The Lazy Lambda Calculus

Samson Abramsky

Department of Computing

Imperial College of Science, Technology and Medicine

Published in *Research Topics in Functional Programming*, ed. D. Turner,

pages 65–117, Addison Wesley 1990

December 17, 1987

1 Introduction

The commonly accepted basis for functional programming is the λ-calculus; and it is folklore that the λ-calculus *is* the prototypical functional language in purified form. But what is the λ-calculus? The syntax is simple and classical; variables, abstraction and application in the pure calculus, with applied calculi obtained by adding constants. The further elaboration of the theory, covering conversion, reduction, theories and models, is laid out in Barendregt's already classical treatise [Bar84]. It is instructive to recall the following crux, which occurs rather early in that work (p. 39):

Meaning of λ-terms: first attempt

- The meaning of a λ-term is its normal form (if it exists).
- All terms without normal forms are identified.

This proposal incorporates such a simple and natural interpretation of the λ-calculus as a programming language, that if it worked there would surely be no doubt that it was the right one. However, it gives rise to an inconsistent theory! (see the above reference).

Second attempt

- The meaning of λ-terms is based on head normal forms via the notion of *Bohm tree.*
- All unsolvable terms (no head normal form) are identified.

This second attempt forms the central theme of Barendregt's book, and gives rise to a very beautiful and successful theory (henceforth referred to as the "standard theory"), as that work shows.

This, then, is the commonly accepted foundation for functional programming; more precisely, for the *lazy* functional languages, which represent the mainstream of current functional programming practice. Examples: MIRANDA [Tur85], LML [Aug84], LISPKIT [Hen80], ORWELL [Wad85], PONDER [Fai85], TALE [BvL86]. But do these languages as defined and implemented actually evaluate terms to head normal form? To the best of my knowledge, *not a single one of them does so.* Instead, they evaluate to *weak head normal form*, i.e. they do not evaluate under abstractions.

Example

$\lambda x.(\lambda y.y)M$ is in weak head normal form, but not in head normal form, since it contains the head redex $(\lambda y.y)M$.

So we have a mismatch between theory and practice. Since current practice is well-motivated by efficiency considerations and is unlikely to be abandoned readily, it makes sense to see if a good modified theory can be developed for it. To see that the theory really does need to be modified:

Example

Let $\mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$ be the standard unsolvable term. Then

$$\lambda x.\mathbf{\Omega} = \mathbf{\Omega}$$

in the standard theory, since $\lambda x.\mathbf{\Omega}$ is also unsolvable; but $\lambda x.\mathbf{\Omega}$ is in weak head normal form, hence should be distinguished from $\mathbf{\Omega}$ in our "lazy" theory.

We now turn to a second point in which the standard theory is not completely satisfactory.

Is the λ-calculus a programming language?

In the standard theory, the λ-calculus may be regarded as being characterised by the type equation

$$D = [D \to D]$$

(for justification of this in a general categorical framework, see e.g. [Sco80, Koy82, LS86]).

It is one of the most remarkable features of the various categories of domains used in denotational semantics that they admit non-trivial solutions of this equation. However, there is no *canonical* solution in any of these categories (in particular, the initial solution is trivial – the one-point domain).

I regard this as a symptom of the fact that the pure λ-calculus in the standard theory *is not a programming language*. Of course, this is to some extent a matter of terminology, but I feel that the expression "programming language" should be reserved for a formalism with a definite computational interpretation (an operational semantics). The pure λ-calculus as ordinarily conceived is too schematic to qualify.

A further indication of the same point is that studies such as Plotkin's "LCF Considered as a Programming Language" [Plo77] have not been carried over to the pure λ-calculus, for lack of any convincing way of doing do in the standard theory. This in turn impedes the development of a theory which integrates the λ-calculus with concurrency and other computational notions.

We shall see that by contrast with this situation, the lazy λ-calculus we shall develop does have a canonical model; that Plotkin's ideas can be carried over to it in a very natural way; and that the theory we shall develop will run quite strikingly in parallel with our treatment of concurrency in [Abr87a].

The plan of the remainder of the paper is as follows. In the next section, we introduce the intuitions on which our theory is based, in the concrete setting of λ-terms. We then set up the axiomatic framework for our theory, based on the notion of *applicative transition systems*. This forms a bridge both to the standard theory, and to concurrency and other computational notions. We then introduce a domain equation for applicative transition systems, and use it to derive a *domain logic* in the sense of [Abr87c, Abr87b]. We prove Duality, Characterisation, and Final Algebra theorems; and obtain a strikingly simple proof of a Computational Adequacy Theorem, which asserts that a term converges operationally if and only if it denotes a non-bottom element in our domain.

We then show how the ideas of [Plo77] can be formulated in our setting. Two distinctive features of our approach are:

- the axiomatic treatment of concepts and results usually presented concretely in work on programming language semantics
- the use of our domain logic as a tool in studying the equational theory over our "programs" (λ-terms).

Our results can also be interpreted as settling a number of questions and conjectures concerning the Domain Interpretation of Martin-Lof's Intuitionistic Type Theory raised at the 1983 Chalmers University Workshop on Semantics of Programming Languages [DNPS83].

Finally, we consider some extensions and variations of the theory.

2 The Lazy Lambda-Calculus

We begin with the syntax, which is standard.

Definition 2.1 We assume a set Var of variables, ranged over by x, y, z. The set $\mathbf{\Lambda}$ of λ-terms, ranged over by M, N, P, Q, R is defined by

$$M \;\; ::= \;\; x \mid \lambda x.M \mid MN.$$

For standard notions of free and bound variables etc. we refer to [Bar84]. The reader should also refer to that work for definitions of notation such as: $\mathsf{FV}(M)$, $C[\cdot]$, Λ^0. Our one point of difference concerns substitution; we write $M[N/x]$ rather than $M[x := N]$.

Definition 2.2 The relation $M \Downarrow N$ ("M converges to principal weak head normal form N") is defined inductively over Λ^0 as follows:

- $\lambda x.M \Downarrow \lambda x.M$
- $\dfrac{M \Downarrow \lambda x.P \quad P[N/x] \Downarrow Q}{MN \Downarrow Q}$

Notation

$$\begin{array}{lcll} M\Downarrow & \equiv & \exists N. M \Downarrow N & (\text{"}M \text{ converges"}) \\ M\Uparrow & \equiv & \neg(M\Downarrow) & (\text{"}M \text{ diverges"}) \end{array}$$

It is clear that $\_\Downarrow\_$ is a partial function, i.e. evaluation is deterministic.

We now have an (unlabelled) transition system $(\Lambda^0, \_\Downarrow\_)$. The relation $\Downarrow$ by itself is too "shallow" to yield information about the behaviour of a term under all experiments. However, just as in the study of concurrency, we shall use it as a building block for a deeper relation, which we shall call *applicative bisimulation*. To motivate this relation, let us spell out the observational scenario we have in mind.

Given a closed term M, the only experiment of depth 1 we can do is to evaluate M and see if it converges to some abstraction (weak head normal form) $\lambda x.M_1$. If it does so, we can continue the experiment to depth 2 by supplying a term N_1 as input to M_1, and so on. Note that what the experimenter can observe at each stage is only the *fact* of convergence, not which term lies under the abstraction. We can picture matters thus:

$$\begin{array}{ll} \text{Stage 1 of experiment:} & M \Downarrow \lambda x.M_1; \\ & \text{environment "consumes" } \lambda, \\ & \text{produces } N_1 \text{ as input} \\ \text{Stage 2 of experiment:} & M_1[N_1/x] \Downarrow \ldots \\ \vdots & \end{array}$$

Definition 2.3 (Applicative Bisimulation) We define a sequence of relations $\{\lesssim_k\}_{k\in\omega}$ on Λ^0:

$$\begin{array}{lcl} M\lesssim_0 N & \equiv & \mathsf{true} \\ M\lesssim_{k+1} N & \equiv & M\Downarrow\lambda x.M_1 \;\Rightarrow\; \exists N_1.[N\Downarrow\lambda y.N_1 \;\&\; \forall P\in\Lambda^0.\, M_1[P/x]\lesssim_k N_1[P/y]] \\ M\lesssim^B N & \equiv & \forall k\in\omega.\, M\lesssim_k N \end{array}$$

Clearly each $\lesssim_k$ and $\lesssim^B$ is a preorder. We extend $\lesssim^B$ to Λ by:

$$M\lesssim^B N \;\equiv\; \forall\sigma : \mathsf{Var}\to\Lambda^0.\, M\sigma\lesssim^B N\sigma$$

(where e.g. $M\sigma$ means the result of substituting σx for each $x\in FV(M)$ in M). Finally,

$$M\sim^B N \;\equiv\; M\lesssim^B N \;\&\; N\lesssim^B M.$$

Using standard techniques [Mos74, Mil83], $\sim^B$ can be shown to be the maximal fixpoint of a certain function, and hence to satisfy:

$$M\lesssim^B N \iff M\Downarrow\lambda x.M_1 \;\Rightarrow\; \exists N_1.[N\Downarrow\lambda y.N_1 \;\&\; \forall P\in\Lambda^0.\, M_1[P/x]\lesssim^B N_1[P/y]]$$

Further details are given in the next section.

The applicative bisimulation relation can be described in a more traditional way (from the point of view of λ-calculus) as a "Morris-style contextual congruence" [Mor68, Plo77, Mil77, Bar84].

Definition 2.4 The relation $\lesssim^C$ on Λ^0 is defined by

$$M\lesssim^C N \;\equiv\; \forall C[\cdot]\in\Lambda^0.\, C[M]\Downarrow \;\Rightarrow\; C[N]\Downarrow.$$

This is extended to Λ in the same way as $\lesssim^B$.

Proposition 2.5 $\lesssim^B = \lesssim^C$.

This is a special case of a result we will prove later. Our proof will make essential use of domain logic, despite the fact that the *statement* of the result does not mention domains at all. The reader who may be sceptical of our approach is invited to attempt a direct proof.

We now list some basic properties of the relation $\lesssim^B$ (superscript omitted).

Proposition 2.6 *For all* $M, N, P\in\Lambda$*:*

$$\begin{array}{rl} (i) & M\lesssim M \\ (ii) & M\lesssim N \;\&\; N\lesssim P \;\Rightarrow\; M\lesssim P \\ (iii) & M\lesssim N \;\Rightarrow\; M[P/x]\lesssim N[P/x] \\ (iv) & M\lesssim N \;\Rightarrow\; P[M/x]\lesssim P[N/x] \\ (v) & \lambda x.M \sim \lambda y.M[y/x] \\ (vi) & M\lesssim N \;\Rightarrow\; \lambda x.M\lesssim\lambda x.N \\ (vii) & M_i\lesssim N_i\; (i=1,2) \;\Rightarrow\; M_1M_2\lesssim N_1N_2. \end{array}$$

PROOF. (i)–(iii) and (v)–(vi) are trivial; (vii) follows from (ii) and (iv), since taking $C_1 \equiv [\cdot]M_2$, $M_1M_2 \lesssim N_1M_2$, and taking $C_2 \equiv N_1[\cdot]$, $N_1M_2 \lesssim N_1N_2$, whence $M_1M_2 \lesssim N_1N_2$. It remains to prove (iv), which by 2.5 is equivalent to

$$M \lesssim^C N \;\Rightarrow\; P[M/x] \lesssim^C P[N/x].$$

We rename all bound variables in P to avoid clashes with M and N, and replace x by $[\cdot]$ to obtain a context $P[\cdot]$ such that

$$P[M/x] = P[M], \quad P[N/x] = P[N].$$

Now let $C[\cdot] \in \Lambda^0$ and $\sigma \in \mathsf{Var} \to \Lambda^0$ be given. Let $C_1[\cdot] \equiv C[P[\cdot]\sigma]$. $M \lesssim^C N$ implies

$$C_1[M\sigma]\Downarrow \;\Rightarrow\; C_1[N\sigma]\Downarrow$$

which, since $(P[M/x])\sigma = (P[\cdot]\sigma)[M\sigma]$, yields

$$C[(P[M/x])\sigma]\Downarrow \;\Rightarrow\; C[(P[N/x])\sigma]\Downarrow,$$

as required. ∎

This Proposition can be summarised as saying that $\lesssim^B$ is a *precongruence*. We thus have an (in)equational theory $\lambda\ell = (\Lambda, \sqsubseteq, =)$, where:

$$\lambda\ell \vdash M \sqsubseteq N \;\equiv\; M \lesssim^B N$$
$$\lambda\ell \vdash M = N \;\equiv\; M \sim^B N.$$

What does this theory look like?

