathbb{C}_t$. We denote the unique total map from a to 1 as $!_a$.

REMARK **5.1.5.2** It is easy to verify that an object 1 in a p-category $\mathbb{C}$ is a 1-element object iff given any family of *total* maps $f_a : a \to 1$ for all $a \in \mathrm{Obj}(\mathbb{C})$, $p_{a,1}$ and $\mathrm{id}_a \times f_a \circ \Delta_a$ are mutually inverse.

DEFINITION **5.1.5.3** Let $\mathbb{C}$ be a pointed p-category with a 1-element object 1. (We have proved that such a category has zero maps.) We say $\mathbb{C}$ *has enough points* if the functor

$$\mathbb{C}(1, -) : \mathbb{C} \to 1/\mathbb{S}\mathrm{et}$$

is faithful, where the canonical point in the hom-set $\mathbb{C}(1, a)$ is $0_{1,a}$. Note that $1/\mathbb{S}\mathrm{et}$ is the (co-sliced) category of pointed sets.

Since $hom(1, -) : \mathrm{Ptl}(\mathbb{S}\mathrm{et}) \xrightarrow{\sim} 1/\mathbb{S}\mathrm{et}$ is an equivalence between the p-category of sets and partial functions and the category of pointed sets with the p-category structure given by the "smash" product, it follows that a p-category $\mathbb{C}$ with enough points is *concrete* in the sense that there is a faithful forgetful functor

$$U : \mathbb{C} \to \mathrm{Ptl}(\mathbb{S}\mathrm{et}).$$

DEFINITION **5.1.5.4** The 1-element object, 1, of a pointed p-category is *atomic* if it has precisely two domains — the identity and zero morphism, i.e.

$$\forall f : 1 \to a.\mathrm{dom}(f) = \mathrm{id}_1 \text{ or } 0_{1,1}.$$

LEMMA **5.1.5.5** *Suppose* $\mathbb{C}$ *is a pointed p-category with an atomic 1-element object. Then,*

$$\mathbb{C}(1,\_) : \mathbb{C} \to 1/\mathcal{S}et$$

preserves the p-category structure. In particular, it preserves 1-element object and zero morphisms.

PROOF Easy exercise. □

THEOREM **5.1.5.6** *Let* $\mathbb{C}$ *be a pointed p-category with an atomic 1-element object. If* $\mathbb{C}$ *has enough points, then* $\mathbb{C}$ *is dominical*

PROOF First observe that the p-category of sets and partial functions and hence $1/\mathcal{S}et$ are dominical. To show that $\mathbb{C}$ is dominical, we need to show:

(1) $\mathbb{C}$ has cancellation property on morphisms,

(2) all weakly total morphisms are total.

(1): Since $\mathbb{C}(1,-)$ is faithful and it preserves p-category structure, *i.e.* $(-) \times (-)$ and zero morphisms, we have

$$f \times g = 0 \iff \mathbb{C}(1,f) \times \mathbb{C}(1,g) = 0$$

in $1/\mathcal{S}et$, which holds iff either one of the two, say, $\mathbb{C}(1,f) = 0$. Since $\mathbb{C}(1,-)$ is faithful, we have f = 0.
(2): Suppose $f : a \to b$ is not total. We have $\mathrm{dom}(f) \neq \mathrm{id}_a : a \to a$. It follows that

$$\mathbb{C}(1,\mathrm{dom}(f)) \neq \mathrm{id} : \mathbb{C}(1,a) \to \mathbb{C}(1,a).$$

Choose $0_{1,a} \neq g : 1 \to a$ such that $\mathbb{C}(1,\mathrm{dom}(f))g = \mathrm{dom}(f) \circ g \neq g$. Since $\mathbb{C}(1,-)$ preserves domains it follows that $\mathbb{C}(1,\mathrm{dom}(f))$ is a sub-function of the identity; and we have

$$\mathrm{dom}(f) \circ g = 0_{1,a}. \qquad \text{(like mapping to "bottom")}$$

Hence, $\mathrm{dom}(f \circ g) = 0$ *i.e.* $f \circ g = 0$. f is therefore not weakly total. □

5.1.6 Cartesian Closed Categories and its Partial Version

We recapitulate the definition of a Cartesian closed category (CCC).

DEFINITION **5.1.6.1** (i) 1 is a *terminal object* of $\mathbb{C}$ if for any $a \in \mathbb{C}$ there exists a unique morphism $!_a : a \to 1$

(ii) $a \xleftarrow{\pi_1} a \times b \xrightarrow{\pi_2} b$ is a *product* of a and b iff for any $f : c \to a$ and $g : c \to b$ there exists unique $\langle f, g\rangle : c \to a \times b$ such that $f = \pi_1 \circ \langle f, g\rangle$ and $g = \pi_2 \circ \langle f, g\rangle$.

(iii) $(a \to b) \times a \xrightarrow{eval} b$ is a *function space* from a to b iff for any $f : c \times a \to b$ there exists unique $\Lambda_{c,a,b}(f) : c \to (a \to b)$ such that $f = \mathrm{eval}_{a,b} \circ (\Lambda_{c,a,b}(f) \times \mathrm{id}_a)$.

DEFINITION **5.1.6.2** Let $[i_1, f_1] : c \rightharpoonup a$ and $[i_2, f_2] : c \rightharpoonup b$, then their *partial tupling*

$$\langle [i_1, f_1], [i_2, f_2]\rangle \stackrel{\text{def}}{=} [i, \langle f_1 \circ i_1', f_2 \circ i_2'\rangle]$$

where $d_1 \xleftarrow{i_1'} d \xrightarrow{i_2'} d_2$ is a pullback of $d_1 \xhookrightarrow{i_1} c \xhookleftarrow{i_2} d_2$ and $i \stackrel{\text{def}}{=} i_1 \circ i_1' = i_2 \circ i_2'$. The subobject $[i]$ is called the *intersection* of $[i_1]$ and $[i_2]$ and we denote the intersection i as $i_1 \cap i_2$.

DEFINITION **5.1.6.3** (i) $\mathrm{peval}_{a,b} : (a \rightharpoonup b) \times a \rightharpoonup b$ is a *partial function space* from a to b if for any $f : c \times a \rightharpoonup b$ there exists unique $\mathrm{p}\Lambda_{c,a,b}(f) : c \to (a \rightharpoonup b)$ such that

$$f = \mathrm{peval}_{a,b} \circ (\mathrm{p}\Lambda_{c,a,b}(f) \times \mathrm{id}_a).$$

$$\frac{f : c \times a \rightharpoonup b}{\mathrm{p}\Lambda_{c,a,b}(f) : c \to (a \rightharpoonup b)}$$

Equivalently, we have a natural isomorphism

$$\phi_{\_}^{a,b} : \mathcal{M}\text{-Ptl}(\mathsf{A})(\_ \times a, b) \cong \mathsf{A}(\_, (a \rightharpoonup b)).$$

$\mathrm{peval}_{a,b} : (a \rightharpoonup b) \times a \rightharpoonup b$ is the partial morphism $(\phi_{a \rightharpoonup b}^{a,b})^{-1}\mathrm{id}_{(a \rightharpoonup b)}$. Unlike the total case, for $f : c \rightharpoonup (a \rightharpoonup b)$, $\mathrm{p}\Lambda_{c,a,b}(\mathrm{peval}_{a,b} \circ (f \times \mathrm{id}_a))$, a total morphism, *extends* f and is not equal to f in general.

(ii) $b_\perp$ is a *lifting* of b if for any $f : c \rightharpoonup b$ there exists unique $\overline{f} : c \to b_\perp$ which we call f *lifted* such that $f = \mathrm{open}_b \circ \overline{f}$ where $\mathrm{open}_b : b_\perp \rightharpoonup b$.

$$\frac{f : c \rightharpoonup b}{\overline{f} : c \to b_\perp}$$

Equivalently, we have a natural isomorphism

$$\eta^b_{\_}: \mathcal{M}\text{-Ptl}(\mathsf{A})(\_, b) \cong \mathsf{A}(\_, b_\perp);$$

and open_b is just $(\eta^b_{b_\perp})^{-1}\text{id}_{b_\perp}$.

DEFINITION **5.1.6.4 (Moggi)** $\mathcal{M}$-Ptl(A) is a *partial Cartesian closed category* if A has terminal object, products and partial function spaces.

In the category of sets and partial functions, lifting corresponds to adjoining an extra element representing undefinedness to a set. Given that A has function spaces, then lifting and partial function spaces are inter-definable.

PROPOSITION **5.1.6.5** *(i)* $open_b \stackrel{\text{def}}{=} (1 \multimap b) \stackrel{\langle \text{id}_{[1\multimap b]}, !_{[1\multimap b]}\rangle}{\longrightarrow} (1 \multimap b) \times 1 \stackrel{peval_{1,b}}{\multimap} b$ *is a lifting of* b.

(ii) $(a \to b_\perp) \times a \stackrel{eval}{\to} b_\perp \stackrel{open_b}{\to} b$ *is a partial function space from* a *to* b.

□

PROPOSITION **5.1.6.6** *If every object of* A *has a lifting, then the canonical embedding* $E : \mathsf{A} \to \mathcal{M}\text{-Ptl}(\mathsf{A})$ *has a right adjoint which is the lifting functor,*

$$\mathcal{M}\text{-Ptl}(\mathsf{A}) \stackrel{(-)_\perp}{\to} \mathsf{A} \stackrel{E}{\to} \mathcal{M}\text{-Ptl}(\mathsf{A}).$$

Note that the functor E *preserves colimit.*

PROOF The right adjoint $(-)_\perp : \mathcal{M}\text{-Ptl}(\mathsf{A}) \to \mathsf{A}$ maps

$a \mapsto a_\perp$ and $f \mapsto \overline{f \circ \text{open}_a}$.

□

We contrast this with the standard category-theoretic characterization of lifting in domain-theory, according to which lifting is the *left* adjoint of the following forgetful functor U:

$$\mathbf{CPO}_\perp \stackrel{U}{\to} \mathbf{CPO} \stackrel{(-)_\perp}{\to} \mathbf{CPO}_\perp.$$

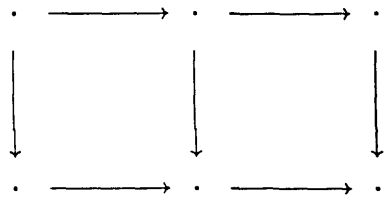
We summarize some useful facts about monos, equalizers and pullback in the following Proposition. The proof in each case is a straightforward exercise.

PROPOSITION **5.1.6.7** *(i) Every equalizer is monic.*

(ii) Every mono in $P(\mathsf{C}, \mathcal{M})$ *is the image of a mono in* C.

(iii) Suppose $f \circ f' = g \circ g'$ *is a pullback. Then, if* g *is monic, so is* f', *the pullback of* g *along* f, *which we sometimes denote as* $f^{-1}(g)$.

(iv) If both small "squares" are pullbacks, so is the outer rectangle.



(v) Suppose the relevant pullbacks exist, then

$$h^{-1}(f \circ g) = h^{-1}(f) \circ (f^{-1}h)^{-1}g.$$

□

5.2 The Formal Lazy Lambda Calculus

In this section, we introduce a language for type structures of partial functions and a corresponding formal system: λ_L which we call the (formal) *lazy λ-calculus*, or λ_L-calculus. λ_L is a variant of λ_P, Moggi and Plotkins' partial λ-calculus [Plo85,Mog86]. In the following, we will compare and contrast λ_L and λ_P and provide an interpretation of λ_L in partial Cartesian closed categories.

We will have occasions to refer to Moggi's paper *op. cit.* whose inspiration and influence on this work is gratefully acknowledged. The reader is further referred to Moggi's Ph.D thesis [Mog88b] for a careful and thorough proof-theoretic investigation of the partial λ-calculus. For an account of the motivation behind such formal systems, see §0.1 of Moggi's thesis and [Plo85].

The formal system consists of two parts:

- a logic for partial functions,
- inference rules pertaining to constructors and operations of the λ-calculus.

5.2.1 Formulation of the Formal System λ_L

We will adhere to Moggi's terminology as in [Mog86] and concern ourselves with the following fragments of the language of partial elements. Let $n \geqslant 0$.

- *S-equations* (Strong-equations)
  $t_1 \simeq t_2$, $t\!\downarrow$;
- *ECS-equations* (Existentially Conditioned S-equations)
  $\{ t_i\!\downarrow : 1 \leqslant i \leqslant n \} \supset q$ where q is an S-equation;

- *QS-equations* (Quasi S-equations)
  $\{ q_i : 1 \leqslant i \leqslant n \} \supset p$ where q_i, p are S-equations.

When the terms of the language are λ-terms, we emphasize the respective equations as S-λ_L, ECS-λ_L and QS-λ_L equations. A formula of the form

$$\{ q_i \mid 1 \leqslant i \leqslant n \} \supset p$$

should be understood as an abbreviation for the more pedantic

$$* \qquad \forall \vec{x} : \vec{\sigma}.(\bigwedge_{1 \leqslant i \leqslant n} q_i) \supset p$$

where $\vec{x}$ is a sequence of variables of types $\vec{\sigma}$ respectively (see later), containing all the free variables in $q_1, \cdots, q_n, p$. We will sometimes use the more elaborate representation of the QS-equations as in $(*)$ (for instance, in the completeness proof later on).

Types and Terms of the language λ_L

Types of the language are defined by the following grammar. Let **At** be a set of atomic types. Let a range over **At** and σ, τ over **Types**.

$$\sigma \in \mathbf{Types} ::= a \mid \vec{\sigma} \rightarrow_! \tau$$

where $\vec{\sigma}$ is a possibly empty sequence of types. Note that $\rightarrow_!$, the lazy function space type constructor, is to be distinguished from $\rightarrow$ and $\rightsquigarrow$ (see [Mog86]), the usual and strict (or partial) function space type constructor respectively.

Terms of the language are defined parametrically over a set of atomic functions **Af**. Let f range over the atomic functions and s, t, u over terms. In particular, we consider application as an atomic function $\mathrm{eval}_{\vec{\sigma},\tau}(-,-)$ which has type $(\vec{\sigma} \rightarrow_! \tau), \vec{\sigma} \rightarrow_! \tau$. As usual, we abbreviate $\mathrm{eval}_{\vec{\sigma},\tau}(s, \vec{t})$ simply as $s(\vec{t})$. The set of terms are defined by the following grammar:

$$t \in \mathbf{Terms} ::= x \mid f(\vec{t}) \mid s(\vec{t}) \mid (\lambda \vec{x} : \sigma.t)$$

where, as usual, $\vec{t}, \vec{x}$ are possibly empty sequences of terms and variables respectively.

A *type assignment* is a map from Var, the set of variables, to **Types** with finite domain. A type assignment B with $\mathrm{dom}(B)$ finite may be represented as a finite set of statements of the form $x : \sigma$. The *typing rules* for terms are as follows:

$$\text{(Log)} \qquad \frac{x : \sigma \in B}{B.x : \sigma}$$

(Seq) $$\frac{B.t_1 : \sigma_1 \quad \cdots \quad B.t_n : \sigma_n}{B.\vec{t} : \vec{\sigma}}$$

($\rightarrow_!$E1) $$\frac{B.\vec{t} : \vec{\sigma} \qquad f : \vec{\sigma} \rightarrow_! \tau}{B.f(\vec{t}) : \tau}$$

($\rightarrow_!$E2) $$\frac{B.\vec{t} : \vec{\sigma} \qquad B.s : \vec{\sigma} \rightarrow_! \tau}{B.s(\vec{t}) : \tau}$$

($\rightarrow_!$I) $$\frac{B, \vec{x} : \vec{\sigma}.t : \tau}{B.(\lambda \vec{x} : \vec{\sigma}.t) : \vec{\sigma} \rightarrow_! \tau}$$

Let s be a term of the language. We say that s is well-typed with respect to B and has type σ or simply $B.s : \sigma$ is well-typed if $B.s : \sigma$ is provable.

DEFINITION **5.2.1.1** (λ_L, **The Formal Typed Lazy λ-Calculus**)
Let $B \equiv \vec{x} : \vec{\sigma}$ be a type-assignment. A statement of the form, $B.\Gamma \supset q$, means that the terms of the respective strong equations in Γ and q are well-typed with respect to B.
Logical Rules:

(Log.1) $$\frac{q \in \Gamma}{B.\Gamma \supset q}$$

(Log.2) $$\frac{\forall p \in \Delta . B.\Gamma \supset p \qquad B.\Delta \supset q}{B.\Gamma \supset q}$$

(Subst) $$\frac{B, \vec{y} : \vec{\tau}.\Gamma \supset q}{C.\Gamma[\vec{y} := \vec{u}] \supset q[\vec{y} := \vec{u}]} \qquad \text{where } B \subseteq C.\vec{u} : \vec{\tau}$$

Existential Rules:

(Ex) $$\frac{B.\Gamma \supset (s \simeq t) \qquad B.\Gamma \supset s\downarrow}{B.\Gamma \supset t\downarrow}$$

(LStr) $$\frac{B.\Gamma \supset s(\vec{t})\downarrow}{B.\Gamma \supset s\downarrow}$$

(Ex.λ) $$\frac{}{B.\Gamma \supset (\lambda \vec{y} : \vec{\tau}.t)\downarrow}$$

Equivalence Rules:

(Eq.1) $$\frac{}{B.\Gamma \supset (t \simeq t)}$$

(Eq.2) $$\frac{B.\Gamma \supset (s \simeq t)}{B.\Gamma \supset (t \simeq s)}$$

(Eq.3) $$\frac{B.\Gamma \supset (s \simeq t) \qquad B.\Gamma \supset (t \simeq u)}{B.\Gamma \supset (s \simeq u)}$$

(Eq.4) $$\frac{B.\Gamma, s\downarrow \supset (s \simeq t) \qquad B.\Gamma, t\downarrow \supset (s \simeq t)}{B.\Gamma \supset (s \simeq t)}$$

Congruence Rules:

(Cong.1) $$\frac{B.\Gamma \supset (s_1 \simeq s_2) \qquad \forall 1 \leqslant i \leqslant n \quad B.\Gamma \supset (t_i \simeq t_i')}{B.\Gamma \supset (s_1\vec{t} \simeq s_2\vec{t'})}$$

(Cong.2) $$\frac{\forall 1 \leqslant i \leqslant n \quad B.\Gamma \supset (s_i \simeq t_i)}{B.\Gamma \supset (f(\vec{s}) \simeq f(\vec{t}))} \qquad f : \vec{\sigma} \to_! \tau \text{ atomic function}$$

λ-Rules:

(ξ) $$\frac{B, \vec{y} : \vec{\tau}.\Gamma \supset (s \simeq t)}{B.\Gamma \supset ((\lambda\vec{y} : \vec{\tau}.s) \simeq (\lambda\vec{y} : \vec{\tau}.t))} \qquad y_i \notin \mathsf{FV}(\Gamma)$$

(β) $$\frac{}{B.\Gamma \supset ((\lambda\vec{y} : \vec{\tau}.t)(\vec{y}) \simeq t)}$$

(Cond-η) $$\frac{B \vdash \Gamma \supset t\downarrow}{B.\Gamma \supset ((\lambda\vec{y} : \vec{\tau}.t(\vec{y})) \simeq t)} \qquad y_i \notin \mathsf{FV}(t)$$

We denote provability in λ_L as $\lambda_L \vdash$.