Proposition 2.7 *(i) The theory λ [Bar84] is included in $\lambda\ell$; in particular,*

$$\lambda\ell \;\vdash\; (\lambda x.M)N = M[N/x] \quad (\beta).$$

(ii) $\mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$ is a least element for $\sqsubseteq$, i.e.

$$\lambda\ell \;\vdash\; \mathbf{\Omega} \sqsubseteq x.$$

(iii) (η) is not valid in $\lambda\ell$, e.g.

$$\lambda\ell \;\nvdash\; \lambda x.\mathbf{\Omega}x = \mathbf{\Omega},$$

but we do have the following conditional version of η:

$$(\Downarrow\eta) \;\; \lambda\ell \;\vdash\; \lambda x.Mx = M \quad (M\Downarrow,\; x \notin FV(M))$$

$$(M\Downarrow \;\equiv\; \forall\sigma \in \mathsf{Var} \to \Lambda^0.\, (M\sigma)\Downarrow).$$

(iv) $\mathbf{YK}$ is a greatest element for $\sqsubseteq$, i.e.

$$\lambda\ell \;\vdash\; x \sqsubseteq \mathbf{YK}.$$

Proof. *(i)* is an easy consequence of 2.6.
(ii). $\mathbf{\Omega}\Uparrow$, hence $\mathbf{\Omega}\lesssim^{B} M$ for all $M \in \Lambda^0$.
(iii). $\lambda x.\mathbf{\Omega}x \not\lesssim_1 \mathbf{\Omega}$, since $(\lambda x.\mathbf{\Omega}x)\Downarrow$. Now suppose $M\Downarrow$, and let $\sigma : \mathsf{Var} \to \Lambda^0$ be given. Then $(M\sigma)\Downarrow\lambda y.N$. For any $P \in \Lambda^0$,

$$\begin{aligned} (M\sigma)P\Downarrow Q &\Leftrightarrow ((M\sigma)x)[P/x]\Downarrow Q \quad \text{since } x \notin FV(M), \\ &\Leftrightarrow ((\lambda x.Mx)\sigma)P\Downarrow Q, \end{aligned}$$

and so $M \sim^B \lambda x.Mx$, as required.
(iv). Note that $\mathbf{YK}\Downarrow\lambda y.N$, where $N \equiv (\lambda x.\mathbf{K}(xx))(\lambda x.\mathbf{K}(xx))$, and that for all P,

$$N[P/y]\Downarrow\lambda y.N.$$

Hence for all $P_1, \ldots, P_n$ $(n \geq 0)$,

$$\mathbf{YK}P_1 \ldots P_n\Downarrow,$$

and so $M\lesssim^{B}\mathbf{YK}$ for all $M \in \Lambda^0$. ∎

To understand *(iv)*, we can think of $\mathbf{YK}$ as the infinite process

$$\overset{\lambda}{\circlearrowleft}$$

solving the equation

$$\xi = \lambda x.\xi.$$

This is a top element in our applicative bisimulation ordering because it converges under all finite stages of evaluation for all arguments—the experimenter can always observe convergence (or "consume an infinite λ-stream").

We can make some connections between the theory $\lambda\ell$ and [Lon83], as pointed out to me by Chih-Hao Ong. Firstly, 2.7(ii) can be generalised to:

- The set of terms in Λ^0 which are least in $\lambda\ell$ are exactly the PO_0 terms in the terminology of [Lon83].

Moreover, $\mathbf{YK}$ is an O_∞ term in the terminology of [Lon83], although it is *not* a greatest element in the ordering proposed there.

3 Applicative Transition Systems

The theory $\lambda\ell$ defined in the previous section was derived from a particular operational model, the transition system $(\Lambda^0, \Downarrow)$. What is the general concept of which this is an example?

Definition 3.1 A *quasi-applicative transition system* is a structure (A, ev) where

$$ev : A \rightharpoonup (A \to A).$$

Notations:

$$\begin{array}{rrcl} (i) & a \Downarrow f & \equiv & a \in \mathsf{dom}\, ev \;\&\; ev(a) = f \\ (ii) & a \Downarrow & \equiv & a \in \mathsf{dom}\, ev \\ (iii) & a \Uparrow & \equiv & a \notin \mathsf{dom}\, ev \end{array}$$

Definition 3.2 (Applicative Bisimulation) Let (A, ev) be a quasi-ats. We define

$$F : Rel(A) \to Rel(A)$$

by

$$F(R) = \{(a, b) : a \Downarrow f \implies b \Downarrow g \;\&\; \forall c \in A.\, f(c) R g(c)\}.$$

Then $R \in Rel(A)$ is an *applicative bisimulation* iff $R \subseteq F(R)$; and $\lesssim^B \in Rel(A)$ is defined by

$$a \lesssim^B b \;\equiv\; aRb \text{ for some applicative bisimulation } R.$$

Thus $\lesssim^B = \bigcup\{R \in Rel(A) : R \subseteq F(R)\}$, and hence is the maximal fixpoint of the monotone function F. Since the relation $\Downarrow$ is a partial function, it is easily shown that the closure ordinal of F is $\leq \omega$, and we can thus describe $\lesssim^B$ more explicitly as follows:

- $a \lesssim^B b \quad \equiv \quad \forall k \in \omega.\, a \lesssim_k b$
- $a \lesssim_0 b \quad \equiv \quad \mathsf{true}$
- $a \lesssim_{k+1} b \quad \equiv \quad a \Downarrow f \implies b \Downarrow g \;\&\; \forall c \in A.\, f(c) \lesssim_k g(c)$
- $a \sim^B b \quad \equiv \quad a \lesssim^B b \;\&\; b \lesssim^B a.$

It is easily seen that $\lesssim^B$, and also each $\lesssim_k$, is a preorder; $\sim^B$ is therefore an equivalence.

We now come to our main definition.

Definition 3.3 An *applicative transition system* (ats) is a quasi-ats (A, ev) satisfying:

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

An ats has a well-defined quotient $(A/{\sim^B}, ev/{\sim^B})$, where

$$ev/{\sim^B}([a]) = \begin{cases} [b] \mapsto [f(b)], & a{\Downarrow}f \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The reader should now refresh her memory of such notions as *applicative structure, combinatory algebra* and *lambda model* from [Bar84, Chapter 5].

Definition 3.4 A *quasi-applicative structure with divergence* is a structure $(A, \cdot\ , \Uparrow)$ such that $(A, \cdot\ )$ is an applicative structure, and $\Uparrow \subseteq A$ is a divergence predicate satisfying

$$x{\Uparrow} \implies (x \cdot y){\Uparrow}.$$

Given $(A, \cdot\ , \Uparrow)$, we can define

$$a \lesssim^A b \;\equiv\; a{\Downarrow} \implies b{\Downarrow} \;\&\; \forall c \in A.\, a \cdot c \lesssim^A b \cdot c$$

as the maximal fixpoint of a monotone function along identical lines to 3.2.

Applicative transition systems and applicative structures with divergence are not quite equivalent, but are sufficiently so for our purposes:

Proposition 3.5 *Given an ats $\mathcal{B} = (A, ev)$, we define $\mathcal{A} = (A, \cdot\ , \Uparrow)$ by*

$$a \cdot b \;\equiv\; \begin{cases} a, & a{\Uparrow} \\ f(b) & a{\Downarrow}f. \end{cases}$$

Then

$$a \lesssim^A b \iff a \lesssim^B b,$$

and moreover we can recover $\mathcal{B}$ from $\mathcal{A}$ by

$$ev(a) = \begin{cases} b \mapsto a \cdot b, & a{\Downarrow} \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

Furthermore, $\cdot$ is compatible with $\lesssim^B$, i.e.

$$a_i \lesssim^B b_i \ (i = 1, 2) \;\Rightarrow\; a_1 \cdot a_2 \lesssim^B b_1 \cdot b_2. \quad \blacksquare$$

We now turn to a language for talking about these structures.

Definition 3.6 We assume a fixed set of variables Var. Given an applicative structure $\mathcal{A} = (A, \cdot\ )$, we define $CL(\mathcal{A})$, the *combinatory terms over $\mathcal{A}$*, by

- $\mathsf{Var} \subseteq CL(\mathcal{A})$
- $\{c_a : a \in A\} \subseteq CL(\mathcal{A})$
- $M, N \in CL(\mathcal{A}) \;\Rightarrow\; MN \in CL(\mathcal{A})$.

Let $Env(\mathcal{A}) \equiv \mathsf{Var} \to A$. Then the *interpretation function*

$$[\![\cdot]\!]^{\mathcal{A}} : CL(\mathcal{A}) \to Env(\mathcal{A}) \to A$$

is defined by:

$$\begin{array}{rcl} [\![x]\!]^{\mathcal{A}}_{\rho} & = & \rho x \\ [\![c_a]\!]^{\mathcal{A}}_{\rho} & = & a \\ [\![MN]\!]^{\mathcal{A}}_{\rho} & = & ([\![M]\!]^{\mathcal{A}}_{\rho}) \cdot ([\![N]\!]^{\mathcal{A}}_{\rho}). \end{array}$$

Given an ats $\mathcal{A} = (A, ev)$, with derived applicative structure $(A, \cdot\ )$, the satisfaction relation between $\mathcal{A}$ and atomic formulae over $CL(\mathcal{A})$, of the forms

$$M \sqsubseteq N, \quad M = N, \quad M\Downarrow \quad M\Uparrow$$

is defined by:

$$\begin{array}{lcl} \mathcal{A}, \rho \models M \sqsubseteq N & \equiv & [\![M]\!]^{\mathcal{A}}_{\rho} \lesssim^{B} [\![N]\!]^{\mathcal{A}}_{\rho} \\ \mathcal{A}, \rho \models M = N & \equiv & [\![M]\!]^{\mathcal{A}}_{\rho} \sim^{B} [\![N]\!]^{\mathcal{A}}_{\rho} \\ \mathcal{A}, \rho \models M\Downarrow & \equiv & [\![M]\!]^{\mathcal{A}}_{\rho}\Downarrow \\ \mathcal{A}, \rho \models M\Uparrow & \equiv & [\![M]\!]^{\mathcal{A}}_{\rho}\Uparrow \end{array}$$

while

$$\mathcal{A} \models \phi \equiv \forall \rho \in Env(\mathcal{A}). \mathcal{A}, \rho \models \phi.$$

This is extended to first-order formulae in the usual way.

Note that equality in $CL(\mathcal{A})$ is being interpreted by bisimulation in $\mathcal{A}$. We could have retained the standard notion of interpretation as in [Bar84] by working in the quotient structure $(A/{\sim^B}, \cdot / {\sim^B})$. This is equivalent, in the sense that the same sentences are satisfied.

Definition 3.7 A *lambda transition system* (lts) is a structure (A, ev, k, s), where:

- (A, ev) is an ats
- $k, s \in A$, and A satisfies the following axioms (writing **K, S** for c_k, c_s):

  - $\mathbf{K}\Downarrow, \quad \mathbf{K}x\Downarrow$
  - $\mathbf{K}xy = x$
  - $\mathbf{S}\Downarrow, \quad \mathbf{S}x\Downarrow, \quad \mathbf{S}xy\Downarrow$
  - $\mathbf{S}xyz = (xz)(yz)$

We now check that these definitions do indeed capture our original example.

Example

We define $\ell = (\Lambda^0, ev)$, where

$$ev(M) = \begin{cases} P \mapsto N[P/x], & M{\Downarrow}\lambda x.N \\ \text{undefined} & \text{otherwise.} \end{cases}$$

ℓ is indeed an ats by 2.6(iv). Moreover, it is an lts via the definitions

$$\begin{array}{rcl} k & \equiv & \lambda x.\lambda y.x \\ s & \equiv & \lambda x.\lambda y.\lambda z.(xz)(yz). \end{array}$$

We now see how to interpret λ-terms in any lts.

Definition 3.8 Given an lts $\mathcal{A}$, we define $\Lambda(\mathcal{A})$, the λ-terms over $\mathcal{A}$, by the same clauses as for $CL(\mathcal{A})$, plus the additional one:

- $x \in \mathsf{Var}, M \in \Lambda(\mathcal{A}) \;\Rightarrow\; \lambda x.M \in \Lambda(\mathcal{A})$.

We define a translation

$$(\cdot)_{CL} : \Lambda(\mathcal{A}) \to CL(\mathcal{A})$$

by

$$\begin{array}{rcl} (x)_{CL} & \equiv & x \\ (c_a)_{CL} & \equiv & c_a \\ (MN)_{CL} & \equiv & (M)_{CL}(N)_{CL} \\ (\lambda x.M)_{CL} & \equiv & \lambda^* x.(M)_{CL} \end{array}$$

where

$$\begin{array}{rcl} \lambda^* x.x & \equiv & \mathbf{I} \ (\equiv \mathbf{SKK}) \\ \lambda^* x.M & \equiv & \mathbf{K}M \quad (x \notin FV(M)) \\ \lambda^* x.MN & \equiv & \mathbf{S}(\lambda^* x.M)(\lambda^* x.N). \end{array}$$

We now extend $[\![\cdot]\!]$ to $\Lambda(\mathcal{A})$ by:

$$[\![M]\!]^{\mathcal{A}}_{\rho} \;\equiv\; [\![(M)_{CL}]\!]^{\mathcal{A}}_{\rho}.$$

Definition 3.9 We define two sets of formulae over Λ:

- *Atomic formulae:*

$$\mathsf{AF} \equiv \{M \sqsubseteq N,\ M = N,\ M{\Downarrow},\ N{\Uparrow} : M, N \in \Lambda\}$$

- *Conditional formulae:*

$$\mathsf{CF} \equiv \{\bigwedge_{i\in I} M_i\Downarrow \wedge \bigwedge_{j\in J} N_j\Uparrow \Rightarrow F : F \in \mathsf{AF}, M_i, N_i \in \Lambda, I, J \text{ finite}\}$$

Note that, taking $I = J = \varnothing$, $\mathsf{AF} \subseteq \mathsf{CF}$. Now given an lts $\mathcal{A}$, $\Im(\mathcal{A})$, the *theory* of $\mathcal{A}$, is defined by

$$\Im(\mathcal{A}) \;\equiv\; \{C \in \mathsf{CF} : \mathcal{A} \models C\}.$$

We also write $\Im^0(\mathcal{A})$ for the restriction of $\Im(\mathcal{A})$ to closed formulae; and given a set Con of constants and an interpretation $\mathsf{Con} \to A$, we write $\Im(\mathcal{A}, \mathsf{Con})$ for the theory of conditional formulae built from terms in $\Lambda(\mathsf{Con})$.

Example (continued). We set $\lambda\ell = \Im(\ell)$. This is consistent with our usage in the previous section. We saw there that $\lambda\ell$ satisfied much stronger properties than the simple combinatory algebra axioms in our definition of lts. It might be expected that these would fail for general lts; but this is to overlook the powerful extensionality principle built into our definition of the theory of an ats through the applicative bisimulation relation.

Proposition 3.10 *Let $\mathcal{A}$ be an ats. The axiom scheme of* conditional extensionality *over $CL(\mathcal{A})$:*

$$(\Downarrow\text{ext}) \quad M\Downarrow \,\&\, N\Downarrow \;\Rightarrow\; ([\forall x.Mx = Nx] \;\Rightarrow\; M = N) \qquad (x \notin FV(M) \cup FV(N))$$

is valid in $\mathcal{A}$.

Proof. Let $\rho \in Env(\mathcal{A})$.

$$\begin{array}{ll}
\bullet & \mathcal{A}, \rho \models M\Downarrow \,\&\, N\Downarrow \,\&\, \forall x.\, Mx = Nx \\
\Rightarrow & [\![M]\!]^{\mathcal{A}}_{\rho}\Downarrow \,\&\, [\![N]\!]^{\mathcal{A}}_{\rho}\Downarrow \,\&\, \forall a \in A.\, [\![M]\!]^{\mathcal{A}}_{\rho} \cdot a = [\![N]\!]^{\mathcal{A}}_{\rho} \cdot a \\
 & \text{since } x \notin FV(M) \cup FV(N) \\
\Rightarrow & [\![M]\!]^{\mathcal{A}}_{\rho} \sim^{A} [\![N]\!]^{\mathcal{A}}_{\rho} \\
\Rightarrow & [\![M]\!]^{\mathcal{A}}_{\rho} \sim^{B} [\![N]\!]^{\mathcal{A}}_{\rho} \\
\Rightarrow & \mathcal{A}, \rho \models M = N. \;\blacksquare
\end{array}$$

Using this Proposition, we can now generalise most of 2.7 to an arbitrary lts.

Theorem 3.11 *Let $\mathcal{A} = (A, ev, k, s)$ be an lts. Then*
(i) $(A, ., k, s)$ is a lambda model, and hence $\lambda \subseteq \Im(\mathcal{A})$.
(ii) $\mathcal{A}$ satisfies the conditional η axiom scheme:

$$(\Downarrow\eta) \;\; M\Downarrow \;\Rightarrow\; \lambda x.Mx = M \quad (x \notin FV(M))$$

(iii) For all $M \in \Lambda^0$:

$$\lambda\ell \vdash M\Downarrow \;\Rightarrow\; \mathcal{A} \models M\Downarrow$$

*(iv) $\mathcal{A} \models x \sqsubseteq$ **YK**.*
(v) $\sqsubseteq$ is a precongruence in $\Im(\mathcal{A})$.

PROOF. (i). Firstly, by the very definition of lts, $\mathcal{A}$ is a combinatory algebra. We now use the following result due to Meyer and Scott, cited from [Bar84, Theorem 5.6.3, p. 117]:

- Let $\mathcal{M}$ be a combinatory algebra. Define

$$\mathbf{1} \equiv \mathbf{1}_1 \equiv \mathbf{S}(\mathbf{KI}), \quad \mathbf{1}_{k+1} \equiv \mathbf{S}(\mathbf{K1}_{\mathrm{k}}).$$

  Then $\mathcal{M}$ is a lambda model iff it satisfies

$$\begin{array}{rl} \text{(I)} & \forall x.\, ax = bx \Rightarrow \mathbf{1}a = \mathbf{1}b \\ \text{(II)} & \mathbf{1}_2\mathbf{K} = \mathbf{K} \\ \text{(III)} & \mathbf{1}_3\mathbf{S} = \mathbf{S}. \end{array}$$

Thus it is sufficient to check that $\mathcal{A}$ satisfies (I)–(III). For (I), note firstly that $\mathcal{A} \models \mathbf{1}a{\Downarrow} \,\&\, \mathbf{1}b{\Downarrow}$ by the convergence axioms for an lts. Hence we can apply 3.10 to obtain

$$\mathcal{A} \models [\forall x.\, \mathbf{1}ax = \mathbf{1}bx] \Rightarrow \mathbf{1}a = \mathbf{1}b.$$

We now assume $\forall x.\, ax = bx$ and prove $\forall x.\, \mathbf{1}ax = \mathbf{1}bx$:

$$\begin{array}{rcl} \mathbf{1}ax & = & \mathbf{S}(\mathbf{KI})ax \\ & = & (\mathbf{KI})x(ax) \\ & = & (\mathbf{KI})x(bx) \\ & = & \mathbf{S}(\mathbf{KI})bx \\ & = & \mathbf{1}bx. \end{array}$$

(II) and (III) are proved similarly.
(ii). Let $\rho \in Env(\mathcal{A})$, and assume $\mathcal{A}, \rho \models M{\Downarrow}$. We must prove that

$$\mathcal{A}, \rho \models \lambda x.Mx = M.$$

Firstly, note that for any abstraction $\lambda z.P$,

$$\mathcal{A} \models \lambda z.P{\Downarrow}$$

by the definition of $\lambda^* z.P$ and the convergence axioms for an lts. Thus since $x \notin FV(M)$, we can apply (${\Downarrow}$ext) to obtain

$$\mathcal{A}, \rho \models [\forall x.\, (\lambda x.Mx)x = Mx] \rightarrow \lambda x.Mx = M.$$

It is thus sufficient to show

$$\mathcal{A} \models (\lambda x.Mx)x = Mx.$$

But this is just an instance of (β), which $\mathcal{A}$ satisfies by (i).

(iii). We calculate:

$$\begin{aligned} \lambda\ell \vdash M\Downarrow &\Rightarrow M\Downarrow\lambda x.N \\ &\Rightarrow \lambda \vdash M = \lambda x.N \\ &\Rightarrow \mathcal{A} \models M = \lambda x.N \\ &\Rightarrow \mathcal{A} \models M\Downarrow, \end{aligned}$$

since $\mathcal{A} \models \lambda x.N\Downarrow$, as noted in (ii).
(iv). By (i) and (iii),

$$\mathcal{A} \models \mathbf{YK}\Downarrow \,\&\, \forall x.\, (\mathbf{YK})x = \mathbf{YK}.$$

Hence we can use the same argument as in 2.7(iv) to prove that

$$\mathcal{A} \models x \sqsubseteq \mathbf{YK}.$$

(v). This assertion amounts to the same list of properties as Proposition 2.6, but with respect to $\Im(\mathcal{A})$. The only difference in the proof is that 2.6(vii) follows immediately from 3.5 and the fact that $\mathcal{A}$ is an ats, and can then be used to prove 2.6(iv) by induction on P. ∎

Part (iii) of the Theorem tells us that all the closed terms which we expect to converge must do so in any lts. What of the converse? For example, do we have

$$\mathcal{A} \models \mathbf{\Omega}\Uparrow$$

in every lts? This is evidently not the case, since we have not imposed any axioms which require *anything* to be divergent.