NOTATION **5.2.1.2** For simplicity, we will omit the concomitant type assignment B in our treatment of terms and the various equations in the sequel when no confusion is likely. Also, we will sometimes be economical with typing details by abbreviating $\exists\vec{\sigma}.B \supseteq \vec{x} : \vec{\sigma}$ as $B \supseteq \vec{x}$.

DEFINITION **5.2.1.3 (The Formal Untyped λ_L-Calculus)**
The language and the inference rules for the untyped λ_L-calculus are obtained by removing the type information from the language and inference rules of the typed λ_L-calculus.

5.2.2 Comparison with Moggi and Plotkins' λ_P-Calculus

The reader is referred to Definition 2 (The Typed λ_P-Calculus) in [Mog86] (see also [Plo85]).

(1) A crucial difference between λ_L and λ_P is that in the former, variables range over total *as well as* partial elements; whereas variables range over only total elements in the latter. This difference is enunciated in the *absence* of the following λ_P inference rule in λ_L:

(E.1) $$\frac{}{B.\Gamma \supset x\downarrow} \qquad x \text{ variable}$$

Observe also that the substitution axiom (Subst) in λ_P is a conditional one — only substitution of terms that *exist* is permissible. In contrast, λ_L allows for unrestricted substitution.

(2) The β-axiom in λ_P is consonant with a *call-by-value* operational semantics whereas that in λ_L is consonant with a *call-by-name* operational semantics, or the implementation strategy usually referred to as lazy evaluation in the functional programming community. For example, in λ_P, a redex $(\lambda x.M)N$ is undefined if N is undefined. No such restriction applies in λ_L.

(3) The application operation "$\cdot$" is both left- and *right-strict* in λ_P whereas it is only *left-strict* in λ_L. This is reflected in the absence of the following λ_P rule in λ_L:

(E.2) $$\frac{B.\Gamma \supset (s(\vec{t}))\downarrow}{B.\Gamma \supset t_i\downarrow}$$

(4) In the pure untyped case, the two calculi can best be contrasted by noting their respective (category-theoretic) canonical models. In the following table, U ranges over objects of any category in which the respective constructs are well-defined.

Canonical Models of λ-calculi	
Calculi	Canonical Models
λ	$U \to U \lhd U$
λ_P	$U \rightharpoonup U \lhd U$
λ_L	$U_\perp \rightharpoonup U \lhd U$

5.2.3 Category-Theoretic Interpretation of λ_L

We give an interpretation of the typed λ_L-calculus in any category of partial morphisms $\mathbb{C} = \mathcal{M}\text{-Ptl}(\mathbb{A})$ which has enough finite limits and representation of partial morphisms so that the following definition is meaningful.

NOTATION **5.2.3.1** Let $a_1, \cdots, a_n, b_1, \cdots, b_m$ be objects in $\mathbb{C}$. Denote by $\pi_{a_i}^{\Pi a_j} : \Pi a_j \to a_i$ the canonical projection on the i-th component; and by

$$\Pi_{\Pi b_j}^{\Pi a_j \times \Pi b_j} = \langle \pi_{b_1}^{\Pi a_j \times \Pi b_j}, \cdots, \pi_{b_m}^{\Pi a_j \times \Pi b_j} \rangle : \Pi a_j \times \Pi b_j \to \Pi b_j$$

the canonical *thinning*. Let $B \equiv \vec{x} : \vec{\sigma}$, $D \equiv \vec{y} : \vec{\tau}$ and $C \equiv B \cup D$. For simplicity, we shall adopt the following abbreviation whenever confusion is unlikely to arise.

$$\Pi_B^C \equiv \Pi_B^{B,D} \equiv \Pi_{\Pi[\![\sigma_i]\!]_\perp}^{\Pi[\![\sigma_i]\!]_\perp \times \Pi[\![\tau_i]\!]_\perp}.$$

DEFINITION **5.2.3.2 (Interpretation of Typed λ_L in Categories)**
The category-theoretic denotations of the various syntactic entities in our language are as follows: Let $B \equiv \vec{x} : \vec{\sigma}$ be a type assignment and $\Sigma \equiv \Pi[\![\sigma_i]\!]_\perp$.

Interpretation of Types:

- $[\![a]\!]$ — an object in $\mathbb{C}$,
- $[\![\vec{\sigma}]\!] = (\Pi_{1 \leqslant i \leqslant n} [\![\sigma_i]\!])$,
- $[\![\vec{\sigma} \to_| \tau]\!] = (\Pi_{1 \leqslant i \leqslant n}) [\![\sigma_i]\!]_\perp \rightharpoonup [\![\tau]\!]$.

In general, we have

- For $\sigma \in$ **Types**, $[\![\sigma]\!]$ is an object in $\mathbb{C}$.
- If $B.t : \tau$, then $[\![t]\!](B) : \Sigma \rightharpoonup [\![\tau]\!]$, *i.e.* a partial morphism from Σ to $[\![\tau]\!]$.
- If $B.q$ where q is a S-λ_L-equation, then $[\![q]\!](B)$ is a subobject of Σ.
- If $B.\Gamma \supset q$ is a QS-λ_L-equation, then $[\![\Gamma \supset q]\!](B)$ is a truth value *i.e.* true or false.

Interpretation of Terms: Let $\Sigma \equiv \Pi[\![\sigma_i]\!]_\perp$.

- $[\![x_i]\!](\vec{x} : \vec{\sigma}) = \text{open}_{[\![\sigma_i]\!]} \circ \Pi_i^\Sigma : \Sigma \rightharpoonup [\![\sigma_i]\!]$ where $\Pi_i^\Sigma : \Pi[\![\sigma_i]\!]_\perp \to [\![\sigma_i]\!]_\perp$.
- $[\![\vec{t}]\!](\vec{x} : \vec{\sigma}) = \langle [\![t_i]\!](\vec{x} : \vec{\sigma}) \mid 1 \leqslant i \leqslant n \rangle : \Sigma \rightharpoonup ([\![\vec{\tau}]\!] = \Pi[\![\tau_i]\!])$ where $B.\vec{t} : \vec{\tau}$ and $|\vec{t}| = n$.
- $[\![s(\vec{t})]\!](\vec{x} : \vec{\sigma}) = \text{peval}_{\Pi[\![\tau_i]\!]_\perp, [\![\nu]\!]} \circ \langle [\![s]\!](\vec{x} : \vec{\sigma}), \langle \overline{[\![t_i]\!](\vec{x} : \vec{\sigma})} \mid 1 \leqslant i \leqslant n \rangle \rangle : \Sigma \rightharpoonup [\![\nu]\!]$. where $B.\vec{t}, s : \vec{\tau}, (\vec{\tau} \to_| \nu)$.
- $[\![\lambda \vec{y} : \vec{\tau}.t]\!](\vec{x} : \vec{\sigma}) = \text{p}\Lambda_{\Sigma, \Pi[\![\tau_i]\!]_\perp, [\![\nu]\!]}([\![t]\!](\vec{x}, \vec{y} : \vec{\sigma}, \vec{\tau})) : \Sigma \to (\Pi[\![\tau_i]\!]_\perp \rightharpoonup [\![\nu]\!])$ where $B, \vec{y} : \vec{\tau}.t : \nu$.

- $[\![t\!\downarrow]\!](\vec{x} : \vec{\sigma}) = \mathrm{dom}([\![t]\!](\vec{x} : \vec{\sigma})) \in \mathrm{SubObj}(\Sigma)$.
- $[\![s \simeq t]\!](\vec{x} : \vec{\sigma}) = \mathrm{eq}_{\mathcal{M}\text{-}\mathrm{Ptl}(\mathbf{A})}([\![s]\!](\vec{x} : \vec{\sigma}), [\![t]\!](\vec{x} : \vec{\sigma})) \in \mathrm{SubObj}(\Sigma)$ where $\mathrm{eq}_{\mathbf{C}}(f, g)$ denotes the equalizer of f and g in $\mathbf{C}$.
- $[\![\Gamma \supset q]\!](\vec{x} : \vec{\sigma}) \stackrel{\mathrm{def}}{=} \bigcap_{1 \leqslant i \leqslant l} [\![p_i]\!](\vec{x} : \vec{\sigma}) \leqslant [\![q]\!](\vec{x} : \vec{\sigma})$ where $\Gamma = \{p_1, \cdots, p_l\}$. Note that $[\![\Gamma \supset q]\!](\vec{x} : \vec{\sigma})$ is a *truth* value. In other words, the value is true iff the intersection of $[\![p_i]\!](\vec{x} : \vec{\sigma})$ for all $p_i \in \Gamma$ *factors through* [5] $[\![q]\!](\vec{x} : \vec{\sigma})$.

REMARK **5.2.3.3** (i) Note that a term s such that $\vec{x} : \vec{\sigma}.s : \tau$, *i.e.* a term s of type τ, which has free variables among $\vec{x}$ of types $\vec{\sigma}$ is interpreted as a partial morphism

$$[\![s]\!](\vec{x} : \vec{\sigma}) : \Pi[\![\sigma_i]\!]_\perp \rightharpoonup [\![\tau]\!].$$

The source of the partial morphism is the product of *lifted* objects $[\![\sigma_i]\!]_\perp$. This accords with the requirement in λ_L that free variables range over all *partial* elements, not merely the total ones. Contrast this with the interpretation of λ_P.

(ii) Both the pairing operation $\langle -, - \rangle$ and the composition $- \circ -$ in a category of partial morphisms are *strict* in the sense that if either one of the morphism f and g is genuinely partial, then so are $\langle f, g \rangle$ and $f \circ g$. Now, consider the interpretation of the *non* right-strict application in λ_L. The result of such an application may well be total, even though the operand or the second argument is partial. To reflect this behaviour, the partial morphism corresponding of the denotation of the operand must be *lifted* before composing with the peval morphism.

(iii) Observe that the denotation of the lazy function space $\sigma \to_! \tau$ is $[\![\sigma]\!]_\perp \rightharpoonup [\![\tau]\!]$. In contrast, that of the strict (or partial) function space is $[\![\sigma]\!] \rightharpoonup [\![\tau]\!]$.

Next we provide an interpretation of the untyped λ_L-calculus by way of translation into the typed λ_P-calculus, and so, through the typed λ_P, obtain a category-theoretic interpretation. This approach of interpreting the untyped calculus in the typed calculus is well-known from Scott's work, notably [Sco80b]. As pointed out to me by Eugenio Moggi, the relative interpretation of the untyped λ_L in the typed λ_P provides an insight into why one would *not* expect completeness of the category-theoretic interpretation with respect to pure λ_L (*i.e.* without any constants): because there are not enough terms. Completeness is, however, guaranteed whenever *convergence testing* is definable in the language. This, together with the fully abstract model obtainable for $\lambda\ell_{\mathbf{C}}$ (see Chapter 4), puts the importance of convergence testing and its apparent indispensability on a firm theoretical footing.

[5] Let f, g be monos in $\mathbf{C}$. We say that f *factors through* g, notationally, $f \leqslant g$ iff $\exists h. f = g \circ h$.

Relative Interpretation of Untyped λ_L in Typed λ_P

DEFINITION **5.2.3.4** We assume that the reader is familiar with [Mog86]. Consider a typed λ_P-calculus augmented with the following (atomic) types, constants and axioms:

- types: 1, D;
- constants:

  $$* : 1,$$
  $$\mathsf{Gr} : (D_\perp \rightharpoonup D) \rightharpoonup D,$$
  $$\mathsf{Fun} : D \rightharpoonup (D_\perp \rightharpoonup D),$$

  where $D_\perp$ is a shorthand for $1 \rightharpoonup D$.
- axioms:

  (Terl) $$\vec{x}, y : \vec{\sigma}, 1.\Gamma \supset (y \simeq *),$$

  (Ret) $$\vec{x}, z : \vec{\sigma}, (D_\perp \rightharpoonup D).\Gamma \supset (\mathsf{Fun}(\mathsf{Gr}z) \simeq z),$$

  (Ex.Fun) $$\vec{x}, z : \vec{\sigma}, D.\Gamma \supset \mathsf{Fun}z{\downarrow}.$$

Note that (Terl) and (Ret) enunciate that 1 is the terminal object and that $(D_\perp \rightharpoonup D)$ is a retract of D respectively.

The translation $(-)'$ maps an untyped term M to a typed term M' of typed D such that all free variables in M' (which are precisely those in M) have type $D_\perp$. The translation is defined as follows:

$$\begin{aligned} x' &\stackrel{\text{def}}{=} (x*), \\ (MN)' &\stackrel{\text{def}}{=} \mathsf{seval}_{D_\perp, D}(\mathsf{Fun}M', \lambda y : 1.N'), \\ (\lambda x.M)' &\stackrel{\text{def}}{=} \mathsf{Gr}(\lambda x : D_\perp.M'); \end{aligned}$$

where $\mathsf{seval}_{D_\perp, D}$ is the (strict) application operation in λ_P of type

$$(D\perp \rightharpoonup D), D_\perp \rightharpoonup D.$$

By structural induction, it is easy to verify that for $M \in \Lambda$, M' has type D given that all free variables of M have type $D_\perp$.

It is easy to see the following:

LEMMA **5.2.3.5** *$(-)'$ is a relative interpretation, i.e. for $M, N \in \Lambda$,*

$$\lambda_L \vdash \supset (M \simeq N) \iff \lambda_P \vdash \supset (M' \simeq N').$$

□

REMARK **5.2.3.6 (Moggi)** There is a closed term of type D, namely

$$M \equiv \mathsf{Gr}(\lambda y : D_\perp.(\lambda z : D.\mathbf{I}')(y*))$$

such that for all untyped term N, the above proof system obtained by augmenting λ_P with the three axioms cannot prove $M \simeq N'$.

Semantically, the interpretation of M in the domain $D \cong D_\perp \rightharpoonup D$ is convergence testing.

Contrast this with the usual translation $(-)''$ of untyped λ-calculus to typed λ-calculus with reflexive object which satisfies the following property:

> Any typed term M of type D with all free variables of type D is provably equivalent to N'' for some untyped term N,

This observation throws light on why completeness of λ_L with respect to categories of partial morphisms is unlikely.

LEMMA **5.2.3.7** *Let* $\mathbb{C}$ *be a category of partial morphisms which has partial function space. Suppose* $h : c \times a \rightharpoonup b$ *and* $g : d \to c$ *is total., then*

$$p\Lambda_{d,a,b}(h \circ (g \times \mathsf{id}_a)) = p\Lambda_{c,a,b}(h) \circ g.$$

PROOF Straightforward corollary of the naturality of $\phi_{-}^{a,b}$ (with reference to Definition 5.1.6.3). □

LEMMA **5.2.3.8** *Let* I *be an indexing set. Let* h *and* l_i *for* $i \in I$ *be monos in a category* $\mathbb{C}$ *in which pullbacks of monos exist. Then,*

$$(\dagger) \qquad \bigcap_i h^{-1}(l_i) = h^{-1}(\bigcap_i l_i).$$

PROOF To prove (†), it suffices to show: for any L such that:

$$\forall i.\exists t_i.L = h^{-1}(l_i) \circ t_i \Rightarrow \exists!k.L = h^{-1}(\bigcap_i l_i) \circ k.$$

Suppose the antecedent, then $\forall i.\exists t_i.h \circ L = h \circ h^{-1}(l_i) \circ t_i = l_i \circ l_i^{-1}(h) \circ t_i$. Hence, $\exists!k'.h \circ L = (\bigcap_i l_i) \circ k'$. Finally, by pullback of h and $\bigcap_i l_i$, $\exists!k.L = h^{-1}(\bigcap_i l_i) \circ k$. □

LEMMA **5.2.3.9 (Substitution 1)** *(i) Let* $C \equiv \vec{x}, \vec{y} : \vec{\sigma}, \vec{\tau}$ *and* $B \equiv \vec{x} : \vec{\sigma}$. *Let* $\Gamma \equiv \{ q_i : 1 \leqslant i \leqslant n \}, s \simeq t$ *be as before and suppose that they are well-typed with respect to the type assignment* B. *Note that this implies that the free variables of all equations in* Γ *and* s, t *are included in* $\vec{x}$; *and that they are also well-typed with respect to* C. *Then,*

(1) $[\![s]\!](C) = [\![s]\!](B) \circ \Pi_B^C$,

(2) $[\![s \simeq t]\!](C) = (\Pi_B^C)^{-1}([\![s \simeq t]\!](B))$,

(3) $\bigcap_i [\![q_i]\!](C) = (\Pi_B^C)^{-1}(\bigcap_i [\![q_i]\!](B))$,

(ii) Let $B, \vec{y} : \vec{\tau}.s : \nu$ *and* $B \subseteq C.\vec{u} : \vec{\tau}$. *Then,*

$$[\![s[\vec{y} := \vec{u}]]\!](C) = [\![s]\!](B, \vec{y} : \vec{\tau}) \circ \langle \Pi_B^C, \overline{[\![u_1]\!](C)}, \cdots, \overline{[\![u_m]\!](C)} \rangle.$$

PROOF (i) We prove each item in turn.