Observation 3.12 *Let $\mathcal{A} = (A, ev)$ be an ats in which ev is* total, *i.e.* $\mathsf{dom}\ ev = A$. *Then $\Im(\mathcal{A})$ is* inconsistent, *in the sense that*

$$\mathcal{A} \models x = y.$$

This is of course because the distinctions made by applicative bisimulation are based on divergence.

In the light of this observation and 3.11, it is natural to make the following definition in analogy with that in [Bar84]:

Definition 3.13 An lts $\mathcal{A}$ is *sensible* if the converse to 3.11(iii) holds, i.e. for all $M \in \Lambda^0$:

$$\mathcal{A} \models M\Downarrow \iff \lambda\ell \vdash M\Downarrow \iff \exists x, N.\ \lambda \vdash M = \lambda x.N.$$

(The second equivalence is justified by an appeal to the Standardisation Theorem [Bar84].)

4 A Domain Equation for Applicative Bisimulation

We now embark on the same programme as in [Abr87a]; to obtain a domain-theoretic analysis of our computational notions, based on a suitable domain equation. What this should be is readily elicited from the definition of ats. The structure map

$$ev : A \rightharpoonup (A \to A)$$

is *partial*; the standard approach to partial maps in domain theory (*pace* Plotkin's recent work on predomains [Plo85]) is to make them into total ones by sending undefined arguments to a "bottom" element, i.e. changing the type of ev to

$$A \to (A \to A)_\perp.$$

This suggests the domain equation

$$D = (D \to D)_\perp$$

i.e. the denotation of the type expression $\mathsf{rec}\, t.(t \to t)_\perp$. This equation is composed from the function space and lifting constructions. Since **SDom** is closed under these constructions, D is a Scott domain. Indeed, by the same reasoning it is an algebraic lattice. The crucial point is that this equation has a *non-trivial initial solution*, and thus there is a good candidate for a canonical model. To see this, consider the "approximants" D_k, with $D_0 \equiv \mathbf{1}$, $D_{k+1} \equiv (D_k \to D_k)_\perp$. Then

$$\begin{array}{rcl} D_1 & = & (\mathbf{1} \to \mathbf{1})_\perp \cong (\mathbf{1})_\perp \cong \mathbb{O} \\ D_2 & \cong & (\mathbb{O} \to \mathbb{O})_\perp, \quad \text{with four elements} \\ & \vdots & \end{array}$$

etc. We now unpack the structure of D. Our treatment will be rather cursory, as it proceeds along similar lines to our work in [Abr87a]. Firstly, there is an isomorphism pair

$$\mathsf{unfold} : D \to (D \to D)_\perp, \quad \mathsf{fold} : (D \to D)_\perp \to D.$$

Next, we recall the categorical description of lifting, as the left adjoint to the forgetful functor

$$U : \mathbf{Dom}_\perp \to \mathbf{Dom}$$

where $\mathbf{Dom}_\perp$ is the sub-category of strict functions. Thus we have:

- A natural transformation $\mathsf{up} : I_{\mathbf{Dom}} \to U \circ (\cdot)_\perp$.
- For each continuous map $f : D \to UE$ its unique strict extension

$$\mathsf{lift}(f) : (D)_\perp \to_\perp E.$$

Concretely, we can take

$$
\begin{array}{rcl}
(D)_\perp & \equiv & \{\perp\} \cup \{<0, d> \mid d \in D\} \\
x \sqsubseteq y & \equiv & x = \perp \\
& & \text{or } x = <0, d> \;\&\; y = <0, d'> \;\&\; d \sqsubseteq_D d' \\
\mathsf{up}_D(d) & \equiv & <0, d> \\
\mathsf{lift}(f)(\perp) & \equiv & \perp_E \\
\mathsf{lift}(f)<0, d> & \equiv & f(d).
\end{array}
$$

We can now define

$$ev : D \rightharpoonup (D \to D)$$

by

$$ev(d) = \begin{cases} f, & \mathsf{unfold}(d) = <0, f> \\ \text{undefined} & \mathsf{unfold}(d) = \perp. \end{cases}$$

Thus (D, ev) is a quasi-ats, and we write $d \Downarrow f$, $d \Uparrow$ etc. Note that we can recover d from $ev(d)$ by

$$d = \begin{cases} \mathsf{fold}(<0, f>), & d \Downarrow f \\ \perp_D & d \Uparrow. \end{cases}$$

The final ingredient in the definition of D is initiality. The only direct consequence of this which we will use is contained in

Theorem 4.1 *D is internally fully abstract, i.e.*

$$\forall d, d' \in D.\, d \sqsubseteq d' \iff d \lesssim^B d'.$$

Proof. Unpacking the definitions, we see that for all $d, d' \in D$:

$$d \sqsubseteq d' \iff d \Downarrow f \Rightarrow d' \Downarrow g \;\&\; \forall d'' \in D.\, f(d'') \sqsubseteq g(d'').$$

Thus the domain ordering is an applicative bisimulation, and so is included in $\lesssim^B$. For the converse, we need some additional notions. We define d_k, f_k for $d \in D$, $f \in [D \to D]$, $k \in \omega$ by:

- $d_0 \Uparrow$
- $d \Uparrow \Rightarrow d_k \Uparrow$
- $d \Downarrow f \Rightarrow d_{k+1} \Downarrow f_k$
- $f_k : d \mapsto (fd)_k$.

We can use standard techniques to prove, from the initiality of D:

- $\forall d \in D.\, d = \bigsqcup_{k \in \omega} d_k$.

The proof is completed with a routine induction to show that:

$$\forall k \in \omega.\, d \lesssim_k d' \;\Rightarrow\; d_k \sqsubseteq d'_k. \;\blacksquare$$

As an immediate corollary of this result, we see that D is an ats. We thus have an interpretation function

$$[\![\cdot]\!]^D : CL(D) \to Env(D) \to\!\!\to D.$$

We extend this to $\Lambda(D)$ by:

$$[\![\lambda x.M]\!]^D_\rho = \mathsf{fold}(\mathsf{up}(\lambda d \in D.[\![M]\!]^D_{\rho[x \mapsto d]})).$$

Note that the application induced from (D, ev) can be described by

$$d \cdot d' = \mathsf{lift}(Ap)\ \mathsf{unfold}(d)\ d'$$

where

$$Ap : [D \to D] \to D \to D$$

is the standard application function; and is therefore continuous. This together with standard arguments about environment semantics guarantees that our extension of $[\![]\!]^D$ is well-defined. Note also that $[\![\lambda x.M]\!]^D_\rho \neq \bot_D$, as expected.

We can now define

$$\begin{array}{rcl} k & \equiv & [\![\lambda x.\lambda y.x]\!]^D_\rho, \\ s & \equiv & [\![\lambda x.\lambda y.\lambda z.(xz)(yz)]\!]^D_\rho \end{array}$$

for D. It is straightforward to verify

Proposition 4.2 *D is an lts.* $\blacksquare$

Thus far, we have merely used our domain equation to construct a particular lts D. However, its "categorical" or "absolute" nature should lead us to suspect that we can use D to study the whole class of lts. The medium we will use for this purpose is a suitable *domain logic* in the sense of [Abr87b].

5 A Domain Logic for Applicative Transition Systems

Definition 5.1 The syntax of our domain logic $\mathcal{L}$ is defined by

$$\phi \;::=\; t \mid \phi \wedge \psi \mid (\phi \to \psi)_{\perp}$$

Definition 5.2 (Semantics of $\mathcal{L}$) Given a quasi ats $\mathcal{A}$, we define the satisfaction relation $\models_{\mathcal{A}} \subseteq \mathcal{A} \times \mathcal{L}$:

$$\begin{array}{lcl}
a \models_{\mathcal{A}} t & \equiv & \mathsf{true} \\
a \models_{\mathcal{A}} \phi \wedge \psi & \equiv & a \models_{\mathcal{A}} \phi \;\&\; a \models_{\mathcal{A}} \psi \\
a \models_{\mathcal{A}} (\phi \to \psi)_{\perp} & \equiv & a \Downarrow f \;\&\; \forall b \in A.\, b \models_{\mathcal{A}} \phi \Rightarrow f(b) \models_{\mathcal{A}} \psi.
\end{array}$$

Notation:

$$\begin{array}{lcl}
\mathcal{L}(a) & \equiv & \{\phi \in \mathcal{L} \,:\, a \models_{\mathcal{A}} \phi\} \\
\mathcal{A} \models \phi \leq \psi & \equiv & \forall a \in A.\, a \models_{\mathcal{A}} \phi \implies a \models_{\mathcal{A}} \psi \\
\mathcal{A} \models \phi = \psi & \equiv & \forall a \in A.\, a \models_{\mathcal{A}} \phi \iff a \models_{\mathcal{A}} \psi \\
\models \phi \leq \psi & \equiv & \forall \mathcal{A}.\, \mathcal{A} \models \phi \leq \psi \\
\lambda & \equiv & (t \to t)_{\perp} \\
a \sqsubseteq^{\mathcal{L}} b & \equiv & \mathcal{L}(a) \subseteq \mathcal{L}(b).
\end{array}$$

Note that: $\forall a \in A.\, a\Downarrow \iff a \models_{\mathcal{A}} \lambda$.

Lemma 5.3 *Let $\mathcal{A}$ be a quasi ats. Then*

$$\forall a, b \in A.\, a \lesssim^{B} b \implies a \sqsubseteq^{\mathcal{L}} b.$$

Proof. We assume $a \lesssim^{B} b$ and prove $\forall \phi \in \mathcal{L}.\, a \models_{\mathcal{A}} \phi \Rightarrow b \models_{\mathcal{A}} \phi$ by induction on ϕ. The non-trivial case is $(\phi \to \psi)_{\perp}$.

$$\begin{array}{cll}
\bullet & a \models_{\mathcal{A}} (\phi \to \psi)_{\perp} & \\
\implies & a \Downarrow f & \\
\implies & b \Downarrow g \;\&\; \forall c.\, f(c) \lesssim^{B} g(c) & \\
\implies & \forall c.\, c \models_{\mathcal{A}} \phi \implies f(c) \lesssim^{B} g(c) \;\&\; f(c) \models_{\mathcal{A}} \psi & \\
\implies & \forall c.\, c \models_{\mathcal{A}} \phi \Rightarrow g(c) \models_{\mathcal{A}} \psi & \text{ind. hyp.} \\
\implies & b \models_{\mathcal{A}} (\phi \to \psi)_{\perp}. \;\blacksquare &
\end{array}$$

To get a converse to this result, we need a condition on $\mathcal{A}$.

Definition 5.4 A quasi ats A is *approximable* iff

$$\forall a, b_1, \ldots, b_n \in A.\, ab_1 \ldots b_n \Downarrow \;\Rightarrow\; \exists \phi_1, \cdots, \phi_n.$$

$$a \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_n \to \lambda)_\perp \cdots)_\perp \;\;\&\;\; b_i \models_{\mathcal{A}} \phi_i, \;\; 1 \le i \le n.$$

This is a natural condition, which says that convergence of a function application is caused by some finite amount of information (observable properties) of its arguments.