(1) Entirely similar to the proof of Lemma 5.3.2.8 (Substitution)(i) in the sequel.

(2) By an appeal to (1), it suffices to show:
CLAIM: Suppose in a category $\mathbb{C}$, $f_i = g_i \circ l$ for $i = 1, 2$. Then,

$$\mathrm{eq}_{\mathbb{C}}(f_1, f_2) = l^{-1}(\mathrm{eq}_{\mathbb{C}}(g_1, g_2)).$$

Writing Q for $\mathrm{eq}_{\mathbb{C}}(g_1, g_2)$, we have

$$\begin{aligned} f_1 \circ l^{-1}(Q) &= g_1 \circ l \circ l^{-1}(Q) \\ &= g_1 \circ Q \circ Q^{-1}(l) \\ &= g_2 \circ Q \circ Q^{-1}(l) \\ &= f_2 \circ l^{-1}(Q). \end{aligned}$$

Suppose $f_1 \circ h = f_2 \circ h$, then $g_1 \circ l \circ h = g_2 \circ l \circ h$. By definition, $\exists! k.l \circ h = Q \circ k$. By pullback of l and Q, we have

$$\exists! k'.h = l^{-1}(Q) \circ k'.$$

(3) Follows immediately from (2) and Lemma 5.2.3.8. (ii) Entirely similar to the proof of Lemma 5.3.2.8(ii).

□

THEOREM **5.2.3.10** (Soundness) *The typed QS-λ_L-calculus is sound with respect to interpretation in categories of partial morphisms.*

PROOF The proof consists in verifying the soundness of each inference rule. The rule (Log.1) is readily seen to be valid. Rule (Log.2) follows from the transitivity of the "factoring" relation, *i.e.* if $f \leqslant g$ and $g \leqslant h$, then $f \leqslant h$. The rest of the rules will be dealt with in turn.

Let $\Gamma \equiv \{q_1, \cdots, q_n\}$.

- (Subst)
  Let $\Gamma = \{q_1, \cdots, q_n\}$.
  CLAIM: For q a strong equation,

  $$[\![q[\vec{y} := \vec{u}]]\!](C) = h^{-1}([\![q]\!](B, \vec{y} : \vec{\tau}))$$

  where $h = \langle \Pi^B_C, \overline{[\![u_1]\!](C)}, \cdots, \overline{[\![u_m]\!](C)} \rangle$. This can be verified by an appeal to (ii) and the Claim in the proof of (i)(2) of the above Substitution Lemma. By Claim, we have

  $$\begin{aligned}
  [\![\Gamma[\vec{y} := \vec{u}]]\!](C) &= (\bigcap h^{-1}([\![q_i]\!](B, \vec{y} : \vec{\tau})) && \text{Lemma 5.2.3.8} \\
  &= (h^{-1}(\bigcap [\![q_i]\!](B, \vec{y} : \vec{\tau})) && \text{some } k\text{; by premise} \\
  &= h^{-1}([\![q]\!](B, \vec{y} : \vec{\tau}) \circ k) \\
  &= h^{-1}([\![q]\!](B, \vec{y} : \vec{\tau})) \circ k' && \text{by Claim} \\
  &= [\![q[\vec{y} := \vec{u}]]\!](C) \circ k'
  \end{aligned}$$

  and we are done.
  NOTATION: We introduce standard names for some recurring notions as follows. Let $l \equiv [\![\Gamma]\!](B)$, $[\![s]\!](B) \equiv [i_1, f_1], [\![t]\!](B) \equiv [i_2, f_2]$ and that $[g] = [\![s \simeq t]\!](B)$.
- (Ex)
  Note that $[\![s\downarrow]\!](B) = [i_1]$. Now, by premise, $l \leqslant g$ and $l \leqslant i_1$. By pullback of g and i, there is a morphism u s.t.

  $$l = g \circ g^{-1}(i_1) \circ u = i_1 \circ i_1^{-1}(g) \circ u.$$

  Since g is an equalizer in $\mathcal{M}$-Ptl(A), there exists an isomorphism v such that

  $$i_1 \circ i_1^{-1}(g) = i_2 \circ i_2^{-1}(g) \circ v.$$

  Hence, $l = i_2 \circ i_2^{-1}(g) \circ v \circ u$, *i.e.* $l \leqslant i_2$.
- (LStr)
  Let f, f' be partial morphisms. We remark that

$$\mathsf{dom}(f \circ f') \leqslant \mathsf{dom}(f').$$

Omitting the cumbersome details about types, recall that

$$[\![s(\vec{t})]\!] = \mathrm{peval} \circ \langle [\![s]\!], \langle \overline{[\![t_i]\!]} | i \rangle \rangle.$$

By premise, $l \leqslant [\![(s(\vec{t}))\downarrow]\!]$. Then, by the preceding remark,

$$l \leqslant \mathsf{dom}(\langle [\![s]\!], \langle \overline{[\![t_i]\!]} | i \rangle \rangle) = \mathsf{dom}([\![s]\!]) \cap \mathsf{dom}(\langle \overline{[\![t_i]\!]} | i \rangle).$$

Since $\mathsf{dom}(\langle \overline{[\![t_i]\!]} | i \rangle) = \mathsf{id}$, we have $l \leqslant \mathsf{dom}([\![s]\!])$.

- (Ex.λ)
  The validity follows trivially from the totality of $[\![\lambda \vec{x}.\vec{\sigma}.t]\!](B)$ which is $\mathrm{p}\Lambda(g)$ for some partial morphism g.
- The rules (Eq.1) and (Eq.2) are obviously valid.
- (Eq.4) By premise, we have $l \cap i_1 \leqslant g$ and $l \cap i_2 \leqslant g$. There exist morphisms e_1, e_2 s.t.

(1) $$i_1 \circ i_1^{-1}(g) \circ e_1 = g \circ g^{-1}(i_1) \circ e_1 = l \circ l^{-1}(i_1),$$

(2) $$i_2 \circ i_2^{-1}(g) \circ e_2 = g \circ g^{-1}(i_2) \circ e_2 = l \circ l^{-1}(i_2).$$

Now, since g is the equalizer of $[\![s]\!](B)$ and $[\![t]\!](B)$, there exists isomorphism v s.t.

(3) $$f_1 \circ i_1^{-1}(g) = f_2 \circ i_2^{-1}(g) \circ v,$$

$$i_1 \circ i_1^{-1}(g) = i_2 \circ i_2^{-1}(g) \circ v.$$

Hence, from (1), we have $l \circ l^{-1}(i_1) = i_2 \circ i_2^{-1}(g) \circ v \circ e_1$; and so, by considering pullback of l and i_2, there exists morphism u s.t.

(4) $$l^{-1}(i_1) = l^{-1}(i_2) \circ u.$$

We assert that u is actually an isomorphism, because by an entirely symmetrical argument, we have

$$l^{-1}(i_2) = l^{-1}(i_1) \circ u';$$

and that both $l^{-1}(i_1)$ and $l^{-1}(i_2)$ are mono.
Also, $i_2 \circ i_2^{-1}(g) \circ v \circ e_1 = i_2 \circ i_2^{-1}(g) \circ e_2 \circ u$. Since $i_2 \circ i_2^{-1}(g)$ are monos, we have

(5) $$v \circ e_1 = e_2 \circ u.$$

Finally, we claim that $[i_1, f_1] \circ [l] = [i_2, f_2] \circ [l]$. Consequently, since g is the equalizer, we have $l \leqslant g$ and we are done.
To prove the claim, observe that from (1) and (2), we have $i_1^{-1}(g) \circ e_1 = i_1^{-1}(l)$ and $i_2^{-1}(g) \circ e_2 = i_2^{-1}(l)$. Hence,

$$\begin{aligned} f_1 \circ i_1^{-1}(l) &= f_1 \circ i_1^{-1}(g) \circ e_1 && \text{by (3)} \\ &= f_2 \circ i_2^{-1}(g) \circ v \circ e_1 && \text{by (5)} \\ &= f_2 \circ i_2^{-1}(g) \circ e_2 \circ u && \text{by (2)} \\ &= f_2 \circ i_2^{-1}(l) \circ u. \end{aligned}$$

Also, from (4), we get $l \circ l^{-1}(i_1) = l \circ l^{-1}(i_2) \circ u$. Hence, $[i_1, f_1] \circ [l] = [i_2, f_2] \circ [l]$.

- (Eq.3)
Suppose the interpretation of the premise is sound. Then, $[\![s]\!](B) \circ l = [\![t]\!](B) \circ l$ and $[\![t]\!](B) \circ l = [\![u]\!](B) \circ l$. Hence, $[\![s]\!](B) \circ l = [\![u]\!](B) \circ l$; and so, by definition of $[\![s \simeq u]\!](B)$, we have

  $$l \equiv [\![\Gamma]\!](B) \leqslant [\![s \simeq u]\!](B).$$

- (ξ)
Let $C \equiv B \cup D$ with $D \equiv \vec{y} : \vec{\tau}$. By naturality of $(-) \times (-)$ in the first argument, we have,

  $$\Pi_B^C \circ [\![\Gamma]\!](B) \times \mathrm{id}_D = [\![\Gamma]\!](B) \circ p_{E,D}$$

  where $[\![\Gamma]\!](B) : E \rightharpoonup B$. By Lemma 5.2.3.9(i)(3), $[\![\Gamma]\!](C) = (\Pi_B^C)^{-1}[\![\Gamma]\!](B)$, and so, $[\![\Gamma]\!](C) \circ h = [\![\Gamma]\!](B) \times \mathrm{id}_D$ for some morphism h.
  Now, suppose interpretation of the premise is sound, *i.e.*

  $$[\![s]\!](C) \circ [\![\Gamma]\!](C) = [\![t]\!](C) \circ [\![\Gamma]\!](C);$$

  composition with h on the right then yields

  $$[\![s]\!](C) \circ [\![\Gamma]\!](B) \times \mathrm{id}_D = [\![t]\!](C) \circ [\![\Gamma]\!](B) \times \mathrm{id}_D.$$

  Applying $\mathrm{p}\Lambda(-)$ to both sides, then by an appeal to Lemma 5.2.3.7, we have

  $$\mathrm{p}\Lambda([\![s]\!](C)) \circ [\![\Gamma]\!](B) = \mathrm{p}\Lambda([\![t]\!](C)) \circ [\![\Gamma]\!](B);$$

  from which we conclude $[\![\Gamma]\!](B) \leqslant \mathrm{eq}([\![\lambda \vec{y} : \vec{\tau}.s]\!](B), [\![\lambda \vec{y} : \vec{\tau}.t]\!](B))$.

- (β)
  Let $B \equiv \vec{x} : \vec{\sigma}, D \equiv \vec{y} : \vec{\tau}$ and $C \equiv B \cup D$. Note that $\langle \overline{[\![y_i]\!](C)} | i \rangle = \Pi_D^C$. Now,

$$\begin{aligned}
[\![(\lambda \vec{y}.\vec{\tau}.t)(\vec{y})]\!](C) &= \text{peval} \circ \langle [\![\lambda \vec{y} : \vec{\tau}.t]\!](C), \langle \overline{[\![y_i]\!](C)} | i \rangle \rangle \\
&= \text{peval} \circ \langle \text{p}\Lambda([\![t]\!](C, D)), \Pi_D^C \rangle && \text{Lemma 5.2.3.9} \\
&= \text{peval} \circ \langle \text{p}\Lambda([\![t]\!](C) \circ \Pi_C^{C,D}), \Pi_D^C \rangle \\
&= \text{peval} \circ \langle \text{p}\Lambda([\![t]\!](C) \circ (\Pi_B^C \times \text{id}_D)), \Pi_D^C \rangle && \text{Lemma 5.2.3.7} \\
&= \text{peval} \circ \langle \text{p}\Lambda([\![t]\!](C)) \circ \Pi_B^C, \Pi_D^C \rangle \\
&= \text{peval} \circ \text{p}\Lambda([\![t]\!](C)) \times \text{id}_D \\
&= [\![t]\!](C).
\end{aligned}$$

  The equalizer of two identical partial morphisms with source Σ is id_Σ, hence the result follows trivially.

- (cond-η)
  Let $l \equiv [\![\Gamma]\!](B), [\![t]\!](B) \equiv [i, f], C \equiv \vec{y} : \vec{\tau}$ and $\Sigma' \equiv \Pi [\![\tau_i]\!]_\perp$. By premise, $l \leqslant i$, say $l = i \circ h$. It suffices to prove

$$(\dagger) \qquad [\![\lambda \vec{y} : \vec{\tau}.t(\vec{y})]\!](B) \circ l = [\![t]\!](B) \circ l;$$

  for then, by definition, $l \leqslant [\![\lambda \vec{y} : \vec{\tau}.t(\vec{y}) \simeq t]\!](B)$. Now,

$$\begin{aligned}
[\![\lambda \vec{y} : \vec{\tau}.t(\vec{y})]\!](B) &= \text{p}\Lambda(\text{peval} \circ \langle [\![t]\!](B, C), \langle \overline{[\![y_i]\!](B, C)} | i \rangle \rangle) \\
&= \text{p}\Lambda(\text{peval} \circ [\![t]\!](B) \times \text{id}_{\Sigma'})
\end{aligned}$$

  which is a total morphism extending $[\![t]\!](B)$; hence

$$[\![\lambda \vec{y} : \vec{\tau}.t(\vec{y})]\!](B) \circ \text{dom}([\![t]\!](B)) = [\![t]\!](B) \circ \text{dom}([\![t]\!](B)).$$

  Since $l = i \circ h$ where $i \equiv \text{dom}([\![t]\!](B))$, we have $(\dagger)$.

□

5.2.4 Completeness of the Category-Theoretic Interpretation

This section will be devoted to the construction of a category of partial morphisms that correspond to the "initial term model" for a certain class of typed λ_L-calculus. Consider $\lambda_{L,\mathsf{C}}$, the class of λ_L-calculi in which *convergence testing* is definable. That is to say, there exist $\mathsf{C}_{\tau,\nu} \in \textbf{Terms}$ such that for all well-typing $\vec{x} : \vec{\sigma}.s : \tau$,

$$\lambda_L \vdash \vec{x} : \vec{\sigma}.\Gamma \supset s\downarrow \iff \lambda_{L,\mathsf{C}} \vdash \vec{x} : \vec{\sigma}.\Gamma \supset (\mathsf{C}_{\tau,\nu} s \simeq \lambda y : \nu.y)$$

where σ, τ, ν range over **Types**.

In the following, all conjunctions of the form $\bigwedge_{i\in I} p_i(\vec{x})$ are assumed to be finite, *i.e.* I is finite.

DEFINITION **5.2.4.1** We define a category $\mathbb{C}(\lambda_{L,\mathsf{C}})$ as follows:

- Objects: entities of the form $\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\}$;
- Morphisms: equivalence classes, $\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\} \xrightarrow{[\vec{s}(\vec{x})]} \{\, \vec{y} : \vec{\tau}. \bigwedge_j q_j(\vec{y}) \,\}$, satisfying the following:

  (1) $\vec{x} : \vec{\sigma}.\vec{s}(\vec{x}) : \vec{\tau}$ is well-typed,

  (2) $\forall j. \lambda_{L,\mathsf{C}} \vdash \bigwedge_i p_i(\vec{x}) \supset q_j(\vec{s}(\vec{x}))$;

  and that $[\vec{s}(\vec{x})]$ is the equivalence class with respect to the relation $\sim_{\bigwedge_i p_i(\vec{x})}$ defined as follows:

$$\vec{s}(\vec{x}) \sim_{\bigwedge_i p_i(\vec{x})} \vec{t}(\vec{x}) \stackrel{\text{def}}{=} \bigwedge_i p_i(\vec{x}) \supset (\vec{s}(\vec{x}) \simeq \vec{t}(\vec{x})).$$

- For any object $\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\}$, the identity morphism is $[\vec{x}]$.
- For $\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\} \xrightarrow{[\vec{s}(\vec{x})]} \{\, \vec{y} : \vec{\tau}. \bigwedge_i q_i(\vec{y}) \,\} \xrightarrow{[\vec{t}(\vec{y})]} \{\, \vec{z} : \vec{\mu}. \bigwedge_i r_i(\vec{z}) \,\}$, the composition $[\vec{t}(\vec{y})] \circ [\vec{s}(\vec{x})]$ is $[\vec{t}(\vec{s}(\vec{x}))]$.

It is easy to see that $\mathbb{C}(\lambda_{L,\mathsf{C}})$ is a category.

Next, we ascribe a domain structure to $\mathbb{C}(\lambda_{L,\mathsf{C}})$. Define, for each object, $\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\}$, the set of admissible subobjects, $\mathcal{M}(\{\, \vec{x} : \vec{\sigma}. \bigwedge_i p_i(\vec{x}) \,\})$, consisting of all morphisms of the form:

$$\{\, \vec{x} : \vec{\sigma}. \bigwedge_i q_i(\vec{x}) \,\} \xrightarrow{[\vec{x}]} \{\, \vec{x} : \vec{\sigma}. \bigwedge_j p_j(\vec{x}) \,\}.$$

Before we prove that $\mathcal{M}$ is indeed a domain structure, we note the following useful Lemma:

LEMMA **5.2.4.2** *The following is a pullback square:*

$$\begin{array}{ccc}
\{\, \vec{x} : \vec{\sigma}. \bigwedge p_i(\vec{x}) \wedge \bigwedge q_j(\vec{x}) \,\} & \xrightarrow{[\vec{x}]} & \{\, \vec{x} : \vec{\sigma}. \bigwedge p_i(\vec{x}) \,\} \\
\Big\downarrow{\scriptstyle [\vec{x}]} & & \Big\downarrow{\scriptstyle [\vec{x}]} \\
\{\, \vec{x} : \vec{\sigma}. \bigwedge q_j(\vec{x}) \,\} & \xrightarrow[[\vec{x}]]{} & \{\, \vec{x} : \vec{\sigma}. \bigwedge r_k(\vec{x}) \,\}
\end{array}$$

□

LEMMA **5.2.4.3** *$\mathcal{M}$ is a domain structure on $\mathbb{C}(\lambda_{L,\mathbb{C}})$*

PROOF We verify the four conditions as stated in Definition 5.1.1.3. First, observe that

$$\{\,\vec{x}:\vec{\sigma}.\bigwedge q_i(\vec{x})\,\} \xrightarrow{[\vec{x}]} \{\,\vec{x}:\vec{\sigma}.\bigwedge p_i(\vec{x})\,\}$$

is a mono. For suppose for $k = 1, 2$,

$$\{\,\vec{y}:\vec{\tau}.\bigwedge r_i(\vec{y})\,\} \xrightarrow{[\vec{s_k}(\vec{y})]} \{\,\vec{x}:\vec{\sigma}.\bigwedge q_i(\vec{x})\,\}$$

and that

$$[\vec{x}] \circ [\vec{s_1}(\vec{y})] \sim_{\bigwedge r_i(\vec{y})} [\vec{x}] \circ [\vec{s_2}(\vec{y})].$$

Then, $[\vec{s_1}(\vec{y})] \sim_{\bigwedge r_i(\vec{y})} [\vec{s_2}(\vec{y})]$. Condition (3) is obvious. The following pullback square (which is easy to verify) should suffice for (4).

$$\begin{array}{ccc}
\{\,\vec{y}:\vec{\tau}.\bigwedge r_i(\vec{y}) \wedge \bigwedge q_i(\vec{s}(\vec{y}))\,\} & \xrightarrow{[\vec{s}(\vec{y})]} & \{\,\vec{x}:\vec{\sigma}.\bigwedge q_i(\vec{x})\,\} \\
\Big\downarrow{\scriptstyle [\vec{y}]} & & \Big\downarrow{\scriptstyle [\vec{x}]} \\
\{\,\vec{y}:\vec{\tau}.\bigwedge r_i(\vec{y})\,\} & \xrightarrow[[\vec{s}(\vec{y})]]{} & \{\,\vec{x}:\vec{\sigma}.\bigwedge p_i(\vec{x})\,\}
\end{array}$$

□

The interpretation of types is as follows. Let $\vec{x} \equiv x_1, \cdots, x_n$

$$\forall \vec{\sigma} \in \mathbf{Types}.[\![\vec{\sigma}]\!] \stackrel{\text{def}}{=} \{\,\vec{x}:\vec{\sigma}.x_1\downarrow, \cdots, x_n\downarrow\,\}.$$

REMARK **5.2.4.4** We claim that the interpretation is "good enough" for a *sound* category-theoretic interpretation of $\lambda_{L,\mathbb{C}}$. Note that with reference to the above interpretation, it is easy to verify that:

$$[\![\vec{\sigma}]\!] = \Pi_i[\![\sigma_i]\!] \quad \text{and } [\![\sigma]\!]_\perp = \{\,x:\sigma.x \simeq x\,\}.$$

Henceforth, we will abbreviate $\{\,\vec{x}:\vec{\sigma}.\bigwedge_i x_i \simeq x_i\,\}$ as $\{\,\vec{x}:\vec{\sigma}\,\}$.

Note, however, $[\![\vec{\sigma} \rightharpoonup \tau]\!] \stackrel{\text{def}}{=} \{\,z:\vec{\sigma} \rightharpoonup \tau.z\downarrow\,\}$ is, in general, *not* $(\Pi[\![\sigma_i]\!]_\perp) \rightharpoonup [\![\tau]\!]$, — the partial function space from $(\Pi[\![\sigma_i]\!]_\perp)$ to $[\![\tau]\!]$. Nevertheless, to ensure a sound interpretation of $\lambda_{L,\mathbb{C}}$, it suffices to check that

$$(*) \qquad f = \text{peval} \circ \text{p}\Lambda(f) \times \text{id}$$

holds for all partial morphisms f that might possibly arise in the course of the interpretation given a prescription of the morphisms "peval" and "p$\Lambda(f)$". As

it turns out, the respective prescriptions do not qualify as peval or $p\Lambda(-)$ (in that the respective isomorphisms $\phi_{-}^{a,b}, \eta_{-}^{b}$ fail to satisfy the naturality condition) according to their definitions to be introduced next; they only *behave* as peval and $p\Lambda(-)$ *should* when their use is restricted to the class of partial morphisms that might possibly be encountered in the interpretation.

NOTATION **5.2.4.5** From now on, we present partial morphisms of the form $[\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})$ as follows:

$$\begin{array}{ccc} \{\,\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}.\bigwedge p_i^t\,\} & \xrightarrow{[t(\vec{x},\vec{y})]} & \{\,z:\mu.z{\downarrow}\,\} \\ \Big\downarrow{\scriptstyle [\vec{x},\vec{y}]} & & \\ \{\,\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}\,\} & \xrightarrow[[\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})]{} & \{\,z:\mu.z{\downarrow}\,\} \end{array}$$

Note that for all such terms t, $\lambda_{L,C} \vdash \bigwedge p_i^t \supset t{\downarrow}$. The strong equations parametrized over t, p_i^t, may be seen as "filtering" out the extent over which free variables in t should range so as to ensure $t{\downarrow}$, the existence of t.

DEFINITION **5.2.4.6** The *pseudo* peval and $p\Lambda(-)$ prescribed to interpret abstraction and application are as follows:

Define $\text{peval}_{\vec{\tau},\mu} : [\![\vec{\tau} \rightharpoonup \mu]\!] \times \Pi[\![\tau_i]\!]_\perp \rightharpoonup [\![\mu]\!]$:

$$\begin{array}{ccc} \{\,y,\vec{z}:\vec{\tau}\rightharpoonup\mu,\vec{\tau}.y{\downarrow}\wedge(\mathsf{C}(y(\vec{z}))\simeq\mathbf{I})\,\} & \xrightarrow{[y(\vec{z})]} & \{\,u:\mu.u{\downarrow}\,\} \\ \Big\downarrow{\scriptstyle [y,\vec{z}]} & & \\ \{\,y:\vec{\tau}\rightharpoonup\mu.y{\downarrow}\,\}\times\{\,\vec{z}:\vec{\tau}\,\} & \xrightarrow[\text{peval}_{\vec{\tau},\mu}]{} & \{\,u:\mu.u{\downarrow}\,\} \end{array}$$

We define $p\Lambda_{\vec{\sigma},\vec{\tau},\mu}([\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})) \stackrel{\text{def}}{=} [\lambda\vec{y}:\tau.[\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})] : \Pi[\![\sigma_i]\!]_\perp \rightharpoonup [\![\vec{\tau}\rightharpoonup\mu]\!]$ (a total morphism) such that

$$\{\,\vec{x}:\vec{\sigma}\,\} \xrightarrow{[\lambda\vec{y}.[\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})]} \{\,u:\vec{\tau}\rightharpoonup\mu.u{\downarrow}\,\}.$$

Observe that to ensure a sound interpretation, it suffices to verify, for all partial morphisms of the form $[\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})$ with $\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}.t:\mu$:

$$(\ddagger) \qquad [\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}) = \text{peval}_{\vec{\tau},\mu} \circ p\Lambda_{\vec{\sigma},\vec{\tau},\mu}([\![t]\!](\vec{x},\vec{y}:\vec{\sigma},\vec{\tau})) \times \text{id}_{\vec{\tau}}$$

which we will prove as part of the following Proposition.

PROPOSITION **5.2.4.7** *Suppose* $\vec{x} : \vec{\sigma}.s(\vec{x}) : \tau$ *is well-typed. Then,*

(i) according to the prescribed category-theoretic interpretation prescribed earlier,

$$\begin{array}{ccc} \{\vec{x} : \vec{\sigma}.\bigwedge p_i^s\} & \xrightarrow{[s(\vec{x})]} & \{y : \tau.y\downarrow\} \\ \Big\downarrow{\scriptstyle [\vec{x}]} & & \\ \{\vec{x} : \vec{\sigma}\} & \xrightarrow[[\![s(\vec{x})]\!](\vec{x} : \vec{\sigma})]{} & \{y : \tau.y\downarrow\} \end{array}$$

(†) $$\forall i.\lambda_{L,C} \vdash s(\vec{x})\downarrow \supset p_i^s.$$

(ii) The preceding statement (‡) is valid.

PROOF (i) We prove by structural induction.

- For the base case of variables: $[\![x_i]\!](\vec{x} : \vec{\sigma})$, we have

$$\begin{array}{ccc} \{\vec{x} : \vec{\sigma}.x_i\downarrow\} & \xrightarrow{[x_i]} & \{x_i : \sigma_i.x_i\downarrow\} \\ \Big\downarrow{\scriptstyle [\vec{x}]} & & \\ \{\vec{x} : \vec{\sigma}\} & \xrightarrow[[\![x_i]\!](\vec{x} : \vec{\sigma})]{} & \{x_i : \sigma_i.x_i\downarrow\} \end{array}$$

which is composed of the following:

$$\begin{array}{ccccc} \{\vec{x} : \vec{\sigma}.x_i\downarrow\} & \xrightarrow{[x_i]} & \{x_i : \sigma_i.x_i\downarrow\} & \xrightarrow{[x_i]} & \{x_i : \sigma_i.x_i\downarrow\} \\ \Big\downarrow{\scriptstyle [\vec{x}]} & & \Big\downarrow{\scriptstyle [x_i]} & & \\ \{\vec{x} : \vec{\sigma}\} & \xrightarrow{[x_i]} & \{x_i : \sigma_i\} & \xrightarrow{\text{open}_{\sigma_i}} & \{x_i : \sigma_i.x_i\downarrow\} \\ \Big\downarrow{\scriptstyle [\vec{x}]} & & & & \\ \{\vec{x} : \vec{\sigma}\} & \xrightarrow[\Pi_{\sigma_i}^{\vec{\sigma}} = [x_i]]{} & \{x_i : \sigma_i\} & & \end{array}$$

(†) is clearly valid.

- The case for abstraction is trivial and that (†) is clearly true.
- The case for $[\![\vec{t}]\!](\vec{x} : \vec{\sigma})$ is straightforward.
- For the inductive case of $[\![s(\vec{t})]\!](\vec{x} : \vec{\sigma})$, the following diagram which reveals the composition:

$$[\![\vec{t}]\!](\vec{x} : \vec{\sigma}) = \mathrm{peval}_{\vec{\tau},\mu} \circ \langle [\![s]\!](\vec{x} : \vec{\sigma}), \langle \overline{[\![t_i]\!](\vec{x} : \vec{\sigma})} | i \rangle \rangle$$

should suffice.

$$\begin{array}{ccccc}
\{\vec{x} : \vec{\sigma}. \bigwedge p_i^s \wedge \\ s(\vec{x})(\vec{t}(\vec{x}))\downarrow\} & \xrightarrow{[s(\vec{x}), \vec{t}(\vec{x})]} & \{y, \vec{z} : \vec{\tau} \rightharpoonup \mu, \vec{\tau}. y\downarrow \wedge \\ (\mathsf{C}(y(\vec{z})) \simeq \mathbf{I})\} & \xrightarrow{[y(\vec{z})]} & \{u : \mu. u\downarrow\} \\
\downarrow{\scriptstyle [\vec{x}]} & \text{(pllbk sq.)} & \downarrow{\scriptstyle [y, \vec{z}]} & & \\
\{\vec{x} : \vec{\sigma}. \bigwedge p_i^s\} & \xrightarrow{[s(\vec{x}), \vec{t}(\vec{x})]} & \{y : \vec{\tau} \rightharpoonup \mu. y\downarrow\} \\ \times \{\vec{z} : \vec{\tau}\} & & \xrightarrow[\mathrm{peval}_{\vec{\tau},\mu}]{} & \{u : \mu. u\downarrow\} \\
\downarrow{\scriptstyle [\vec{x}]} & & & & \\
\{\vec{x} : \vec{\sigma}\} & \xrightarrow[\langle [\![s]\!], \langle \overline{[\![t_i]\!]} | i \rangle \rangle]{} & \{y : \vec{\tau} \rightharpoonup \mu. y\downarrow\} \\ \times \{\vec{z} : \vec{\tau}\} & &
\end{array}$$

We have $[\![s(\vec{x})(\vec{t}(\vec{x}))]\!](\vec{x} : \vec{\sigma}) = [y(\vec{z})] \circ [s(\vec{x}), \vec{t}(\vec{x})] = [s(\vec{x})(\vec{t}(\vec{x}))]$ and note that

$$\bigwedge p_i^{s(\vec{t})} \equiv \bigwedge p_i^s \wedge s(\vec{x})(\vec{t}(\vec{x}))\downarrow.$$

To show (†), observe that by induction hypothesis, we have

$$\forall i. \lambda_{L,\mathsf{C}} \vdash s(\vec{x})\downarrow \supset p_i^s.$$

Also, $s(\vec{x})(\vec{t}(\vec{x}))\downarrow \supset s(\vec{x})\downarrow$ by (LStr); and so, by (Log.2),

$$\forall i. \lambda_{L,\mathsf{C}} \vdash s(\vec{x})(\vec{t}(\vec{x}))\downarrow \supset p_i^s.$$

Hence, $\forall i. \lambda_{L,\mathsf{C}} \vdash s(\vec{x})(\vec{t}(\vec{x}))\downarrow \supset p_i^{s(\vec{t})}$.

(ii): We show: for any well-typing $\vec{x}, \vec{y} : \vec{\sigma}, \vec{\tau}. t(\vec{x}, \vec{y}) : \mu$,

(‡) $$[\![t]\!](\vec{x}, \vec{y} : \vec{\sigma}, \vec{\tau}) = \mathrm{peval}_{\vec{\tau},\mu} \circ \mathrm{p}\Lambda_{\vec{\sigma},\vec{\tau},\mu}([\![t]\!](\vec{x}, \vec{y} : \vec{\sigma}, \vec{\tau})) \times \mathrm{id}_{\vec{\tau}}.$$

Observe that (‡) implies that the category-theoretic interpretation of $\lambda_{L,\mathsf{C}}$ is sound.

Consider the following diagram:

$$\begin{array}{ccccc}
\{\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}.\bigwedge q_i^t\} & \xrightarrow{[\lambda\vec{y}.t,\vec{y}]} & \{u,\vec{y}\ :\ \vec{\tau}\ \rightharpoonup\ \mu,\vec{\tau}.u{\downarrow} \wedge(\mathsf{C}(u(\vec{y}))\simeq \mathbf{I})\} & \xrightarrow{[u(\vec{y})]} & \{z:\mu.z{\downarrow}\} \\
\Big\downarrow{\scriptstyle [\vec{x},\vec{y}]} & \text{(pllbk sq.)} & \Big\downarrow{\scriptstyle [u,\vec{y}]} & & \\
\{\vec{x}:\vec{\sigma}\}\times\{\vec{y}:\vec{\tau}\} & \xrightarrow{\mathrm{p}\Lambda([\![t]\!])\times \mathrm{id}} & \{u:\vec{\tau}\rightharpoonup\mu.u{\downarrow}\}\times\{\vec{y}:\vec{\tau}\} & \xrightarrow[\mathrm{peval}_{\vec{\tau},\mu}]{} & \{z:\mu.z{\downarrow}\}
\end{array}$$

The above composition yields:

$$\begin{array}{ccc}
\{\vec{x},\vec{y}:\vec{\sigma},\vec{\tau}.\bigwedge q_i^t\} & \xrightarrow{[t]} & \{z:\mu.z{\downarrow}\} \\
\Big\downarrow{\scriptstyle [\vec{x},\vec{y}]} & & \\
\{\vec{x}:\vec{\sigma}\}\times\{\vec{y}:\vec{\tau}\} & \xrightarrow[\mathrm{peval}\circ \mathrm{p}\Lambda([\![t]\!])\times \mathrm{id}]{} & \{z:\mu.z{\downarrow}\}
\end{array}$$

To show (‡), it remains to show:

$$\lambda_{L,\mathsf{C}} \vdash \bigwedge p_i^t \supset \bigwedge q_i^t \ \&\ \lambda_{L,\mathsf{C}} \vdash \bigwedge q_i^t \supset \bigwedge p_i^t.$$

By definition of the above partial morphism peval $\circ$ p$\Lambda([\![t]\!]) \times$ id, we have $\bigwedge q_i^t \supset t{\downarrow}$. Applying (i) to t, we have $t{\downarrow} \supset \bigwedge p_i^t$; and so, by (Log.2)

$$\lambda_{L,\mathsf{C}} \vdash \bigwedge q_i^t \supset \bigwedge p_i^t.$$

With reference to the pullback square, observe that for Γ — any finite conjunction of strong equation such that

$$\Gamma \supset (u \simeq \lambda\vec{y}.t)\ \&\ \Gamma \supset (\mathsf{C}(u(\vec{y})) \simeq \mathbf{I}),$$

(which imply, by (β) and (Cong) that $\Gamma \supset t{\downarrow}$) we deduce, by universality of pullback, that

$$\lambda_{L,\mathsf{C}} \vdash \Gamma \supset \bigwedge q_i^t.$$

Hence, in particular, $\lambda_{L,\mathsf{C}} \vdash \bigwedge p_i^t \supset \bigwedge q_i^t$ and we are done. □

THEOREM **5.2.4.8 (Soundness and Completeness of $\lambda_{L,C}$)**
The category-theoretic interpretation is complete with respect to the class of λ_L-calculi in which convergence testing is internally definable. That is to say, suppose $\vec{x} : \vec{\sigma}.p_1, \cdots, p_n, q$ are well-typed, then

$$\bigcap_i [\![p_i]\!](\vec{x} : \vec{\sigma}) \leqslant [\![q]\!](\vec{x} : \vec{\sigma}) \iff \lambda_{L,C} \vdash \vec{x} : \vec{\sigma}. \bigwedge_i p_i \supset q.$$

PROOF We prove the completeness part. Let $\Gamma \equiv p_1, \cdots, p_n$, $n \geqslant 0$ and let $q \equiv (s(\vec{x}) \simeq t(\vec{x}))$. Note that the strong equation $s{\downarrow}$ can be represented equivalently as $\mathsf{C}(s(\vec{x})) \simeq \mathbf{I}$. The following proof depends crucially on the assumption that convergence testing is internally definable.