As expected, we have

Theorem 5.5 (Characterisation Theorem) *Let $\mathcal{A}$ be an approximable quasi ats. Then*

$$\lesssim^B = \lesssim^{\mathcal{L}}.$$

PROOF. By 5.3, $\lesssim^B \subseteq \lesssim^{\mathcal{L}}$. For the converse, suppose $a \not\lesssim^B b$. Then for some k, $a \not\lesssim^B_k b$, and so for some $c_1, \cdots, c_k \in A$:

$$ac_1 \cdots c_k \Downarrow \;\&\; bc_1 \cdots c_k \Uparrow.$$

By approximability, for some $\phi_1, \cdots, \phi_k \in \mathcal{L}$,

$$a \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \;\&\; c_i \models_{\mathcal{A}} \phi_i, \;\; 1 \le i \le k.$$

Clearly $b \not\models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$, and so $a \not\lesssim^{\mathcal{L}} b$. ∎

As a further consequence of approximability, we have:

Proposition 5.6 *An approximable quasi ats is an ats.*

PROOF. Suppose $a \Downarrow f$ and $b \lesssim^B c$. We must show $f(b) \lesssim^B f(c)$. It is sufficient to show that for all $k \in \omega$, $d_1, \ldots, d_k \in A$:

$$f(b) d_1 \ldots d_k \Downarrow \;\;\Rightarrow\;\; f(c) d_1 \ldots d_k \Downarrow.$$

Now $f(b) d_1 \ldots d_k \Downarrow$ implies $abd_1 \ldots d_k \Downarrow$; hence by approximability, for some $\phi, \phi_1, \ldots \phi_k \in \mathcal{L}$:

$$a \models_{\mathcal{A}} (\phi \to (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp$$

and

$$b \models_{\mathcal{A}} \phi, \;\; b_i \models_{\mathcal{A}} \phi_i, \;\; 1 \le i \le k.$$

By 5.5, $c \models_{\mathcal{A}} \phi$, and so $abd_1 \ldots d_k \models_{\mathcal{A}} \lambda$, and $f(c) d_1 \ldots d_k \Downarrow$ as required. ∎

We now introduce a proof system for assertions of the form $\phi \le \psi$, $\phi = \psi$ $(\phi, \psi \in \mathcal{L})$.

Proof System For $\mathcal{L}$

$$\text{(REF)} \quad \phi \leq \phi \qquad \text{(TRANS)} \quad \frac{\phi \leq \psi \quad \psi \leq \xi}{\phi \leq \xi}$$

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

$$(t \ - I) \quad \phi \leq t$$

$$(\wedge - I) \quad \frac{\phi \leq \phi_1 \quad \phi \leq \psi_2}{\phi \leq \phi_1 \wedge \phi_2} \qquad (\wedge - E) \quad \phi \wedge \psi \leq \phi \quad \phi \wedge \psi \leq \psi$$

$$((\rightarrow)_\perp - \leq) \quad \frac{\phi_2 \leq \phi_1 \quad \psi_1 \leq \psi_2}{(\phi_1 \rightarrow \psi_1)_\perp \leq (\phi_2 \rightarrow \psi_2)_\perp}$$

$$((\rightarrow)_\perp - \wedge) \quad (\phi \rightarrow \psi_1 \wedge \psi_2)_\perp = (\phi \rightarrow \psi_1)_\perp \wedge (\phi \rightarrow \psi_2)_\perp$$

$$((\rightarrow)_\perp - t) \quad (\phi \rightarrow t)_\perp \leq (t \rightarrow t)_\perp.$$

We write $\mathcal{L} \vdash A$ or just $\vdash A$ to indicate that an assertion A is derivable from these axioms and rules. Note that the converse of $((\rightarrow)_\perp - t)$ is derivable from $(t - I)$ and $((\rightarrow)_\perp - \leq)$; by abuse of notation we refer to the corresponding equation by the same name.

Theorem 5.7 (Soundness Theorem) $\vdash \phi \leq \psi \implies \models \phi \leq \psi$.

PROOF. By a routine induction on the length of proofs. ∎

So far, our logic has been presented in a syntax-free fashion so far as the elements of the ats are concerned. Now suppose we have an lts $\mathcal{A}$. λ-terms can be interpreted in $\mathcal{A}$, and for $M \in \mathbf{\Lambda}^0$, $\rho \in Env(\mathcal{A})$, we can define:

$$M, \rho \models_{\mathcal{A}} \phi \quad \equiv \quad [\![M]\!]^{\mathcal{A}}_{\rho} \models_{\mathcal{A}} \phi.$$

We can extend this to arbitrary terms $M \in \mathbf{\Lambda}$ in the presence of *assumptions* $\Gamma : \mathsf{Var} \rightarrow \mathcal{L}$ on the variables:

$$M, \Gamma \models_{\mathcal{A}} \phi \quad \equiv \quad \forall \rho \in \mathsf{Env}(\mathcal{A}).\ \rho \models_{\mathcal{A}} \Gamma \quad \Rightarrow \quad [\![M]\!]^{\mathcal{A}}_{\rho} \models_{\mathcal{A}} \phi$$

where

$$\rho \ \models_{\mathcal{A}} \ \Gamma \quad \equiv \quad \forall x \in \mathsf{Var}.\ \rho x \models_{\mathcal{A}} \Gamma x.$$

We write

$$M, \Gamma \ \models \ \phi \quad \equiv \quad \forall \mathcal{A}.\ M, \Gamma \models_{\mathcal{A}} \phi.$$

We now introduce a proof system for assertions of the form $M, \Gamma \ \vdash \ \phi$.

Notation: $\Gamma \leq \Delta \equiv \forall x \in \mathsf{Var}. \Gamma x \leq \Delta x$.

Proof System For Program Logic

$$M, \Gamma \vdash t$$

$$\frac{M, \Gamma \vdash \phi \quad M, \Gamma \vdash \psi}{M, \Gamma \vdash \phi \wedge \psi}$$

$$\frac{\Gamma \leq \Delta \quad M, \Delta \vdash \phi \quad \phi \leq \psi}{M, \Gamma \vdash \psi}$$

$$x, \Gamma[x \mapsto \phi] \vdash \phi$$

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

$$\frac{M, \Gamma \vdash (\phi \rightarrow \psi)_\perp \quad N, \Gamma \vdash \phi}{MN, \Gamma \vdash \psi}.$$

Theorem 5.8 (Soundness of Program Logic) *For all M, Γ, ϕ:*

$$M, \Gamma \vdash \phi \implies M, \Gamma \models \phi. \quad \blacksquare$$

The proof is again routine. Note the striking similarity of our program logic with type inference, in particular with the intersection type discipline and Extended Applicative Type Structures of [CDHL84]. The crucial *difference* lies in the entailment relation $\leq$, and in particular the fact that their axiom (in our notation)

$$t \leq (t \rightarrow t)_\perp$$

is *not* a theorem in our logic; instead, we have the weaker $((\rightarrow)_\perp)$. This reflects a different notion of "function space"; we discuss this further in section 7.

We now come to the expected connection between the domain logic $\mathcal{L}$ and the domain D. The connecting link is the domain equation used to define D, and from which $\mathcal{L}$ is derived. Since this equation corresponds to the type expression $\sigma \equiv \mathsf{rec}\, t.(t \rightarrow t)_\perp$, it falls within the scope of the general theory developed in [Abr87b]. The logic $\mathcal{L}$ presented in this section is a streamlined version of $\mathcal{L}(\sigma)$ as defined in [Abr87b]. Once we have shown that $\mathcal{L}$ is equivalent to $\mathcal{L}(\sigma)$, we can apply the results of [Abr87b] to obtain the desired relationships between $\mathcal{L} \simeq \mathcal{L}(\sigma)$ and $D \simeq D(\sigma)$.[1]

Firstly, note that $\mathcal{L}$ as presented contains no disjunctive structure, while the constructs $\rightarrow$, $(\cdot)_\perp$ appearing in σ generate no inconsistencies according to the definition of C in [Abr87b]. Thus (the Lindenbaum algebra of) $\mathcal{L}_\wedge(\sigma)$, the purely conjunctive part of $\mathcal{L}(\sigma)$, is a meet-semilattice, and applying [Abr87b, Theorem 2.3.4], we obtain

$$\mathsf{Spec}\,(\mathcal{L}(\sigma)/{=_\sigma}, {\leq_\sigma}/{=_\sigma}) \;\cong\; \mathsf{Filt}(\mathcal{L}_\wedge(\sigma)/{=_\sigma}, {\leq_\sigma}/{=_\sigma}).$$

It remains to show that $\mathcal{L}$ is pre-isomorphic to $\mathcal{L}_\wedge(\sigma)$. We can describe the syntax of $\mathcal{L}_\wedge(\sigma)$ as follows:

[1] The reader unfamiliar with [Abr87b] who is prepared to take Theorems 5.12 and 5.14 on trust is advised to skip the details till after 5.14.

- $L_\wedge(\sigma)$:

$$\phi \ ::= \ t \mid \phi \wedge \psi \mid (\phi)_\perp \ \ (\phi \in L(\sigma \to \sigma))$$

- $L_\wedge(\sigma \to \sigma)$:

$$\phi \ ::= \ t \mid \phi \wedge \psi \mid (\phi \to \psi) \ \ (\phi, \psi \in L(\sigma)).$$

Using $(()_\perp - \wedge)$ and $(\to -t)$ (i.e. the nullary instances of $(\to -\wedge)$) from [Abr87b], we obtain the following normal forms for $L_\wedge(\sigma)$:

$$\phi \ ::= \ t \mid \phi \wedge \psi \mid (\phi \to \psi)_\perp.$$

In this way we see that $L \subseteq L_\wedge(\sigma)$, and that each $\phi \in L_\wedge(\sigma)$ is equivalent to one in L. Moreover, the axioms and rules of $\mathcal{L}$ are easily seen to be derivable in $\mathcal{L}_\wedge(\sigma)$. For example, $((\to)_\perp - t)$ is derivable, since

$$\mathcal{L}_\wedge(\sigma) \ \vdash \ (\phi \to \psi)_\perp = (t)_\perp = (t \to t)_\perp.$$

It remains to show the converse, i.e. that for $\phi, \psi \in \mathcal{L}$:

$$\mathcal{L}_\wedge(\sigma) \ \vdash \ \phi \leq \psi \ \implies \ \mathcal{L} \ \vdash \ \phi \leq \psi.$$

For this purpose, we use $((\to)_\perp - \wedge)$ and $((\to)_\perp - t)$ to get normal forms for $\mathcal{L}$.