CLAIM: $[\![s \simeq t]\!](\vec{x} : \vec{\sigma}) = \{\, \vec{x} : \vec{\sigma}.s \simeq t \,\} \xrightarrow{[\vec{x}]} \{\, \vec{x} : \vec{\sigma} \,\}$.

We need to verify that the rhs is the equalizer of $[\![s]\!](\vec{x} : \vec{\sigma})$ and $[\![t]\!](\vec{x} : \vec{\sigma})$. Consider the following diagram:

$$\begin{array}{ccccc}
\{\, \vec{x} : \vec{\sigma}.s \simeq t \wedge \bigwedge p_i^s \,\} & \xrightarrow{[\vec{x}]} & \{\, \vec{x} : \vec{\sigma}. \bigwedge p_i^s \,\} & \xrightarrow{[s(\vec{x})]} & \{\, y : \tau.y{\downarrow} \,\} \\
\Big\downarrow{\scriptstyle [\vec{x}]} & \text{pllbk} & \Big\downarrow{\scriptstyle [\vec{x}]} & & \\
\{\, \vec{x} : \vec{\sigma}.s \simeq t \,\} & \xrightarrow{[\vec{x}]} & \{\, \vec{x} : \vec{\sigma} \,\} & \xrightarrow[[\![t]\!]]{[\![s]\!]} & \{\, y : \tau.y{\downarrow} \,\} \\
\Big\uparrow{\scriptstyle [\vec{x}]} & \text{pllbk} & \Big\uparrow{\scriptstyle [\vec{x}]} & & \\
\{\, \vec{x} : \vec{\sigma}.s \simeq t \wedge \bigwedge p_i^t \,\} & \xrightarrow{[\vec{x}]} & \{\, \vec{x} : \vec{\sigma}. \bigwedge p_i^t \,\} & \xrightarrow{[t(\vec{x})]} & \{\, y : \tau.y{\downarrow} \,\}
\end{array}$$

To verify Claim, it suffices to show:

$$\lambda_{L,C} \vdash (s \simeq t \wedge \bigwedge p_i^s) \supset (s \simeq t \wedge \bigwedge p_i^t) \qquad \&$$

$$\lambda_{L,C} \vdash (s \simeq t \wedge \bigwedge p_i^t) \supset (s \simeq t \wedge \bigwedge p_i^s).$$

To see this, we have, by an appeal to Proposition 5.2.4.7,

$$s \simeq t \wedge \bigwedge p_i^s \supset s{\downarrow};$$

and so, by rule (Ex),

$$s \simeq t \wedge \bigwedge p_i^s \supset t{\downarrow}.$$

By Proposition 5.2.4.7(i), we have

$$s \simeq t \wedge \bigwedge p_i^s \supset s \simeq t \wedge \bigwedge p_i^t.$$

The other direction is completely symmetrical.

Hence, $\bigcap_i [\![p_i]\!](\vec{x} : \vec{\sigma})$ is just $\{ \vec{x} : \vec{\sigma}. \bigwedge p_i \} \xrightarrow{[\vec{x}]} \{ \vec{x} : \vec{\sigma} \}$. The premise $[\![\Gamma]\!] \leqslant [\![q]\!]$ implies

$$\{ \vec{x} : \vec{\sigma}. \bigwedge p_i \} \xrightarrow{[\vec{x}]} \{ \vec{x} : \vec{\sigma}. s \simeq t \}.$$

By definition of morphism in $\mathbb{C}(\lambda_{L,\mathbb{C}})$, we conclude, $\lambda_{L,\mathbb{C}} \vdash \Gamma \supset q$. □

5.2.5 Correctness of The Formal λ_L-Calculus

The formal λ_L-calculus may be seen as a proof system for deriving quasi-strong equations between λ-terms. For the proof system to be of any use and relevance, formulae derivable by the system must reflect actual programming behaviour (in our case) in the lazy regime accurately. This is the question of the *correctness* of a calculus (or a proof system) with respect to a programming language.

As far as I know, Gordon Plotkin was the first to study the correctness of variants of λ-calculus (seen here as proof systems yielding equations between programs) with respect to their corresponding programming languages. In the paper [Plo75], Plotkin defines a calculus and a quintessential programming language corresponding to each of two different calling mechanisms: call-by-name and call-by-value. Regarding λ-terms as programs, as is classical, Plotkin maintains that if a reduction relation on λ-terms could be defined which reflects faithfully how computation is carried out, indeed enabling all possible "normal forms" to be captured in a deterministic fashion, then the programming language in question may be said to have been completely determined. In the same spirit, we will regard a programming language as specified by a deterministic reduction relation on λ-terms.

> "On the other hand, the [programming] language can be regarded as giving true equations between programs [= terms of the calculus]. Informally, one programs equals another, operationally, if it can be substituted for the other in all contexts 'without changing the results'. From this point of view, a calculus can be correct with respect to the programming language."
> [Plo75, pp 125-6]

Taking the cue form Plotkin, we turn now to the lazy regime. The calculus or proof system in question is the formal λ_L-calculus and the formulae derivable are *quasi-strong* equations. The programming language is exemplified in the shape of the pure lazy language $\langle \Lambda^o, \Downarrow \rangle$. Note that Abramsky shows that operational equivalence or contextual equivalence i.e.

$$M \sim^{c} N \stackrel{\text{def}}{=} \forall C[\,] \in \Lambda^{o}.C[M]\Downarrow \iff C[N]\Downarrow$$

is identical to bisimulation equivalence (Proposition 4.1.3.5). The operational behaviour of the pure lazy language is therefore completely determined by $\lambda\ell$.

DEFINITION **5.2.5.1** $[\![q]\!]$ and $[\![\Gamma \supset q]\!]$, the respective interpretations of the untyped S-λ_L and QS-λ_L equations in the pure lazy language, or equivalently, in $\lambda\ell$ are truth values. They are defined as follows:

- S-λ_L equation:
  - $[\![s\downarrow]\!] \stackrel{\text{def}}{=} \forall$ closed substitutions $\sigma : \mathsf{Var} \to \Lambda^{o}.s_\sigma\Downarrow$,
  - $[\![s \simeq t]\!] \stackrel{\text{def}}{=} \forall$ closed substitutions $\sigma : \mathsf{Var} \to \Lambda^{o}.s_\sigma \sim^{B} t_\sigma$.
- QS-λ_L equation: Writing p_σ as $s\Downarrow$ or $s_\sigma \sim^{B} t_\sigma$ according as $p \equiv s$ or $s \simeq t$;

$$[\![p_1, \cdots, p_n \supset q]\!] \stackrel{\text{def}}{=} \forall \text{ closed } \sigma : \mathsf{Var} \to \Lambda^{o}.p_{1\sigma} \,\&\, \cdots \,\&\, p_{n\sigma} \Rightarrow q_\sigma.$$

REMARK **5.2.5.2** Recall that for (possibly open) $s, t \in \Lambda$,

$$s \sim^{B} t \stackrel{\text{def}}{=} \text{ for } \textit{all} \text{ closed substitutions } \sigma, s_\sigma \sim^{B} t_\sigma \text{ is valid.}$$

Compare this with $[\![p_1, \cdots, p_n \supset s \simeq t]\!]$ which means: for all those closed substitutions σ which satisfy the precondition $p_{1\sigma} \,\&\, \cdots \,\&\, p_{n\sigma}$, $s_\sigma \simeq t_\sigma$ is valid. According to this interpretation, QS-λ_L-equation furnishes a mechanism with which to reason about the *partial* bisimulation equivalences between two possibly open λ-terms — partial in the sense that one may now specify formally only those closures of s and t that need enter into the consideration.

Alternatively, the "antecedent" Γ in the QS-λ_L equation $\Gamma \supset q$ may be seen as enunciating the *extent* over which free variables in q need to range in asserting q. Herein lies an advantage of QS-λ_L equations over S-λ_L equations — λ_L is more expressive than λ_N which is the formal calculus corresponding to call-by-name argument passing mechanism.

THEOREM **5.2.5.3** (**Correctness**)
The formal untyped λ_L-calculus is correct with respect to $\lambda\ell$ or the pure lazy language. That is to say: for all QS-λ_L equations, $\Gamma \supset q$,

$$\lambda_L \vdash \Gamma \supset q \Rightarrow [\![\Gamma \supset q]\!].$$

PROOF

- (Log.1) and (Log.2) are obvious.

- (Subst) We want to show $\forall\sigma : \mathrm{Var} \to \Lambda^o.(\Gamma[\vec{x} := \vec{P}])_\sigma \supset (q[\vec{x} := \vec{P}])_\sigma$. where $\vec{x} \supseteq \mathrm{FV}(\Gamma, q)$. Given $\sigma : \mathrm{Var} \to \Lambda^o$, we define accordingly an associated closed substitution $\sigma' : \{\vec{x}\} \to \Lambda^o$ by $\sigma'(x_i) = (P_i)_\sigma \in \Lambda^o$. Then, for $p \in \Gamma \cup \{q\}$, $(p[\vec{x} := \vec{P}])_\sigma = p_{\sigma'}$. The result then follows.
- (Ex)
  NOTATION: Let $\Gamma \equiv p_1, \cdots, p_n$ and $q \equiv M \simeq N$. We write $\sigma \in \Lambda^o \restriction \Gamma$ if $p_{1\sigma}$ & $\cdots$ & $p_{n\sigma}$ are satisfied .
  Let $\sigma \in \Lambda^o \restriction \Gamma$. By premise, $M_\sigma \sim^B N_\sigma$ & $M_\sigma\Downarrow$. Since $M_\sigma \sim^B N_\sigma$ is by definition

  $$\forall \vec{P} \subseteq \Lambda^o.M_\sigma\vec{P}\Downarrow \iff N_\sigma\vec{P}\Downarrow.$$

  The result then follows trivially by setting $\vec{P}$ to $\emptyset$.
- (LStr) follows immediately from the observation that

  $$\forall L \in \Lambda^o.L\Uparrow \Rightarrow \forall \vec{P} \subseteq \Lambda^o.L\vec{P}\Uparrow.$$

- The cases of (Eq.1), (Eq.2) and (Eq.3) are valid because $\sim^B$ is an equivalence relation.
- (Eq.4) Let $\sigma \in \Lambda^o \restriction \Gamma$ and $\vec{P} \subseteq \Lambda^o$. Suppose $M_\sigma\vec{P}\Downarrow$. Then $M_\sigma\Downarrow$, which implies, together with our supposition $\sigma \in \Lambda^o \restriction \Gamma$, that $\sigma \in \Lambda^o \restriction (\Gamma, M\downarrow)$. Hence, $M_\sigma \sim^B N_\sigma$ *i.e.* $N_\sigma\vec{P}\Downarrow$. The other direction is entirely similar.
- (Cong) and (ξ) follow from the fact that $\sim^B$ is a congruence.
- (β) follows from $\lambda\ell$ being a λ-theory.
- (Cond-η) follows from: for $A \in \Lambda^o.\lambda\beta \vdash A\Downarrow \Rightarrow \lambda x.Ax = A$, for x not free in A.

□

Consider the example $M \equiv xx, N \equiv x(\lambda y.xy)$. It is an easy exercise to see that $M \sim^B N$ and $\lambda_L \vdash M \simeq N$. Note that $\lambda\beta \nvdash M = N$. Hence, even restricting to strong-equations, λ_L is more expressive than $\lambda\beta$.

We claim that λ_L is *not* a pre-lazy λ-theory. In particular, $\lambda_L \nvdash \Delta_2\Delta_2 \simeq \Delta_3\Delta_3$ where $\Delta_n \equiv \lambda x.\underbrace{x\cdots x}_{n}$. It therefore follows that λ_L is *not* complete with respect to $\lambda\ell$.

5.3 Towards a Category-Theoretic Semantics

Typed Theories and Categories

It is well-known that there is a tight correspondence between Cartesian closed

categories on the one hand, and *theories* of typed $\lambda\beta\eta$-calculus with surjective pairing on the other, as expounded clearly in Part I of Lambek and Scotts' book [LS86]; see also [Lam80].

Correspondence bet. CCC's and Typed $\beta\eta$ Theories	
Categories	**Theories**
objects	types
structures	type constructors
universal properties	axioms

From Typed Theories to Type-Free Theories

Scott observed that given an atomic type D whose function space $[D \to D]$ is a *retract* of itself (which, when couched in category-theoretic terms, means that D is a *reflexive object*), there is a well-defined translation of any untyped λ-term M into a typed λ-term M' of type D such that all free variables of M' (which are precisely those of M) are of type D.

The important corollary is that given an interpretation of any typed $\lambda\beta$-theory, there is a natural *relative interpretation* of the type-free $\lambda\beta$-theory. It is interesting to note that whereas the interpretation of the *typed* theory involves the *entire* Cartesian closed category, the relative interpretation of the untyped theory entails only *a* reflexive object.[6] Indeed, one only needs to consider the Cartesian closed subcategory freely generated by D.[7]

Models of the Untyped Theory and Categories

λ-models are λ-algebras which satisfy the weak extensionality axiom (see Chapter 3). λ-algebras are, in essence, just type-free *theories* of $\lambda\beta$. Since there is already[8]

[6]This corroborates the view that the typed calculus is prior to the untyped calculus. However, this view is not entirely satisfactory, at least from a computational perspective. It is well-known from the work of Church and Kleene that all (partial) recursive functions can be captured in the type-free calculus (see Chapter 1 for a restatement of the computational power of the type-free λ-calculus *in the lazy regime*); even augmented with recursion constants, typed calculi fail to capture all the recursive functions. [FLO83] gives a highly readable account of the class of numeric functions definable in two typed λ-calculi. The *simply typed calculus* only defines the *elementary* functions. The *second order calculus* defines a gigantic hierarchy of functions so rapidly increasing that Peano arithmetic alone cannot prove that they are total.

[7]In his PhD thesis [Koy84, Chapter 2], Koymans addresses the issue of "how much" of a Cartesian closed category really impinges on the interpretation of a type-free theory (and model). This led him to consider not full-fledged categories in their generality as such, but *monoids* — Cartesian closed monoids to be precise.

[8]With the benefit of hindsight, it is a satisfying exercise to attempt to reconstruct a "history" of how one idea leads (or should lead) to another. It is not entirely clear to me whether the links between models and categories which occurred to Scott follow the sequence I have presented.

a correspondence between reflexive objects of a CCC and type-free theories, the correspondence may be extended to one between models and categories provided the weak extensionality axiom may be pinned down in category-theoretic terms in the shape of some local concreteness property. This has indeed been done by Koymans and the axiom corresponds to the reflexive object *having enough points.* In point of fact, such constructions give rise to *all* possible λ-models; see [Bar84, Chapter 5] and [Koy84] for a comprehensive account.

We have established in section 2 of this Chapter that there is a correspondence between partial Cartesian closed *categories* on the one hand, and *theories* of $\lambda_{L,\mathsf{C}}$ on the other. In this section, we will make some progress towards specializing this correspondence to one between *lazy* $\lambda\mathsf{C}$*-models* and *lazy reflexive objects* which have enough points in *partial Cartesian closed dominical categories.*

5.3.1 Lazy $\lambda\mathsf{C}$-Models and Partial Cartesian Closed Dominical Categories

In Chapter 4, we showed that convergence testing C is not λ-definable. The complementary role C plays in the lazy regime should be evident by now. We formalize the notion of a lazy λC-model.

DEFINITION **5.3.1.1** A *lazy* $\lambda\mathsf{C}$*-model (algebra)* $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle$ is a lazy λ-model (algebra) in which convergence testing is definable.

To recapitulate, this means that there exists a distinguished element $c \in A$ such that $\forall x \in A$, $\mathcal{A}$ satisfies:

- $c\Downarrow$,
- $x\Downarrow \Rightarrow cx = \mathbf{I}$,
- $x\Uparrow \Rightarrow cx\Uparrow$.

LEMMA **5.3.1.2** *Let $\mathcal{A}$ be a lazy $\lambda\mathsf{C}$-model. Then,*

$$\forall M \in \Lambda(\mathsf{C})^{o}.M\Downarrow_{\mathsf{C}} \Rightarrow \mathcal{A} \models M\Downarrow.$$

Hence, $\mathcal{A} \models \lambda\beta\mathsf{C}$ (Refer to Definition 4.4.2.1), i.e.

$$\lambda\beta\mathsf{C} \vdash M = N \Rightarrow \mathcal{A} \models M = N.$$

PROOF Suppose $M\Downarrow_{\mathsf{C}}$. Consider the four inference rules defining $\Downarrow_{\mathsf{C}}$ in Definition 4.4.1.2. We prove $\mathcal{A} \models M\Downarrow$ by establishing a stronger statement: for $M, N \in \Lambda(\mathsf{C})^{o}$,

$$M \Downarrow_{\mathsf{C}} N \Rightarrow \mathcal{A} \models M = N;$$

the result then follows by observing that $\mathcal{A} \models \mathsf{C}\Downarrow \& \lambda x.P\Downarrow$. The stronger statement is proved as follows: the base cases — ($\mathsf{C}\Downarrow_{\mathsf{C}}1$) and ($\mathrm{abs}\Downarrow_{\mathsf{C}}$) — are obvious; ($\beta_N$) is valid since $\mathcal{A}$ models $\lambda\beta$. Finally, ($\mathsf{C}\Downarrow_{\mathsf{C}}2$) is valid by an easy induction argument. □

We introduce the category-theoretic entities that model lazy λC-models.

DEFINITION **5.3.1.3** (i) Let $\mathbb{C} = \mathcal{M}\text{-Ptl}(\mathbb{A})$ be a category of partial morphisms in a domain $\mathcal{M}$ which has partial function space. An object A is *lazy reflexive* if

$$[A_\perp \rightharpoonup A] \lhd_t A,$$

i.e. $[A_\perp \rightharpoonup A]$ is a total retract into A. A is a *strict reflexive object* if

$$[A \rightharpoonup A] \lhd_t A.$$

(ii) A category $\mathbb{C} = \mathcal{M}\text{-Ptl}(\mathbb{A})$ of partial morphisms in a domain $\mathcal{M}$ is a *partial Cartesian closed dominical category* (pCCDC) if $\mathbb{C}$ is dominical, has an atomic 1-element object and partial function space.

(iii) Let $\mathbb{C}$ be a pCCDC. An object A of $\mathbb{C}$ *has enough points* if

$$\forall f, g : A_\perp \rightharpoonup A.[\forall x : 1 \rightharpoonup A. f \circ \overline{x} = g \circ \overline{x}] \Rightarrow f = g.$$

LEMMA **5.3.1.4** *If a pCCDC $\mathbb{C}$ has enough points, than all objects of $\mathbb{C}$ have enough points.*

PROOF Since 1 is atomic, by Definition 5.1.6.3(ii),

$$\mathbb{C}(1, A_\perp) = \{\, \overline{x} : x \in \mathbb{C}(1, A) \,\} \cup \{\, 0_{1,A_\perp} \,\}.$$

Suppose, for some $f, g : A_\perp \rightharpoonup A$,

$$\forall x : 1 \rightharpoonup A. f \circ \overline{x} = g \circ \overline{x}.$$

By definition of zero morphism, $f \circ 0_{1,A_\perp} = g \circ 0_{1,A_\perp} = 0_{1,A}$. Then, $\mathbb{C}(1, f)$ is equal to $\mathbb{C}(1, g)$ as *functions* from the sets $\mathbb{C}(1, A_\perp)$ to $\mathbb{C}(1, A)$ which preserve zero morphisms. Since $\mathbb{C}$ has enough points, $f = g$; whence, A has enough points. □

The main Theorem, which is the focus of this section, is stated as follows:

THEOREM **5.3.1.5** *Lazy reflexive objects with enough points in partial Cartesian closed dominical categories give rise to lazy λC-models.*

5.3.2 Partial Categories Semantics

The rest of this subsection will be devoted to the proof of the Theorem. First, a technical Lemma.

LEMMA **5.3.2.1** *(i) Let* $\mathbb{C}$ *be a category of partial morphisms which has partial function space. Suppose* $U \xrightarrow{f} V \xrightarrow{g} W$ *and that* f *is total. Then,*

$$\overline{g \circ f} = \overline{g} \circ f.$$

(ii) Let $\mathbb{C} = \mathcal{M}\text{-Ptl}(\mathbb{A})$ *be a category of partial morphisms and that* $\mathbb{A}$ *has categorical products. Suppose* $f : U \rightharpoonup V$ *and that* $g : V \rightharpoonup W$ *and* $h : V \rightharpoonup X$. *Then,*

$$\langle g \circ f, h \circ f \rangle = \langle g, h \rangle \circ f.$$

PROOF (i): $\overline{g} \circ f : U \to W_\perp$ is total and $\text{open}_W \circ (\overline{g} \circ f) = g \circ f$. Uniqueness of $\overline{g \circ f}$ then implies $\overline{g \circ f} = \overline{g} \circ f$.
(ii): $\mathbb{C}$ is a p-category. Observe that $\langle g, h \rangle = g \times h \circ \Delta$. Then,

$$\begin{aligned}
\langle g, h \rangle \circ f &= (g \times h) \circ \Delta \circ f && \text{Naturality of } \Delta \\
&= (g \times h) \circ (f \times f) \circ \Delta \\
&= ((g \circ f) \times (h \circ f)) \circ \Delta \\
&= \langle g \circ f, h \circ f \rangle.
\end{aligned}$$

□

NOTATION **5.3.2.2** Let $\mathbb{C} = \mathcal{M}\text{-Ptl}(\mathbb{A})$ be a pCCDC and let U, V range over objects of $\mathbb{C}$. Let $\Delta \equiv x_1, \cdots, x_n$ be a sequence of distinct variables. Write $U^\Delta \equiv U^{|\Delta|} (\equiv U^n)$. Define $U^0 \stackrel{\text{def}}{=} 1$ and inductively, $U^{n+1} \stackrel{\text{def}}{=} U^n \times U$.

(i) Let $\Pi^\Delta_{x_i} : U^\Delta \to U$ be the canonical projection on the i-th coordinate.

(ii) Let $\Gamma \equiv y_1, \cdots, y_m$ with $\{\vec{y}\} \subseteq \{\vec{x}\}$. Define

$$\Pi^\Delta_\Gamma \stackrel{\text{def}}{=} \langle \Pi^\Delta_{y_1}, \cdots, \Pi^\Delta_{y_m} \rangle : U^\Delta \to U^\Gamma;$$

which is the canonical *thinning*. Note that Π^Δ_Γ is a total map.

(iii) If $f_1, \cdots, f_n : U \rightharpoonup V$, then $[f_1, \cdots, f_n] : U \rightharpoonup V^n$ is defined as

$$\begin{aligned}
[] &\stackrel{\text{def}}{=} !_U, \\
[f_1, \cdots, f_{n+1}] &\stackrel{\text{def}}{=} \langle [f_1, \cdots, f_n], f_{n+1} \rangle \text{ for } n \geqslant 1.
\end{aligned}$$

If all f_i's are total, then

$$(*) \qquad \Pi_\Gamma^\Delta \circ [f_1, \cdots, f_n] = f_i;$$

Note that $(*)$ is not necessarily true if the precondition — all f_i being total — is not satisfied. This is a manifestation of the lack of naturality in *both* arguments of the projections. Herein lies the spot where a straightforward extension of the interpretation of λ-terms in a total category to an interpretation in a partial category falls foul of: an ill-defined substitution.

Let $\mathbb{C}$ be a pCCC. For any object A, we define $A_\perp \stackrel{\text{def}}{=} [1 \rightharpoonup A]$ and

$$\begin{aligned} \text{open}_A &\stackrel{\text{def}}{=} \text{peval}_{1,A} \circ \langle \text{id}_{[1\rightharpoonup A]}, !_{[1\rightharpoonup A]} \rangle : A_\perp \rightharpoonup A; \\ \text{lf}_A &\stackrel{\text{def}}{=} \text{p}\Lambda_{A,1,A}(p_{A,1}) : A \to A_\perp. \end{aligned}$$

It is straightforward to verify that $\text{open}_A \circ \text{lf}_A = \text{id}_A$. That is to say, there is, in general, a *partial retract* of A into $A_\perp$. A is said to be *complete* if there is a total retract of A into $A_\perp$; however, we will not need this condition.

LEMMA **5.3.2.3** *Let $\mathbb{C}$ be a pCCC and $A \in Obj(\mathbb{C})$. There is a canonical* total *retract of $[A \rightharpoonup A]$ ("strict" function space) into $[A_\perp \rightharpoonup A]$ ("lazy" function space), i.e.*

$$\langle r, s \rangle : [A \rightharpoonup A] \lhd_{\text{t}} [A_\perp \rightharpoonup A].$$

PROOF Define $r : [A \rightharpoonup A] \to [A_\perp \rightharpoonup A]$ by

$$r \stackrel{\text{def}}{=} \text{p}\Lambda_{[A\rightharpoonup A],A_\perp,A}(\text{peval}_{A,A} \circ \text{id}_{[A\rightharpoonup A]} \times \text{open}_A);$$

and $s : [A_\perp \rightharpoonup A] \to [A \rightharpoonup A]$ by

$$s \stackrel{\text{def}}{=} \text{p}\Lambda_{[A_\perp\rightharpoonup A],A,A}(\text{peval}_{A_\perp,A} \circ \text{id}_{[A_\perp\rightharpoonup A]} \times \text{lf}_A).$$

Then $s \circ r = \text{id}_{[A\rightharpoonup A]}$ follows from an application of Lemma 5.2.3.7. □

COROLLARY **5.3.2.4** *Let $\mathbb{C}$ be a pCCC. If A is a lazy reflexive object with embedding pair $\langle \text{Gr}, \text{Fun} \rangle$, then A is a strict reflexive object via*

$$\langle \text{Gr}^v, \text{Fun}^v \rangle : [A \rightharpoonup A] \lhd^{\text{t}} A;$$

where $\text{Fun}^v \stackrel{\text{def}}{=} s \circ \text{Fun}, \text{Gr}^v \stackrel{\text{def}}{=} \text{Gr} \circ r$, and r and s are defined as in the Lemma.

PROOF Straightforward. □

Partial Categories Semantics of Lazy λC-Models

DEFINITION **5.3.2.5** Let $\mathbb{C}$ be a pCCDC and A a lazy reflexive object of $\mathbb{C}$ with

$$A \xrightarrow{\mathsf{Fun}} [A_\perp \rightharpoonup A] \xrightarrow{\mathsf{Gr}} A.$$

Define Fun^ν and Gr^ν as in the preceding Corollary. Denote

$$\begin{aligned} \mathsf{Ap} &\stackrel{\text{def}}{=} \mathrm{peval}_{A_\perp,A} \circ (\mathsf{Fun} \times \mathrm{id}_{A_\perp}), \\ c &\stackrel{\text{def}}{=} \mathsf{Gr}^\nu \circ \mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \in \mathbb{C}(1,A) \\ &\quad \text{where } i \stackrel{\text{def}}{=} \mathsf{Gr} \circ \mathrm{p}\Lambda_{1,A_\perp,A}(\mathrm{open}_A \circ q_{1,A_\perp}). \end{aligned}$$

We will see later that i above is just the denotation of the λ-term $\mathbf{I}$.

Let Δ be a sequence of distinct variables.

(i) Define, for each Δ, $\cdot_{A_\perp^\Delta} : \mathbb{C}(A_\perp^\Delta, A) \times \mathbb{C}(A_\perp^\Delta, A) \to \mathbb{C}(A_\perp^\Delta, A)$ by

$$f \cdot_{A_\perp^\Delta} g \stackrel{\text{def}}{=} \mathsf{Ap} \circ \langle f, \overline{g} \rangle.$$

(ii) Let $f \in \mathbb{C}(1,A)$, $M \in \Lambda(\mathbb{C}, \underline{\mathbb{C}(1,A)})$ such that $\Delta \supseteq \mathsf{FV}(M)$. Define, by structural induction, partial morphisms $[\![\Delta : M]\!] : A_\perp^\Delta \rightharpoonup A$ as follows:

$$\begin{aligned} [\![\Delta : \underline{f}]\!] &\stackrel{\text{def}}{=} f \circ !_{A_\perp^\Delta}, \\ [\![\Delta : \mathsf{C}]\!] &\stackrel{\text{def}}{=} c \circ !_{A_\perp^\Delta}, \\ [\![\Delta : x]\!] &\stackrel{\text{def}}{=} \mathrm{open}_A \circ \Pi_x^\Delta, \\ [\![\Delta : MN]\!] &\stackrel{\text{def}}{=} [\![\Delta : M]\!] \cdot_{A_\perp^\Delta} [\![\Delta : N]\!], \\ [\![\Delta : \lambda y.M]\!] &\stackrel{\text{def}}{=} \mathsf{Gr} \circ \mathrm{p}\Lambda_{A_\perp^\Delta, A_\perp, A}([\![\Delta, y : M]\!]). \end{aligned}$$

Observe that i defined earlier in the Definition is just $[\![\mathbf{I}]\!] \stackrel{\text{def}}{=} [\![\emptyset : \mathbf{I}]\!]$.

(iii) For a valuation $\rho : \mathsf{Var} \to \mathbb{C}(1,A)$, define $\rho^\Delta \stackrel{\text{def}}{=} \langle \overline{\rho(x_1)}, \cdots, \overline{\rho(x_n)} \rangle$ and $[\![M]\!]_\rho \stackrel{\text{def}}{=} [\![\Delta : M]\!] \circ \rho^\Delta$ with $\Delta \equiv \mathsf{FV}(M)$. Clearly, $[\![M]\!]_\rho \in \mathbb{C}(1,A)$.

(iv) $\mathcal{A}(\mathbb{C},A)$ is the structure $\langle \mathbb{C}(1,A), \cdot, 0_{1,A}, [\![-]\!]_- \rangle$. For $f, g \in \mathbb{C}(1,A)$, we write $f\Uparrow \stackrel{\text{def}}{=} [f = 0_{1,A}]$ and $f\Downarrow \stackrel{\text{def}}{=} \neg(f\Uparrow)$ and define

$$f \cdot g \stackrel{\text{def}}{=} f \cdot_1 g.$$

REMARK **5.3.2.6** (i) Since 1 is the *atomic* 1-element object in $\mathbb{C}$, the only morphism in $\mathbb{C}(1,A)$ which is not total is $0_{1,A}$.

(ii) $\langle \mathbb{C}(1,A), \cdot, 0_{1,A} \rangle$ thus defined is a q-aswd with $\Uparrow = \{ 0_{1,A} \}$. Left-strictness axiom is easily seen to hold since $\langle 0, f \rangle = 0$ and that zero maps are stable under composition.

OPEN QUESTION **5.3.2.7** Is c thus defined the only convergence testing there is?

The following Lemma is crucial to the proof of $\mathcal{A}(\mathbb{C}, A) \models (\beta)$.

LEMMA **5.3.2.8 (Substitution 2)** *(i) Let* $\Delta \supseteq \Gamma \supseteq \mathsf{FV}(M)$. *Then,*

$$[\![\Delta : M]\!] = [\![\Gamma : M]\!] \circ \Pi^{\Delta}_{\Gamma}.$$

(ii) Let $\Delta = \vec{x} \supseteq \mathsf{FV}(M), |\vec{N}| = |\vec{x}|$ *and* $\Gamma \supseteq \mathsf{FV}(\vec{N})$. *Then,*

$$[\![\Gamma : M[\vec{x} := \vec{N}]]\!] = [\![\Delta : M]\!] \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle.$$

(iii) Let $\Delta \supseteq \mathsf{FV}(\lambda x.M), \Gamma \supseteq \mathsf{FV}((\lambda x.M)N)$ *and* $\Gamma \supseteq \Delta$. *Then,*

$$[\![\Gamma : M[x := N]]\!] = [\![\Delta, x : M]\!] \circ \langle \Pi^{\Gamma}_{\Delta}, \overline{[\![\Gamma : N]\!]} \rangle.$$

PROOF (i) By structural induction. The base cases are easily seen to be valid. We consider the two inductive cases.

$$\begin{aligned}
[\![\Delta : PQ]\!] &= \mathsf{Ap} \circ \langle [\![\Delta : P]\!], \overline{[\![\Delta : Q]\!]} \rangle && \text{Ind. Hyp.} \\
&= \mathsf{Ap} \circ \langle [\![\Gamma : P]\!] \circ \Pi^{\Delta}_{\Gamma}, \overline{[\![\Gamma : Q]\!] \circ \Pi^{\Delta}_{\Gamma}} \rangle && \text{Lemma 5.3.2.1} \\
&= \mathsf{Ap} \circ \langle [\![\Gamma : P]\!], \overline{[\![\Gamma : Q]\!]} \rangle \circ \Pi^{\Delta}_{\Gamma} \\
&= [\![\Gamma : PQ]\!] \circ \Pi^{\Delta}_{\Gamma}.
\end{aligned}$$

$$\begin{aligned}
[\![\Gamma : \lambda x.P]\!] \circ \Pi^{\Delta}_{\Gamma} &= \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Gamma, x : P]\!]) \circ \Pi^{\Delta}_{\Gamma} && \text{Lemma 5.2.3.7} \\
&= \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Gamma, x : P]\!] \circ \Pi^{\Delta}_{\Gamma} \times \mathsf{id}_{A_\perp}) \\
&= \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Gamma, x : P]\!] \circ \Pi^{\Delta, x}_{\Gamma, x}) && \text{Ind. Hyp.} \\
&= \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Delta, x : P]\!]) \\
&= [\![\Delta : \lambda x.P]\!].
\end{aligned}$$

(ii) Again, we consider the inductive cases only. First, note that

$$
\begin{aligned}
& \langle \overline{[\![\Gamma, y : N_1]\!]}, \cdots, \overline{[\![\Gamma, y : N_n]\!]}, \overline{[\![\Gamma, y : y]\!]} \rangle \\
= \; & \langle \overline{[\![\Gamma : N_1]\!]} \circ \Pi_{\Gamma}^{\Gamma,y}, \cdots, \overline{[\![\Gamma : N_n]\!]} \circ \Pi_{\Gamma}^{\Gamma,y}, \Pi_{y}^{\Gamma,y} \rangle \quad \text{(i)} \\
= \; & \langle \overline{[\![\Gamma : N_1]\!]} \circ p, \cdots, \overline{[\![\Gamma : N_n]\!]} \circ p, \mathsf{id}_{A_\perp} \circ q \rangle \\
= \; & \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle \times \mathsf{id}_{A_\perp} \circ \langle p, q \rangle \\
= \; & \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle \times \mathsf{id}_{A_\perp}.
\end{aligned}
$$

$$
\begin{aligned}
& [\![\Gamma : (\lambda y.P)[\vec{x} := \vec{N}]]\!] \\
= \; & [\![\Gamma : \lambda y.P[\vec{x}, y := \vec{N}, y]]\!] \\
= \; & \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Gamma, y : P[\vec{x}, y : \vec{N}, y]]\!]) && \text{Ind. Hyp.} \\
= \; & \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Delta, y : P]\!] \circ \langle \overline{[\![\Gamma, y : N_1]\!]}, \cdots, \overline{[\![\Gamma, y : N_n]\!]}, \overline{[\![\Gamma, y : y]\!]} \rangle) && \text{Note above} \\
= \; & \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Delta, y : P]\!] \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle \times \mathsf{id}_{A_\perp}) && \text{Lemma 5.2.3.7} \\
= \; & \mathsf{Gr} \circ \mathsf{p}\Lambda([\![\Delta, y : P]\!]) \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle \\
= \; & [\![\Delta : \lambda y.P]\!] \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle.
\end{aligned}
$$

$$
\begin{aligned}
& [\![\Gamma : PQ[\vec{x} := \vec{N}]]\!] \\
= \; & [\![\Gamma : (P[\vec{x} := \vec{N}])(Q[\vec{x} := \vec{N}])]\!] \\
= \; & \mathsf{Ap} \circ \langle [\![\Gamma : P[\vec{x} := \vec{N}]]\!], \overline{[\![\Gamma : Q[\vec{x} := \vec{N}]]\!]} \rangle && \text{Ind. Hyp.} \\
= \; & \mathsf{Ap} \circ \langle [\![\Delta : P]\!] \circ \langle \overline{[\![\Gamma : N_i]\!]} : i \rangle, \overline{[\![\Delta : Q]\!] \circ \langle \overline{[\![\Gamma : N_i]\!]} : i \rangle} \rangle && \text{Lemma 5.3.2.1} \\
= \; & \mathsf{Ap} \circ \langle [\![\Delta : P]\!], \overline{[\![\Delta : Q]\!]} \rangle \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle \\
= \; & [\![\Delta : PQ]\!] \circ \langle \overline{[\![\Gamma : N_1]\!]}, \cdots, \overline{[\![\Gamma : N_n]\!]} \rangle.
\end{aligned}
$$

(iii) Apply (ii) to $\Delta' \equiv \Delta, x$ and Γ, writing $\Delta \equiv \vec{y}$ and $M[x := N] \equiv M[\vec{y}, x := \vec{y}, N]$.