Lemma 5.9 (Normal Forms) *Every formula in $\mathcal{L}$ is equivalent to one in $N\mathcal{L}$, where:*

- $N\mathcal{L} = \{\bigwedge_{i\in I} \phi_i : I \text{ finite}, \ \phi_i \in SN\mathcal{L}, \ i \in I\}$
- $SN\mathcal{L} = \{(\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp : k \geq 0, \phi_i \in N\mathcal{L}, \ 1 \leq i \leq k\}$. ∎

Now by [Abr87b, Propositions 3.4.5 and 3.4.6], we have

Lemma 5.10 *For ϕ, ψ with*

$$\phi \ \equiv \ \bigwedge_{i\in I} (\phi_i \to \phi'_i)_\perp, \qquad \psi \ \equiv \ \bigwedge_{j \in J} (\psi_j \to \psi'_j)_\perp :$$

$$\mathcal{L}(\sigma) \ \vdash \ \phi \leq \psi \iff \forall j \in J.\, \mathcal{L}(\sigma) \ \vdash \ \bigwedge\{\phi'_i \ : \ \mathcal{L}(\sigma) \ \vdash \ \psi_j \leq \phi_i\} \leq \psi'_j.$$

Proposition 5.11 *For $\phi, \psi \in N\mathcal{L}$, if $\mathcal{L}(\sigma) \ \vdash \ \phi \leq \psi$ then there is a proof of $\phi \leq \psi$ using only the meet-semilattice laws and the derived rule $((\to)_\perp)$.*

Proof. By induction on the complexity of ϕ and ψ, and the preceding Lemma. ∎

We have thus shown that

$$\mathcal{L}(\sigma) \ \cong \ \mathcal{L}_\wedge(\sigma) \ \cong \ \mathcal{L},$$

and we can apply the Duality Theorem of [Abr87b] to obtain

Theorem 5.12 (Stone Duality) *$\mathcal{L}$ is the Stone dual of D:*

$$
\begin{array}{ll}
(i) & D \;\cong\; \mathsf{Filt}\,\mathcal{L} \\
(ii) & (\mathcal{K}(D))^{op} \;\cong\; (L/=, \leq/=)
\end{array}
$$

(where $\mathcal{K}(D)$ is the sub-poset of finite elements of D).

Corollary 5.13 $D \;\models\; \phi \leq \psi \;\iff\; \mathcal{L} \;\vdash\; \phi \leq \psi.$

We can now deal with the program logic over λ-terms in a similar fashion. The denotational semantics for $\mathbf{\Lambda}$ in D given in the previous section can be used to define a translation map

$$(\cdot)^* : \mathbf{\Lambda} \to \mathbf{\Lambda}(\sigma).$$

The logic presented in this section is equivalent to the endogenous logic of [Abr87b] in the sense that

$$M, \Gamma \;\vdash\; \phi \;\iff\; M^*, \Gamma \;\vdash\; \phi$$

where $M \in \mathbf{\Lambda}$, $\Gamma : \mathsf{Var} \to L$, $\phi \in L \subseteq L(\sigma)$. We omit the details, which by now should be routine. As a consequence of this result, we can apply the Completeness Theorem for Endogenous Logic from [Abr87b], to obtain:

Theorem 5.14 *D is $\mathcal{L}$-complete, i.e. for all $M \in \mathbf{\Lambda}$, $\Gamma : \mathsf{Var} \to L$, $\phi \in L \subseteq L(\sigma)$:*

$$M, \Gamma \;\vdash\; \phi \;\iff\; M, \Gamma \;\models_D\; \phi.$$

In the previous section, we defined an lts over D; and we have now shown that D is isomorphic as a domain to $\mathsf{Filt}\,\mathcal{L}$. We can in fact describe the lts structure over $\mathsf{Filt}\,\mathcal{L}$ directly; and this will show how D, defined by a domain equation reminiscent of the D_∞ construction, can also be viewed as a graph model or "PSE algebra" in the terminology of [Lon83].

Notation. For $X \subseteq L$, $X^\dagger$ is the filter generated by X. This can be defined inductively by:

- $X \subseteq X^\dagger$
- $t \in X^\dagger$
- $\phi, \psi \in X^\dagger \;\Rightarrow\; \phi \wedge \psi \in X^\dagger$
- $\phi \in X^\dagger,\; \mathcal{L} \;\vdash\; \phi \leq \psi \;\Rightarrow\; \psi \in X^\dagger$.

Definition 5.15 The quasi-applicative structure with divergence

$$(\mathsf{Filt}\,\mathcal{L}, \cdot\,, \Uparrow)$$

is defined as follows:

$$
\begin{array}{lcl}
x\Uparrow & \equiv & x = \{t\} \\
x \cdot y & \equiv & \{\psi : \exists \phi.\, (\phi \to \psi)_\perp \in x \;\&\; \phi \in y\} \cup \{t\}.
\end{array}
$$

It is easily verified that in this structure

$$x \lesssim^B y \iff x \subseteq y,$$

and hence that application is monotone in each argument, and $\mathsf{Filt}\,\mathcal{L}$ is an ats. Thus we have an interpretation function

$$[\![\cdot]\!]^{\mathsf{Filt}\,\mathcal{L}} : CL(\mathsf{Filt}\,\mathcal{L}) \to Env(\mathsf{Filt}\,\mathcal{L}) \to \mathsf{Filt}\,\mathcal{L}$$

which is extended to $\mathbf{\Lambda}(\mathsf{Filt}\,\mathcal{L})$ by

$$[\![\lambda x.M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\rho} = \{(\phi \to \psi)_{\perp} : \psi \in [\![M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\rho[x \mapsto \uparrow\phi]}\}^{\dagger}.$$

We then define

Definition 5.16

$$\begin{aligned} s &\equiv [\![\lambda x.\lambda y.\lambda z.(xz)(yz)]\!]^{\mathsf{Filt}\,\mathcal{L}} \\ k &\equiv [\![\lambda x.\lambda y.x]\!]^{\mathsf{Filt}\,\mathcal{L}}. \end{aligned}$$

Proposition 5.17 $\mathsf{Filt}\,\mathcal{L}$ *is an lts. Moreover,* $\mathsf{Filt}\,\mathcal{L}$ *and* D *are isomorphic as combinatory algebras.*

Proof. It is sufficient to show that the isomorphism of the Duality Theorem preserves application, divergence and the denotation of λ-terms, since it then preserves s and k and so is a combinatory isomorphism, and $\mathsf{Filt}\,\mathcal{L}$ is an lts, since D is.

Firstly, we show that application is preserved, i.e. for $d_1, d_2 \in D$:

$$(\star) \;\; \mathcal{L}(d_1 \cdot d_2) = \mathcal{L}(d_1) \cdot \mathcal{L}(d_2)$$

The right to left inclusion follows by the same argument as the soundness of the rule for application in 5.7. For the converse, suppose $\psi \in \mathcal{L}(d_1 \cdot d_2)$, $\mathcal{L} \nvdash \psi = t$. By the Duality Theorem, each ψ in $\mathcal{L}$ corresponds to a unique $c \in \mathcal{K}(D)$ with $\mathcal{L}(c) = \uparrow\psi$. Since application is continuous in D, $c \sqsubseteq d_1 \cdot d_2$, $c \neq \perp$ implies that for some $b \in \mathcal{K}(D)$, $\mathsf{fold}(<0, [b, c]>) \sqsubseteq d_1$ and $b \sqsubseteq d_2$. (Here $[b, c]$ is the one step function mapping d to c if $b \sqsubseteq d$, and to $\perp$ otherwise). Let $\mathcal{L}(b) = \uparrow\phi$, then $(\phi \to \psi)_{\perp} \in \mathcal{L}(d_1)$ and $\phi \in \mathcal{L}(d_2)$, as required.

Next, we show that denotations of λ-terms are preserved, i.e. for all $M \in \mathbf{\Lambda}$, $\rho \in Env(D)$:

$$(\star\star) \;\; \mathcal{L}([\![M]\!]^{D}_{\rho}) = [\![M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\mathcal{L} \circ \rho}.$$

This is proved by induction on M. The case when M is a variable is trivial; the case for application uses $(\star)$. For abstraction, we argue by structural induction over $\mathcal{L}$. We show the non-trivial case. Let ϕ, b be paired in the isomorphism of the Duality Theorem. Then

$$\begin{aligned} & \lambda x.M,\, \rho \models_D (\phi \to \psi)_{\perp} \\ \iff & M,\, \rho[x \mapsto b] \models_D \psi \\ \iff & M,\, \mathcal{L}() \circ (\rho[x \mapsto b]) \models_{\mathsf{Filt}\,\mathcal{L}} \psi && \text{ind. hyp.} \\ \iff & M,\, (\mathcal{L}() \circ \rho)[x \mapsto \uparrow\phi] \models_{\mathsf{Filt}\,\mathcal{L}} \psi \\ \iff & \lambda x.M,\, \mathcal{L}() \circ \rho \models_{\mathsf{Filt}\,\mathcal{L}} (\phi \to \psi)_{\perp}. \end{aligned}$$

Finally, divergence is trivially preserved, since the only divergent elements in D, $\mathsf{Filt}\,\mathcal{L}$ are $\perp$, $\{t\}$, and these are in bi-unique correspondence under the isomorphism of the Duality Theorem. ∎

Theorem 5.18 (Computational Adequacy) *For all $M \in \mathbf{\Lambda}$,*

$$M{\Downarrow} \iff [\![M]\!]^D_{\rho_\perp} \neq \perp$$

where $\rho_\perp : x \mapsto \perp$.

PROOF. Firstly, let $\sigma_{\mathbf{\Omega}} : x \mapsto \mathbf{\Omega}$. $M{\Downarrow} \;\Rightarrow\; (M\sigma_{\mathbf{\Omega}}){\Downarrow}\lambda x.N \;\Rightarrow\; [\![M]\!]^D_{\rho_\perp} = [\![\lambda x.N]\!]^D_{\rho_\perp} \neq \perp$. For the converse, let $\Gamma_t : x \mapsto t$.

$$\begin{array}{rcll} [\![M]\!]^D_{\rho_\perp} \neq \perp & \Rightarrow & [\![M]\!]^{\mathsf{Filt}\,\mathcal{L}}_{\rho_\perp} \neq \{t\} & \text{by 5.17} \\ & \Rightarrow & M, \Gamma_t \vdash \lambda & \\ & \Rightarrow & M, \Gamma_t \models \lambda & \text{by 5.8} \\ & \Rightarrow & M{\Downarrow}. & \blacksquare \end{array}$$

The triviality of this proof is notable, since analogous results in the literature have required lengthy arguments involving recursively defined inclusive predicates (*cf.* [Plo85]).

We can now proceed in exact analogy to [Abr87a], and use Stone Duality to convert the Characterisation Theorem into a Final Algebra Theorem.

Definition 5.19 We define a number of categories of transition systems:

ATS Objects: applicative transition systems; morphisms $\mathcal{A} \to \mathcal{B}$: maps $f : A \to B$ satisfying

$$a \models_{\mathcal{A}} \phi \iff f(a) \models_{\mathcal{B}} \phi.$$

LTS The subcategory of **ATS** of lts and morphisms which preserve application, s and k.

CLTS The full subcategory of **LTS** of those $\mathcal{A}$ satisfying *continuity*:

$$\psi \neq t,\; ab \models_{\mathcal{A}} \psi \implies \exists \phi.\, a \models_{\mathcal{A}} (\phi \to \psi)_\perp \;\&\; b \models_{\mathcal{A}} \phi,$$

and also

$$\mathcal{L}(s) = [\![s]\!]^{\mathsf{Filt}\,\mathcal{L}},\;\; \mathcal{L}(k) = [\![k]\!]^{\mathsf{Filt}\,\mathcal{L}}.$$

Note that continuity implies approximability.

Theorem 5.20 (Final Algebra) *(i) D is final in* **ATS**.
(ii) Let $\mathcal{A}$ be an approximable lts. The map

$$\mathsf{t}_{\mathcal{A}} : \mathcal{A} \to D$$

from (i) is an **LTS** *morphism iff $\mathcal{A}$ is continuous.*
(iii) D is final in **CLTS**.