□

PROPOSITION **5.3.2.9** *Let* $M, N \in \Lambda(\mathsf{C}, \underline{\mathsf{C}(1, A)})$ *and* $\Delta \supseteq \mathsf{FV}(MN)$. *Then,*

(i) convergence testing is definable in $\langle \mathcal{A}(\mathsf{C}, A), \cdot, 0_{1,A}, [\![-]\!] \rangle$;

(ii) $\lambda\beta \vdash M = N \Rightarrow [\![\Delta : M]\!] = [\![\Delta : N]\!] \Rightarrow \mathcal{A}(\mathsf{C}, A) \models M = N$.

Hence, $\mathcal{A}(\mathsf{C}, A)$ *is a lazy* $\lambda\mathsf{C}$*-algebra.*

PROOF (i) Clearly, $c\Downarrow$, *i.e.* $c \neq 0_{1,A}$. By a straightforward application of Lemma 5.2.3.7, we remark that

$$\begin{aligned}
& r \circ \mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \\
= \; & \mathrm{p}\Lambda_{1,A_\perp,A}((\mathrm{peval}_{A,A} \circ (\mathrm{id}_{[A\to A]} \times \mathrm{open}_A)) \circ (\mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \times \mathrm{id}_{A_\perp})) \\
= \; & \mathrm{p}\Lambda_{1,A_\perp,A}(\mathrm{peval}_{A,A} \circ (\mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \times \mathrm{open}_A)).
\end{aligned}$$

Let $f \in \mathbb{C}(1, A)$. Then,

$$\begin{aligned}
& c \cdot f \\
= \; & \mathrm{peval}_{A_\perp,A} \circ \mathsf{Fun} \times \mathrm{id}_{A_\perp} \circ \langle c, \overline{f} \rangle \\
= \; & \mathrm{peval}_{A_\perp,A} \circ \langle r \circ \mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}), \overline{f} \rangle \qquad \text{remark above} \\
= \; & \mathrm{peval}_{A_\perp,A} \circ [\mathrm{p}\Lambda_{1,A_\perp,A}(\mathrm{peval}_{A,A} \circ (\mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \times \mathrm{open}_A))] \times \mathrm{id}_{A_\perp} \\
& \circ \langle \mathrm{id}_1, \overline{f} \rangle \\
= \; & \mathrm{peval}_{A,A} \circ (\mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \times \mathrm{open}_A) \circ \langle \mathrm{id}_1, \overline{f} \rangle \\
= \; & \mathrm{peval}_{A,A} \circ (\mathrm{p}\Lambda_{1,A,A}(i \circ p_{1,A}) \times \mathrm{id}_A) \circ \langle \mathrm{id}_1, f \rangle \\
= \; & i \circ p_{1,A} \circ \langle \mathrm{id}_1, f \rangle \\
= \; & i \circ p_{1,A} \circ (\mathrm{id}_1 \times f) \circ \Delta_1 \\
= \; & i \circ \mathrm{dom}(f).
\end{aligned}$$

Since 1 is atomic, we conclude that

$$c \cdot f = \begin{cases} [\![\emptyset : \mathbf{I}]\!] = i & \text{if } f\Downarrow \iff \mathrm{dom}(f) = \mathrm{id}_1; \\ 0_{1,A} & \text{else.} \end{cases}$$

(ii) By induction on the length of proof of $M = N$. We treat only the axioms (β) and (ξ)[9].
Axiom (β):

$$\begin{aligned}
& [\![\Delta : (\lambda x.P)Q]\!] \\
= \; & \mathsf{Ap} \circ \langle \mathsf{Gr} \circ \mathrm{p}\Lambda([\![\Delta, x : P]\!]), \overline{[\![\Delta : Q]\!]} \rangle && \mathsf{Fun} \circ \mathsf{Gr} = \mathrm{id}_{[A_\perp \to A]} \\
= \; & \mathrm{peval} \circ \langle \mathsf{Fun} \circ \mathsf{Gr} \circ \mathrm{p}\Lambda([\![\Delta, x : P]\!]), \overline{[\![\Delta : Q]\!]} \rangle \\
= \; & \mathrm{peval} \circ \mathrm{p}\Lambda([\![\Delta, x : P]\!]) \times \mathrm{id}_{A_\perp} \circ \langle \mathrm{id}_{A_\perp^\Delta}, \overline{[\![\Delta : Q]\!]} \rangle \\
= \; & [\![\Delta, x : P]\!] \circ \langle \mathrm{id}_{A_\perp^\Delta}, \overline{[\![\Delta : Q]\!]} \rangle && \text{Lemma 5.3.2.8(iii)} \\
= \; & [\![\Delta : P[x := Q]]\!].
\end{aligned}$$

[9]This is a proof rule in $\lambda\beta$, not to be confused with weak extensionality axiom.

Rule (ξ): *i.e.* $P = Q \Rightarrow \lambda x.P = \lambda x.Q$.
Now, $[\![\Delta : P]\!] = [\![\Delta : Q]\!]$ by hypothesis. Then,

$$\begin{aligned} & [\![\Delta : P]\!] \circ \Pi_{\Delta}^{\Delta,x} = [\![\Delta : Q]\!] \circ \Pi_{\Delta}^{\Delta,x} \\ \Rightarrow \quad & [\![\Delta, x : P]\!] = [\![\Delta, x : Q]\!] \\ \Rightarrow \quad & [\![\Delta : \lambda x.P]\!] = [\![\Delta : \lambda x.Q]\!]. \end{aligned}$$

To show that $\mathcal{A}(\mathbb{C}, A)$ is a lazy λC-algebra, it remains to show that the denotation of any abstraction term is a total morphism. This is guaranteed by definition, since Gr and $\mathrm{p}\Lambda(f)$ for any morphism f are total, hence, so is their composition. □

The following Proposition characterizes the category-theoretic condition that corresponds to the weak extensionality axiom.

PROPOSITION **5.3.2.10** *Let $\mathcal{A} = \langle A, \cdot, 0_{1,A}, [\![-]\!]_- \rangle$ be as before. Then, A has enough points iff $\mathcal{A}$ is a lazy λC-model.*

PROOF Let $\Delta \supseteq \mathrm{FV}(M)$. First observe that

$$\begin{aligned} [\![\Delta : \mathbf{1}M]\!] &= [\![\Delta : \lambda y.My]\!] \\ &= \mathsf{Gr} \circ \mathrm{p}\Lambda(\mathsf{peval} \circ \langle \mathsf{Fun} \circ [\![\Delta, y : M]\!], \overline{[\![\Delta, y : y]\!]} \rangle) \\ &= \mathsf{Gr} \circ \mathrm{p}\Lambda(\mathsf{peval} \circ \langle \mathsf{Fun} \circ [\![\Delta : M]\!] \circ \Pi_{\Delta}^{\Delta,y}, \Pi_{y}^{\Delta,y} \rangle) \\ &= \mathsf{Gr} \circ \mathrm{p}\Lambda(\mathsf{peval} \circ (\mathsf{Fun} \circ [\![\Delta : M]\!]) \times \mathsf{id}_{A_\perp}). \end{aligned}$$

"⇒": Suppose A has enough points, then it is easy to see that $1 \times A$ also has enough points. For any $f, g \in \mathbb{C}(1, A)$, $\forall x \in \mathbb{C}(1, A)$,

$$\begin{aligned} & \underline{f}\underline{x} = \underline{g}\underline{x} \\ \Rightarrow \quad & \mathsf{peval} \circ \langle \mathsf{Fun} \circ f, \overline{x} \rangle = \mathsf{peval} \circ \langle \mathsf{Fun} \circ g, \overline{x} \rangle \\ \Rightarrow \quad & \mathsf{peval} \circ ((\mathsf{Fun} \circ f) \times \mathsf{id}_{A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle = \mathsf{peval} \circ ((\mathsf{Fun} \circ g) \times \mathsf{id}_{A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle. \end{aligned}$$

Since $1 \times A$ has enough points, we have

$$\mathsf{peval} \circ (\mathsf{Fun} \circ f) \times \mathsf{id}_{A_\perp} = \mathsf{peval} \circ (\mathsf{Fun} \circ g) \times \mathsf{id}_{A_\perp}.$$

By an appeal to the preceding observation, we conclude that $\mathbf{1}\underline{f} = \mathbf{1}\underline{g}$. Hence, $\mathcal{A}$ is a lazy λC-model.
"⇐": Suppose $\mathcal{A}$ is a lazy λC-model and let $f, g : A_\perp \rightharpoonup A$. Then,

$$\begin{aligned} & \forall x : 1 \rightharpoonup A. f \circ \overline{x} = g \circ \overline{x} \\ \Rightarrow \quad & \forall x : 1 \rightharpoonup A. (f \circ q_{1,A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle = (g \circ q_{1,A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle \end{aligned}$$

Hence, $\mathsf{peval} \circ (\mathrm{p}\Lambda(f \circ q_{1,A_\perp}) \times \mathsf{id}_{A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle = \mathsf{peval} \circ (\mathrm{p}\Lambda(g \circ q_{1,A_\perp}) \times \mathsf{id}_{A_\perp}) \circ \langle \mathsf{id}_1, \overline{x} \rangle$; and so, $\mathsf{peval} \circ \mathsf{Fun} \times \mathsf{id}_{A_\perp} \circ \langle \mathsf{Gr} \circ \mathrm{p}\Lambda(f \circ q_{1,A_\perp}), \overline{x} \rangle = \mathsf{peval} \circ \mathsf{Fun} \times \mathsf{id}_{A_\perp} \circ \langle \mathsf{Gr} \circ \mathrm{p}\Lambda(g \circ q_{1,A_\perp}), \overline{x} \rangle$. This means

$$\forall x \in \mathbb{C}(1, A). f' \cdot x = g' \cdot x$$

where $f' = \mathsf{Gr} \circ \mathrm{p}\Lambda(f \circ q_{1,A_\perp})$, similarly for g'. Now, since $\mathcal{A}$ is a lazy λC-model, and so, satisfies weak extensionality axiom, we have, $\mathbf{1}\underline{f'} = \mathbf{1}\underline{g'}$. Then, by the earlier observation and since $\mathsf{Fun} \circ \mathsf{Gr} = \mathsf{id}$ and for any morphism h, we have $h = \mathrm{peval} \circ \mathrm{p}\Lambda(h) \times \mathsf{id}$, we conclude that

$$\mathsf{Gr} \circ \mathrm{p}\Lambda(f \circ q_{1,A_\perp}) = \mathsf{Gr} \circ \mathrm{p}\Lambda(g \circ q_{1,A_\perp}).$$

Then, by the uniqueness of $\mathrm{p}\Lambda(-)$,

$$f \circ q_{1,A_\perp} = g \circ q_{1,A_\perp};$$

from which we deduce $f = g$. □

We have thus concluded the proof of the Theorem.

Chapter 6

Further Directions

In this thesis, we have investigated various foundational issues of the lazy λ-calculus. There are many possibilities for extension of our results. In the following, we will mention in brief some areas for further research.

Local Structure Theorem for D

An obvious open question, and a hard one, is to capture the local structure of D (the initial solution of the domain equation $D \cong [D \to D]_\perp$ in the category of cpo's) syntactically — a classical λ-calculus problem. That is to say, find syntactic preorder $\precsim$ on Λ such that

$$\forall M, N \in \Lambda. M \precsim N \iff D \models M \sqsubseteq N.$$

A related question is to find the λ-theory (which is fully lazy) induced by D. With reference to Theorem 4.6.6.5, the theory will lie somewhere between the λ-theory induced by the lazy PSE-ordering $\lesssim$ (because of Proposition 4.1.3.8) and $\lambda\ell$ (owing to Theorem 3.4.1.3).

At the moment, we know of no (mathematical) cpo-based fully lazy λ-model whose (in)equational theory includes $\lambda\ell$. Note that strict inclusion is not possible because $\lambda\ell$ is the maximal fully lazy λ-model.

Full Abstraction of Lambda Transition Systems

The bisimulation preorder may be characterized as

$$M \precsim^B N \iff M\vec{P}\Downarrow \Rightarrow N\vec{P}\Downarrow.$$

Hence, we can immediately deduce that

$$M \sqsubseteq^B N \Rightarrow \forall P \in \Lambda. MP \sqsubseteq^B NP.$$

To show that the bisimulation is a logical relation (which it is), it remains to show that (right-)application is monotonic w.r.t. $\sqsubseteq^B$ *i.e.*

$$M \sqsubseteq^B N \Rightarrow \forall P \in \Lambda. PM \sqsubseteq^B PN;$$

which turns out to be a highly non-trivial assertion to prove. Abramsky's proof exploits the powerful machinery of the domain logic of lazy λ-calculus. It would be interesting to prove the assertion directly by syntactic means. The technique used in proving that Longo tree preorder is a precongruence (see Chapter 2) might be relevant here.

The problem of constructing a retract of D which is a fully abstract model of $\lambda\ell$ remains open. A closely related question identified in Chapter 4 that remains to be resolved is the relationship between the bisimulation inequational theory (induced by $\sqsubseteq^\omega$) of $\lambda\ell_\omega$ and that of $\lambda\ell$. More precisely, whether it is true that the inequational theory of $\lambda\ell_\omega$ is a *conservative* extension of $\lambda\ell$. Also unresolved is the relationship between $\lambda\ell_\omega$ and $\lambda\ell_c$.

Given a programming language and a non fully abstract denotational model, and suppose the language, rather than the model, is prior, the *restrictive* approach achieves full abstraction by "cutting down" the original model to fit the language according to some operational criteria. Two obvious stipulations that might reasonably be placed on the target fully abstract sub-structure are that it should be a *retract* and/or a *sub-model* of the original model. By sub-model, we mean the existence of a morphism from the sub-structure to the full model that preserves the basic operations of the language — homomorphism in the usual Σ-algebra sense. However, the two requirements *i.e.* retraction and sub-model property, seem to be divergent, even mutually exclusive. This is the case both for the lazy λ-calculus as well as for PCF.

It would be interesting to reformulate Stoughton's construction of the *inductively reachable* fully abstract sub-algebra [Sto88] in the case of lazy λ-calculus; and establish its relationship with fully abstract retracts obtained by the method described in Chapter 4.

Category-Theoretic Characterization

We have not been able to obtain a satisfactory characterization of lazy λ- or λC-models in the framework of partial categories. In our formulation, λ-models are quasi-applicative structures with divergence which are essentially applicative structures with distinguished divergent elements built in and with respect

to which the application operation is left-strict. The logic of existence (or convergence) in lazy λ-models is *classical* — an element in the structure is either divergent or convergent; whereas that in p-categories is essentially *intuitionistic*. For this reason, we have to resort to using *dominical* categories (which some constructivist category-theorists find objectionable) in our attempt to characterize lazy λ-models.

For further research, we intend to consider three categories of entities and show that they are mutually *equivalent*. First, the category of *intuitionistic* (for want of a more descriptive adjective) lazy λ-algebras whose axiom of convergence is cast in an intuitionistic fashion. We seek to relate this category to the category of partial Cartesian closed categories with (lazy) reflexive objects and the category of *partial C-monoids*, using, among others, the technique of *Karoubi construction*. Intuitively, partial C-monoids *cf.* [LS86],[HS86] are an axiomatization of the monoid $\mathbf{C}(U_\perp, U)$ in a partial Cartesian closed category $\mathbf{C}$ where U is a lazy reflexive object. The same program could also be carried out for the category of intuitionistic *call-by-value* λ-algebras.

The Formal λ_L-Calculus and Moggi-Plotkins' λ_P-Calculus

The formal system λ_L-calculus introduced in Chapter 5 is only a beginning in the study of formal systems for proving equivalences between program phrases in the lazy regime. From an operational viewpoint, a programming language is completely determined by specifying a set of programs and prescribing an evaluation mechanism in the shape of a partial function from programs to *values*; in our case, a paradigmatic language $\lambda\ell = \langle \Lambda^\circ, \Downarrow \rangle$ is determined this way. The evaluation mechanism *induces* a notion of operational equivalence — two terms are equivalent if there are no observable differences in their respective behaviours under all program contexts. In the lazy case, Abramsky shows that this notion of equivalence coincides with bisimulation equivalence. Following Plotkin, we accept the induced operational equivalence as given, and look for the corresponding calculi with the criterion for judging a calculus being its *correctness* with respect to the operational equivalence. In this sense, λ_L is correct, but then, there are plenty of such correct calculi.