PROOF. (i). Given $\mathcal{A}$ in **ATS**, define

$$\mathsf{t}_{\mathcal{A}} : \mathcal{A} \to D$$

by

$$\mathsf{t}_{\mathcal{A}} \equiv \mathcal{A} \overset{\mathcal{L}()}{\to} \mathsf{Filt}\,\mathcal{L} \overset{\eta}{\to} D$$

where η is the isomorphism from the Stone Duality Theorem. For $a \in A$,

$$\mathcal{L}(a) = \mathcal{L} \circ \eta \circ \mathcal{L}(a) = \mathcal{L} \circ \mathsf{t}_{\mathcal{A}}(a),$$

and so $\mathsf{t}_{\mathcal{A}}$ is an **ATS** morphism; moreover, it is unique, since for $d, d' \in D$:

$$\mathcal{L}(d) = \mathcal{L}(d') \;\Rightarrow\; \mathcal{K}(d) = \mathcal{K}(d') \;\Rightarrow\; d = d'.$$

(ii). That $\mathcal{L}()$ is a combinatory morphism iff $\mathcal{A}$ is in **CLTS** is an immediate consequence of the definitions; the result then follows from the fact that η is a combinatory isomorphism.
(iii). Immediate from (ii). ∎

Note that if $\mathcal{A}$ is approximable, we have:

$$a \precsim^B b \iff \mathsf{t}_{\mathcal{A}}(a) \precsim^B \mathsf{t}_{\mathcal{A}}(b).$$

Thus we can regard the Final Algebra Theorem as giving a syntax-free fully abstract semantics for approximable ats. However, from the point of view of applications to programming language semantics, this is not very useful. In the next section, we shall study full abstraction in a syntax-directed framework, using our domain logic as a tool.

6 Lambda Transition Systems considered as Programming Languages

The classical discussion of full abstraction in the λ-calculus [Plo77, Mil77] is set in the typed λ-calculus with ground data. As remarked in the Introduction, this material has not to date been transferred successfully to the pure untyped λ-calculus. To see why this is so, let us recall some basic notions from [Plo77, Mil77].

Firstly, there is a natural notion of *program*, namely closed term of ground type. Programs either diverge, or yield a ground constant as result. This provides a natural notion of observable behaviour for programs, and hence an operational order on them. This is extended to arbitrary terms via ground contexts; in other words, the point of view is taken that only program behaviour is directly observable, and the meaning of a higher-type term lies in the observable behaviour of the programs into which it can be embedded. Thus both the presence of ground data, and the fact that terms are typed, enter into the basic definitions of the theory.

By contrast, we have a notion of atomic observation for the lazy λ-calculus in the absence of types or ground data, namely convergence to weak head normal form. This leads to the applicative bisimulation relation, and hence to a natural operational ordering. We can thus develop a theory of full abstraction in the pure untyped λ-calculus. Our results will correspond recognisably to those in [Plo77], although the technical details contain many differences. One feature of our development is that we work axiomatically with classes of lts under various hypotheses, rather than with particular languages. (Note that operational transition systems and "programming languages" such as $\lambda\ell$ actually *are* lts under our definitions.)

Definition 6.1 Let $\mathcal{A}$ be an lts. D is *fully abstract* for $\mathcal{A}$ if $\Im(\mathcal{A}) = \Im(D)$.

This definition is consistent with that in [Plo77, Mil77], provided we accept the applicative bisimulation ordering on $\mathcal{A}$ as the appropriate operational preorder. The argument for doing so is made highly plausible by Proposition 2.5, which characterises applicative bisimulation as a contextual preorder analogous to those used in [Plo77, Mil77]. We shall prove 2.5 later in this section.

We now turn to the question of conditions under which D is fully abstract for $\mathcal{A}$. As emerges from [Plo77, Mil77], this is essentially a question of definability.

Definition 6.2 An ats $\mathcal{A}$ is *$\mathcal{L}$-expressive* if for all $\phi \in \mathcal{L}$, for some $a \in \mathcal{A}$:

$$\mathcal{L}(a) = \uparrow\phi \;\equiv\; \{\psi \in \mathcal{L} \;:\; \mathcal{L} \vdash \; \phi \leq \psi\}.$$

In the light of Stone Duality, $\mathcal{L}$-expressiveness can be read as: "all finite elements of D are definable in $\mathcal{A}$".

Definition 6.3 Let $\mathcal{A}$ be an ats.

- *Convergence testing* is definable in $\mathcal{A}$ if for some $c \in A$, $\mathcal{A}$ satisfies:
  - $c\Downarrow$

- $x\Uparrow \;\Rightarrow\; cx\Uparrow$
- $x\Downarrow \;\Rightarrow\; cx = \mathbf{I}$.

In this case, we use C as a constant to denote c.

- *Parallel convergence* is definable in $\mathcal{A}$ if for some $p \in A$, $\mathcal{A}$ satisfies:
  - $p\Downarrow,\;\; px\Downarrow$
  - $x\Downarrow \;\Rightarrow\; pxy\Downarrow$
  - $y\Downarrow \;\Rightarrow\; pxy\Downarrow$
  - $x\Uparrow \,\&\, y\Uparrow \;\Rightarrow\; pxy\Uparrow$.

  In this case, we use P to denote such a p.

Note that if C is definable, it is unique (up to bisimulation); this is not so for P.

The notion of parallel convergence is reminiscent of Plotkin's parallel or, and will play a similar role in our theory. (A sharper comparison will be made later in this section.) The notion of convergence testing is less expected. We can think of the combinator C as a sort of "1-strict" version of $\mathbf{F} \equiv \lambda x.\lambda y.y$:

$$\mathsf{C}xy = \mathbf{K}xy = y \quad \text{if } x\Downarrow$$

$$\mathsf{C}xy\Uparrow \quad \text{if } x\Uparrow.$$

This 1-strictness allows us to test, sequentially, a number of expressions for convergence. Under the hypothesis that C is definable, we can give a very satisfactory picture of the relationship between all these notions.

Theorem 6.4 (Full Abstraction) *Let $\mathcal{A}$ be a sensible, approximable lts in which C is definable. The following conditions are equivalent:*

(i) *Parallel convergence is definable in $\mathcal{A}$.*

(ii) *$\mathcal{A}$ is $\mathcal{L}$-expressive.*

(iii) *$\mathcal{A}$ is $\mathcal{L}$-complete.*

(iv) *$\mathsf{t}_{\mathcal{A}}$ is a combinatory embedding with $\mathcal{K}(D) \subseteq Im\, \mathsf{t}_{\mathcal{A}}$.*

(v) *D is fully abstract for $\mathcal{A}$.*

Proof. We shall prove a sequence of implications to establish the theorem, indicating in each case which hypotheses on $\mathcal{A}$ are used.
(i) $\Longrightarrow$ *(ii)* ($\mathcal{A}$ sensible, C definable).

Since $\mathcal{A}$ is sensible, $\mathbf{\Omega}$ diverges in $\mathcal{A}$.
Notation. Given a set Con of constants, $\mathbf{\Lambda}(\mathsf{Con})$ is the set of λ-terms over Con.

For each $\phi \in N\mathcal{L}$ we shall define terms $M_\phi, T_\phi \in \mathbf{\Lambda}(\{\mathsf{P}, \mathsf{C}\})$ such that, for all $\psi \in N\mathcal{L}$:

- $M_\phi \models_{\mathcal{A}} \psi \iff \mathcal{L} \vdash \phi \leq \psi$
- $\begin{cases} T_\phi M_\psi \Downarrow & \text{if } M_\psi \models_{\mathcal{A}} \phi, \\ T_\phi M_\psi \Uparrow & \text{otherwise.} \end{cases}$

The definition is by induction on the complexity of

$$\phi \equiv \bigwedge_{i \in I} (\phi_{i,1} \to \cdots (\phi_{i,k_i} \to \lambda)_\perp \cdots)_\perp .$$

If $I = \varnothing$, $M_\phi \equiv \mathbf{\Omega}$. Otherwise, we define $M_\phi \equiv M(\phi, k)$, where $k = \max \{k_i \mid i \in I\}$:

$$\begin{array}{rcl} M(\phi, 0) & \equiv & \mathbf{K\Omega} \\ M(\phi, i+1) & \equiv & \lambda x_j .\, \mathsf{C} N^j M(\phi, i) \end{array}$$

where

$$\begin{array}{rcl} j & \equiv & k - i \\ N^j & \equiv & \sum \{N_i^j : j \leq k_i\} \\ N_i^j & \equiv & \mathsf{C}(T_{\phi_{i,1}} x_1)(\mathsf{C}(T_{\phi_{i,2}} x_2)(\ldots(\mathsf{C}(T_{\phi_{i,j}} x_j))\ldots)) \\ \sum \varnothing & \equiv & \mathbf{\Omega} \\ \sum \{N\} \cup \Theta & \equiv & \mathsf{P} N (\sum \Theta). \\ T_\phi & \equiv & \lambda x.\, \prod \{x M_{\phi_{i,1}} \ldots M_{\phi_{i,k_i}} : i \in I\} \\ \prod \varnothing & \equiv & \mathbf{K\Omega} \\ \prod \{N\} \cup \Theta & \equiv & \mathsf{C} N (\prod \Theta). \end{array}$$

We must show that these definitions have the required properties. Firstly, we prove for all $\phi \in N\mathcal{L}$:

$$\begin{array}{ll} (1) & M_\phi \models_{\mathcal{A}} \phi \\ (2) & a \models_{\mathcal{A}} \phi \;\Rightarrow\; T_\phi a \Downarrow \end{array}$$

by induction on ϕ:

- $\forall i \in I.\, a_j \models_{\mathcal{A}} \phi_{i,j} \;\; (1 \leq j \leq k_i)$

$\Rightarrow$ $M_\phi a_1 \ldots a_{k_i} \Downarrow$ by induction hypothesis (2),

$\therefore$ $M_\phi \vdash_{\mathcal{A}} \phi$.

- $a \models_{\mathcal{A}} \phi$

$\Rightarrow$ $T_\phi a \Downarrow$ by induction hypothesis (1).