Another criterion for judging a calculus is whether it is the unique *sound and complete* calculus with respect to a certain class of models. According to this criterion, $\lambda_{L,\mathbf{C}}$ and not λ_L is the right candidate; because the former is sound and complete with respect to interpretation in partial categories.

The formulae of the proof system λ_L are intuitionistic sequents, more precisely, *quasi-strong* equations rather than equations. What is the relationship between

the equational fragment of λ_L and $\lambda\beta$? More specifically, is the equational fragment of λ_L r. e. axiomatizable over $\lambda\beta$? Is there a (semi)-decision procedure for the equational fragment of λ_L and $\lambda_{L,c}$? To answer the last question, we need an associated notion of reduction.

Closely related to the formal λ_L-calculus is Moggi's work on the λ_P-calculus [Mog88b]. In his thesis, Moggi investigates various formal systems for reasoning about partial functions or partial elements in the framework of Beeson's (intuitionistic) logic of partial terms (LPT) [Bee85]. Variants of LPT are introduced for reasoning about partial terms with a restriction operator (LPT + $\downarrow$), monotonic partial functions (monLPT), λ-terms (λ_P-calculus) and λY-terms ($\lambda_p\mu$Y-calculus). He also develops new techniques for proving properties of the associated reduction systems. An area for further research is to recast λ_L in the framework of LPT and establish (further) relationships between λ_L and the various λ_P-calculi. Although there are fundamental differences between LPT and LPE (Scott's Logic of Partial Element, which is the logic of λ_L) in the interpretation of variables — in the former framework, variables range only over *total* elements, Moggi's work in showing that (different flavours) of LPE are *conservative* extensions of LPT makes this representation plausible. We conjecture that $\lambda_{L,c}$ is equivalent to λ_P.

Also worth studying is the formal $\lambda_{L,p}$-calculus which is λ_L augmented with a parallel convergence constant P. We expect this calculus to be related to Moggi's λ_P with restriction operators.

Index

Index

Bibliography

[Abr87] Samson Abramsky.
Domain Theory and the Logic of Observable Properties.
PhD thesis, University of London, 1987.

[Abr88] Samson Abramsky.
The Lazy Lambda Calculus.
In David Turner, editor, *Declarative Programming*, Addison-Wesley, 1988.
To Appear.

[Acz80] Peter Aczel.
Frege Structures and the Notions of Proposition, Truth and Set.
In J. Barwise, H.J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, pages 31–59, North-Holland Publishing Company, 1980.

[AL86] A. Asperti and G. Longo.
Categories of Partial Morphisms and the Relation between Type Structures.
Nota Scientifica S-7-85, Dipartimento di Informatica, Università di Pisa, 1986.
Invited Lecture, Banach Semester 1985.

[Aug84] L. Augustsson.
A Compiler for Lazy ML.
In *ACM Symposium on Lazy and Functional Programming*, pages 218 – 227, 1984.

[Bac78] J. W. Backus.
Can Programming be Liberated from the von Neuman Style?
Comm. ACM, 21:613–641, 1978.

[Bar71] H. P. Barendregt.
Some Extensional Term Models for Combinatory Logics and λ-Calculi.
PhD thesis, University of Utrecht, 1971.

[Bar84] H. Barendregt.
The Lambda Calculus: Its Syntax and Semantics.

North-Holland, revised edition, 1984.

[BC82] G. Berry and P.-L. Curien.
Sequential Algorithms on Concrete Data Structures.
Theoretical Computer Science, 20:265 – 321, 1982.

[BCD83] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini.
A Filter Lambda Model and the Completeness of Type-Assignment.
J. Symbolic Logic, 48:931–940, 1983.

[BCL85] G. Berry, P.-L. Curien, and J.-J. Lévy.
Full Abstraction for Sequential Languages: the State of the Art.
In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132, Cambridge University Press, 1985.

[Bee85] M. J. Beeson.
Foundations of Constructive Mathematics.
Springer-Verlag, 1985.

[Ber78] G. Berry.
Séquentialité de l'èvaluation formelle des λ-expressions.
In *Proc. 3-e Colloque International sur la Programmtion, Paris*, 1978.

[Ber81] G. Berry.
Some Syntactic and Categorical Constructions of Lambda Calculus Models.
Papport de Recherche 80, Institute National de Recherche en Informatique et en Automatique (INRIA), 1981.

[BK80] H. Barendregt and K. Koymans.
Comparing Some Classes of Lambda-Calculus Models.
In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 287–301, Academic Press, 1980.

[BL80] H. Barendregt and G. Longo.
Equality of λ-Terms in the Model T^{ω}.
In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, 1980.

[BL84] Kim Bruce and Giuseppe Longo.
On Combinatory Algebras and their Expansions.
Theoretical Computer Science, 31:31–40, 1984.

[Boh68] C. Böhm.
Alcune Proprieta delle forme $\beta\eta$-normali nel λK-calculus.
Pubblicazioni 696, Instituto per le Applicazioni del Calcolo, Roma, 1968.

[BvL86] H. Barendregt and M. van Leeuwen.
Functional Programming and the Language TALE.
Technical Report 412, University of Utrecht, Dept. of Mathematics, 1986.

[CDHL84] M. Coppo, M. Dezani-Ciancaglini, Furio Honsell, and G. Longo.
Extended Type Structure and Filter Lambda Models.
In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium '82*, pages 241–262, Elsevier Science Publishers B.V. (North-Holland), 1984.

[CDV81] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri.
Functional Characters of Solvable Terms.
Z. Math. Logik. Grundlagen, 27:44–58, 1981.

[CDZ87] Mario Coppo, Mariangiola Dezani-Ciancaglini, and M. Zacchi.
Type Theories, Normal Forms and D_∞ λ-models.
Information and Computation, 72:85–116, 1987.

[CHS72] H. B. Curry, J. R. Hindley, and J. P. Seldin.
Combinatory Logic II.
North-Holland, Amsterdam, 1972.

[Chu36] Alonzo Church.
An Unsolvable Problem in Elementary Number Theory.
Amer. J. Math., 58:345–363, 1936.

[CO88] P.-L. Curien and Adam Obtułowicz.
Partiality and Cartesian Closedness.
Information and Computation, 1988.
To appear.

[CR36] A. Church and J. B. Rosser.
Some Properties of Conversion.
Tran. Amer. Math. Soc., 39:472–482, 1936.

[Dar84] John Darlington.
Functional Programming.
In Chambers and Duce, editors, *Distributed Computing*, pages 57–77, Academic Press, 1984.

[dH88] R. diPaola and A. Heller.
Dominical Categories.
J. Symbolic Logic, 1988.
To appear.

[DM86] Mariangiola Dezani-Ciancaglini and Ines Margaria.
A Characterization of F-Complete Type Assignments.

Theoretical Computer Science, 45:121–157, 1986.

[DR81] John Darlington and Mike Reeves.
ALICE: A Multi-processor Reduction Machine for the Parallel Evaluation of Applicative Languages.
In *Proc. ACM/MIT Conference on Functional Programming Languages and Computer Architecture, 1981*, pages 65–74, 1981.

[Eng81] E. Engeler.
Algebras and Combinators.
Algebra Universalis, 13:389–392, 1981.

[Fai85] J. Fairbairn.
Design and Implementation of a Simple Typed Language based on the Lambda Calculus.
PhD thesis, Univ. of Cambridge, 1985.

[FLO83] Steven Fortune, Daniel Leivant, and Michael O'Dononell.
The Expressiveness of Simple and Second-Order Type Structure.
J. ACM, 30:151–185, 1983.

[Fre03] G. Frege.
Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet.
1903.
Vol. I and II(Jena).

[FW76] D. P. Friedman and D. S. Wise.
CONS Should Not Evaluate its Arguments.
In Michaelson and Milner, editors, *Automata, Languages and Programming*, Edinburgh University Press, 1976.

[GHK*80] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott.
A Compendium of Continuous Lattices.
Springer-Verlag, Berlin, 1980.

[Gor73] M. J. C. Gordon.
Evaluation and Denotation of Pure LISP: a Worked Example in Semantics.
PhD thesis, Univ. of Edinburgh, 1973.

[HdR88] F. Honsell and S. Ronchi della Rocca.
An Approximation Theorem for Scott's Topological Lambda Models and the Topological Incompleteness of Lambda Calculus.
J. Computer and System Science, 1988.
To appear.

[Hen80] P. Henderson.

Functional Programming: Applications and Implementation.
Prentice Hall, 1980.

[HL80] J. R. Hindley and G. Longo.
Lambda Calculus Models and Extensionality.
Z. Logik Grundlag. Math., 26:289–310, 1980.

[HM76] P. Henderson and J. H. Morris.
A Lazy Evaluator.
In *Third ACM Symposium on The Principles of Programming Languages, Atlanta, GA*, 1976.

[HM85] M. C. B. Hennessy and Robin Milner.
Algebraic Laws for Non-determinism and Concurrency.
JACM, 32:137–161, 85.

[HS86] W. S. Hatcher and J. P. Scott.
Lambda Algebras and C-monoids.
Zeitschrift Mathematische Logik, 32:415–430, 1986.

[Hyl76] Martin Hyland.
A Syntactic Characterization of the Equality in some Models of the Lambda Calculus.
J. London Math. Soc. (2), 12:361–370, 1976.

[Kle36] S. C. Kleene.
General Recursive Functions of Natural Numbers.
Mathematical Annals, 112:727–742, 1936.

[Koy82] C. P. J. Koymans.
Models of the Lambda Calculus.
Information and Control, 52:206–332, 1982.

[Koy84] C. P. J. Koymans.
Models of the Lambda Calculus.
PhD thesis, University of Utrecht, 1984.

[KP78] G. Kahn and G. D. Plotkin.
Structures de Données Concrètes.
Technical Report Report 338, INRIA-LABORIA, 1978.

[Lam80] J. Lambek.
From Lambda Calculus to Cartesian Closed Categories.
In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–402, Academic Press, London, 1980.

[Lei86] Daniel Leivant.
Typing and Computational Properties of Lambda Expressions.

Theoretical Computer Science, 44:51–68, 1986.

[Lev75] Jean-Jacqués Levy.
An Algebraic Interpretation of Equality in Some Models of the Lambda Calculus.
In C. Böhm, editor, *Lambda Calculus and Computer Science Theory*, pages 147–165, Springer-Verlag, 1975.
LNCS No. 37.

[LM84] G. Longo and E. Moggi.
Cartesian Closed Categories of Enumerations for Effective Type Structure, Part I and II.
In G. Kahn, D. B. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*, pages 235–255, Springer-Verlag, 1984.
Proceedings of International Symposium at Sophia-Antipolis, France, June 27–29, 1984. Lecture Notes in Computer Science No. 173.

[Lon83] Giuseppe Longo.
Set-Theoretical Models of Lambda Calculus: Theories, Expansions and Isomorphisms.
Annals of Pure and Applied Logic, 24:153–188, 1983.

[Lon84] G. Longo.
Limits, Higher Type Computability and Type-free Languages.
In *Proc. Math. Foundations of Computer Science*, Springer-Verlag, Berlin, 1984.
Lecture Notes in Computer Science.

[LS86] J. Lambek and P. J. Scott.
Introduction to Higher Categorical Logic.
Cambridge Studies in Advanced Mathematics No. 7, Cambridge University Press, 1986.

[Mey82] Albert Meyer.
What is a Model of the Lambda Calculus?
Information and Control, 52:87–122, 1982.

[Mil74] R. E. Milne.
The Formal Semantics of Computer Languages and their Implementations.
PhD thesis, University of Cambridge, 1974.

[Mil77] Robin Milner.
Fully Abstract Models of Typed Lambda-Calculus.
Theoretical Computer Science, 4:1–22, 1977.

[Mog86] Eugenio Moggi.

Categories of Partial Morphisms and the λ_P-Calculus.
In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Category Theory and Computer Programming*, pages 242–251, Springer-Verlag, 1986.
LNCS No. 240.

[Mog88a] Eugenio Moggi.
Partial Cartesian Closed Categories of Effective Objects.
Information and Computation, 1988.
To Appear.

[Mog88b] Eugenio Moggi.
The Partial Lambda Calculus.
PhD thesis, University of Edinburgh, 1988.

[Mor68] J.-H. Morris.
Lambda Calculus Models of Programming Languages.
PhD thesis, M.I.T., 1968.

[MP87] P. D. Moses and G. D. Plotkin.
On Proving Limiting Completeness.
SIAM J. Computing, 16:179–194, 1987.

[MS76] R. E. Milne and C. Strachey.
A Theory of Programming Language Semantics.
Chapman and Hall, London, 1976.

[Mul86] Ketan Mulmuley.
Fully Abstract Submodels of Typed Lambda Calculus.
Journal of Computer and System Sciences, 33:2–46, 1986.

[PJ87] Simon L. Peyton Jones.
The Implementation of Functional Programming Languages.
Prentice-Hall, 1987.
Prentice-Hall International Series in Computer Science.

[Plo72] G. D. Plotkin.
A Set-Theoretical Definition of Application.
Technical Report MIP-R-95, School of A.I., Univ. of Edinburgh, 1972.

[Plo73] G. D. Plotkin.
Lambda-Definability and Logical Relations.
Technical Report SAI-RM-4, School of A.I., Univ. of Edinburgh, 1973.

[Plo74] Gordon D. Plotkin.
The λ-Calculus is ω-Incomplete.
J. Symbolic Logic, 39:313–317, 1974.

[Plo75] Gordon D. Plotkin.
Call-by-Name, Call-by-Value and the Lambda Calculus.
Theoretical Computer Science, 1:125–159, 1975.

[Plo77] Gordon D. Plotkin.
LCF as a Programming Language.
Theoretical Computer Science, 5:223–255, 1977.

[Plo78] G. D. Plotkin.
T^{ω} as a Universal Domain.
J. Computer and System Sciences, 17:209–236, 1978.

[Plo81] Gordon D. Plotkin.
Post-Graduate Lecture Notes in Advanced Domain Theory.
1981.
Dept. of Computer Science, Univ. of Edinburgh.

[Plo85] Gordon D. Plotkin.
Types and Partial Functions.
1985.
Post-Graduate Lecture Notes, Dept. of Computer Science, University of Edinburgh.

[Rey70] J. C. Reynolds.
GEDANKEN – A Simple Typeless Language based on Principle of Completeness and Reference Concept.
J. ACM, 13:136–140, 1970.

[Rog67] Hartley Jr. Rogers.
Theory of Recursive Functions and Effective Computability.
McGraw-Hill Book Company, 1967.

[Ros86] Giuseppe Rosolini.
Continuity and Effectiveness in Topoi.
PhD thesis, Carnegie-Mellon University, 1986.

[RR88] E. Robinson and G. Rosolini.
Categories of Partial Maps.
J. Symbolic Logic, 1988.
To appear.

[Sco72] Dana S. Scott.
Continuous Lattices.
In E. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, pages 97–136, Springer-Verlag, Berlin, 1972.
Lecture Note in Mathematics 274.

[Sco76] Dana S. Scott.

Data Types as Lattices.
SIAM J. Computing, 5:522–587, 1976.

[Sco79] Dana S. Scott.
Identity and Existence in Intuitionistic Logic.
In M. P. Fourman, C. J. Mulvey, and D. S. Scott, editors, *Applications of Sheaves*, pages 660–696, Springer-Verlag, 1979.

[Sco80a] Dana Scott.
Lambda Calculus: Some Models, Some Philosophy.
In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, pages 223–265, North-Holland Publishing Company, 1980.

[Sco80b] Dana S. Scott.
Relating Theories of Lambda Calculus.
In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 403–450, Academic Press, 1980.

[SP82] M. B. Smyth and G. D. Plotkin.
The Category-Theoretic Solution of Recursive Domain Equations.
SIAM J. Computing, 11:761–783, 1982.

[SS63] J. C. Shepherdson and H. E. Sturgis.
Computability of Recursive Functions.
J. ACM, 10:217–255, 1963.

[Sta85] R. Statman.
Logical Relations and the Typed λ-Calculus.
Information and Control, 65:85–97, 1985.

[Sta86] Rick Statman.
Every Countable Poset is Embeddable in the Poset of Unsolvable Terms.
Theoretical Computer Science, 48:95–100, 1986.

[Sto88] Allen Stoughton.
Fully Abstract Models of Programming Languages.
Pitman, 1988.
Research Notes in Theoretical Computer Science.

[Tur36] Alan Turing.
On Computable Numbers with an Application to the Entscheidungsproblem.
Proc. London Math. Soc., 42:230–265, 1936.

[Tur79] D. A. Turner.
A New Implementation Technique for Applicative Languages.

Software Practice and Experience, 9:31–49, 1979.

[Tur81] D. A. Turner.
Aspects of the Implementation of Programming Languages.
PhD thesis, Oxford University, 1981.

[Tur85] D. A. Turner.
Miranda — a non-strict functional language with polymorphic types.
In J. P. Jouannaud, editor, *Functional Programming Languages and Computer Architectures*, Springer-Verlag, Berlin, 1985.
LNCS 201.

[Wad71] C. P. Wadsworth.
Semantics and Pragmatics of the Lambda Calculus.
PhD thesis, Oxford University, 1971.

[Wad78] Christopher P. Wadsworth.
Approximate Reduction and Lambda Calculus Models.
SIAM J. Computing, 7:337–356, 1978.

[Wad85] P. Wadler.
Introduction to ORWELL.
Technical Report, Oxford University Programming Research Group, 1985.

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

[Win87] Glynn Winskel.
Event Structures.
In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Application and Relationships to Other Models of Concurrency*, pages 325 — 392, Springer Verlag, 1987.
LNCS 255.