We complete the argument by proving, for all $\phi, \psi \in N\mathcal{L}$:

$$
\begin{array}{lccc}
(3) & M_\phi \models_{\mathcal{A}} \psi & \Rightarrow & \mathcal{L} \vdash \phi \leq \psi \\
(4) & M_\psi \models_{\mathcal{A}} \phi & \Rightarrow & \mathcal{L} \vdash \psi \leq \phi \\
(5) & T_\phi M_\psi \Downarrow & \Rightarrow & M_\psi \models_{\mathcal{A}} \phi \\
(6) & T_\psi M_\phi \Downarrow & \Rightarrow & M_\phi \models_{\mathcal{A}} \psi.
\end{array}
$$

The proof is by induction on $n+m$, where n, m are the number of sub-formulae of ϕ, ψ respectively. Let

$$
\begin{array}{lcl}
\phi & \equiv & \bigwedge_{i \in I}(\phi_{i,1} \to \cdots (\phi_{i,k_i} \to \lambda)_\perp \cdots)_\perp, \\
\psi & \equiv & \bigwedge_{j \in J}(\psi_{j,1} \to \cdots (\psi_{j,k_j} \to \lambda)_\perp \cdots)_\perp.
\end{array}
$$

(3):

$$
\begin{array}{lll}
\bullet & M_\phi \models_{\mathcal{A}} \psi & \\
\Rightarrow & \forall j \in J.\, M_\phi M_{\psi_{j,1}} \ldots M_{\psi_{j,k_j}} \Downarrow & \text{by (1)}, \\
\Rightarrow & \forall j \in J.\, \exists i \in I.\, k_j \leq k_i \,\&\, T_{\phi_{i,l}} M_{\psi_{j,l}} \Downarrow,\ \ 1 \leq l \leq k_j & \\
\Rightarrow & M_{\psi_{j,l}} \models_{\mathcal{A}} \phi_{i,l},\ \ 1 \leq l \leq k_j & \text{ind. hyp. (5)} \\
\Rightarrow & \mathcal{L} \vdash \psi_{j,l} \leq \phi_{i,l},\ \ 1 \leq l \leq k_j & \text{ind. hyp. (4)} \\
\Rightarrow & \mathcal{L} \vdash \phi \leq \psi. &
\end{array}
$$

(4): Symmetrical to (3).
(5):

$$
\begin{array}{lll}
\bullet & T_\phi M_\psi \Downarrow & \\
\Rightarrow & \forall i \in I.\, M_\psi M_{\phi_{i,1}} \ldots M_{\phi_{i,k_i}} \Downarrow & \\
\Rightarrow & \forall i \in I.\, \exists j \in J.\, k_i \leq k_j \,\&\, T_{\psi_{j,l}} M_{\phi_{i,l}} \Downarrow,\ \ 1 \leq l \leq k_i & \\
\Rightarrow & M_{\phi_{i,l}} \models_{\mathcal{A}} \psi_{j,l},\ \ 1 \leq l \leq k_i & \text{ind. hyp. (6)} \\
\Rightarrow & \mathcal{L} \vdash \phi_{i,l} \leq \psi_{j,l},\ \ 1 \leq l \leq k_i & \text{ind. hyp. (3)} \\
\Rightarrow & \mathcal{L} \vdash \psi \leq \phi & \\
\Rightarrow & M_\psi \models_{\mathcal{A}} \phi & \text{by (1).}
\end{array}
$$

(6): Symmetrical to (5).
$(ii) \implies (iii)$ ($\mathcal{A}$ approximable).
Notation. For each $\phi \in \mathcal{L}$, $a_\phi \in A$ is the element representing ϕ. Given $\Gamma : \mathsf{Var} \to \mathcal{L}$, $\rho_\Gamma \in Env(\mathcal{A})$ is defined by

$$\rho_\Gamma x = a_{\Gamma x}.$$

Finally, $\Gamma_t : \mathsf{Var} \to \mathcal{L}$ is the constant map $x \mapsto t$.

We begin with some preliminary results.

$$(1)\ \ \mathcal{A} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

One half is the Soundness Theorem for $\mathcal{L}$. For the converse, note that

$$\begin{aligned} \mathcal{A} \models \phi \leq \psi &\Rightarrow a_\phi \models_{\mathcal{A}} \psi \\ &\Rightarrow \mathcal{L} \vdash \phi \leq \psi. \end{aligned}$$

$$(2)\ \ \forall \psi \in N\mathcal{L}.\, \psi \neq t \,\&\, ab \models_{\mathcal{A}} \psi \Rightarrow \exists \phi.\, a \models_{\mathcal{A}} (\phi \to \psi)_\perp \,\&\, b \models_{\mathcal{A}} \phi.$$

This is shown by induction on ψ.

$$\begin{aligned} \bullet\ \ & ab \models_{\mathcal{A}} \textstyle\bigwedge_{i\in I} \psi_i \ \ (I \neq \varnothing) \\ \Rightarrow\ \ & \forall i \in I.\, ab \models_{\mathcal{A}} \psi_i \\ \Rightarrow\ \ & \forall i \in I.\, \exists \phi_i.\, a \models_{\mathcal{A}} (\phi_i \to \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \phi_i \quad \text{by ind. hyp.} \\ \Rightarrow\ \ & \forall i \in I.\, a \models_{\mathcal{A}} (\textstyle\bigwedge_{i\in I} \phi_i \to \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \textstyle\bigwedge_{i\in I} \phi_i \\ \Rightarrow\ \ & a \models_{\mathcal{A}} (\textstyle\bigwedge_{i\in I} \phi_i \to \textstyle\bigwedge_{i\in I} \psi_i)_\perp \,\&\, b \models_{\mathcal{A}} \textstyle\bigwedge_{i\in I} \phi_i. \end{aligned}$$

$$\begin{aligned} \bullet\ \ & ab \models_{\mathcal{A}} (\psi_1 \to \cdots (\psi_k \to \lambda)_\perp \cdots)_\perp \\ \Rightarrow\ \ & aba_{\psi_1} \ldots a_{\psi_k} \Downarrow \\ \Rightarrow\ \ & \exists \phi, \phi_1, \ldots, \phi_k.\, b \models_{\mathcal{A}} \phi \,\&\, a_{\psi_i} \models_{\mathcal{A}} \phi_i \ (1 \leq i \leq k) \\ & \&\, a \models_{\mathcal{A}} (\phi \to (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp, \\ & \text{since A is approximable} \\ \Rightarrow\ \ & \mathcal{L} \vdash \psi_i \leq \phi_i \ (1 \leq i \leq k) \\ \Rightarrow\ \ & \mathcal{L} \vdash (\phi \to (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\ & \ \leq (\phi \to (\psi_1 \to \cdots (\psi_k \to \lambda)_\perp \cdots)_\perp \\ \Rightarrow\ \ & a \models_{\mathcal{A}} (\phi \to \psi)_\perp \,\&\, b \models_{\mathcal{A}} \phi. \end{aligned}$$

$$(3)\ \ \forall M \in \mathbf{\Lambda}.\, M, \Gamma \models_{\mathcal{A}} \phi \iff M, \rho_\Gamma \models_{\mathcal{A}} \phi.$$

The right to left implication is clear, since $\rho_\Gamma \models_{\mathcal{A}} \Gamma$. We prove the converse by induction on M.

$$\begin{aligned} x, \Gamma \models_{\mathcal{A}} \phi &\iff \mathcal{A} \models \Gamma x \leq \phi \\ &\iff \mathcal{L} \vdash \Gamma x \leq \phi \ \ \text{by}(1) \\ &\iff a_{\Gamma x} \models_{\mathcal{A}} \phi \\ &\iff x, \rho_\Gamma \models_{\mathcal{A}} \phi. \end{aligned}$$

The case for $\lambda x.M$ is proved by induction on ϕ. We show the non-trivial case.

$$\begin{array}{ll} \bullet & \lambda x.M, \rho_\Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp \\ \Longrightarrow & M, \rho_\Gamma[x \mapsto a_\phi] \models_{\mathcal{A}} \psi \\ \Longrightarrow & M, \Gamma[x \mapsto \phi] \models_{\mathcal{A}} \psi \qquad \text{by (outer) induction hypothesis} \\ \Longrightarrow & \lambda x.M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp. \end{array}$$

$$\begin{array}{lll} \bullet & MN, \rho_\Gamma \models_{\mathcal{A}} \psi & \\ \Longrightarrow & [\![M]\!]^{\mathcal{A}}_{\rho_\Gamma}[\![N]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} \psi & \\ \Longrightarrow & \exists \phi.\, [\![M]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} (\phi \to \psi)_\perp \;\&\; [\![N]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} \phi & \text{by (2)} \\ \Longrightarrow & M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp \;\&\; N, \Gamma \models_{\mathcal{A}} \phi & \text{ind. hyp.} \\ \Longrightarrow & MN, \Gamma \models_{\mathcal{A}} \psi. & \end{array}$$

(4):

$$\begin{array}{rrcl} (i) & x, \Gamma[x \mapsto \phi] \models_{\mathcal{A}} \psi & \iff & \mathcal{L} \vdash \phi \leq \psi \\ (ii) & \lambda x.M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp & \iff & M, \Gamma[x \mapsto \phi] \models_{\mathcal{A}} \psi \\ (iii) & MN, \Gamma \models_{\mathcal{A}} \psi & \iff & \exists \phi.\, M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp \\ & & & \& \; N, \Gamma \models_{\mathcal{A}} \phi. \end{array}$$

4(i) is proved using (1).
4(ii):

$$\begin{array}{ll} \bullet & \lambda x.M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp \\ \Rightarrow & \forall \rho, a.\, \rho \models_{\mathcal{A}} \Gamma \;\&\; a \models_{\mathcal{A}} \phi \;\Rightarrow\; [\![\lambda x.M]\!]^{\mathcal{A}}_{\rho}.a \models_{\mathcal{A}} \psi \\ \Rightarrow & \forall \rho.\, \rho \models_{\mathcal{A}} \Gamma[x \mapsto \phi] \;\Rightarrow\; M, \rho \models_{\mathcal{A}} \psi \\ & \quad \text{since } [\![\lambda x.M]\!]^{\mathcal{A}}_{\rho}.a = [\![M]\!]^{\mathcal{A}}_{\rho[x \mapsto a]}, \\ \Rightarrow & M, \Gamma[x \mapsto \phi] \models_{\mathcal{A}} \psi. \end{array}$$

The converse follows from the soundness of $\mathcal{L}$.
4(iii):

$$\begin{array}{rcll} MN, \Gamma \models_{\mathcal{A}} \psi & \iff & MN, \rho_\Gamma \models_{\mathcal{A}} \psi & \text{by (3)} \\ & \iff & [\![M]\!]^{\mathcal{A}}_{\rho_\Gamma}[\![N]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} \psi & \\ & \iff & \exists \phi.\, [\![M]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} (\phi \to \psi)_\perp \;\&\; [\![N]\!]^{\mathcal{A}}_{\rho_\Gamma} \models_{\mathcal{A}} \phi & \text{by (2)} \\ & \iff & \exists \phi.\, M, \Gamma \models_{\mathcal{A}} (\phi \to \psi)_\perp \;\&\; N, \Gamma \models_{\mathcal{A}} \phi & \text{by (3)} \end{array}$$

We can now prove

$$M, \Gamma \models_{\mathcal{A}} \phi \;\Rightarrow\; M, \Gamma \vdash \phi$$

by induction on M, using (4).
$(iii) \implies (i)$.

Firstly, note that (iii) implies

$$\mathcal{A} \models \phi \leq \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

One half is the Soundness Theorem. For the converse, suppose $\mathcal{A} \models \phi \leq \psi$ and $\mathcal{L} \nvdash \phi \leq \psi$. Then $\mathbf{I} \models_{\mathcal{A}} (\phi \to \psi)_\perp$ but $\mathbf{I} \nvdash (\phi \to \psi)_\perp$, and so $\mathcal{A}$ is not $\mathcal{L}$-complete.

Now suppose that P is not definable in $\mathcal{A}$, and consider

$$\phi \equiv (\lambda \to (t \to \lambda)_\perp)_\perp \wedge (t \to (\lambda \to \lambda)_\perp)_\perp,$$

$$\psi \equiv (t \to (t \to \lambda)_\perp)_\perp.$$

Clearly, $\mathcal{L} \nvdash \phi \leq \psi$. However, for $a \in \mathcal{A}$, if $a \models_{\mathcal{A}} \phi$, then $x\Downarrow$ or $y\Downarrow$ implies $axy\Downarrow$; since P is not definable in $\mathcal{A}$, and in particular, a does not define P, we must have $axy\Downarrow$ even if $x\Uparrow$ and $y\Uparrow$, and hence $a \models_{\mathcal{A}} \psi$. Thus $\mathcal{A} \models \phi \leq \psi$ and so by our opening remark, $\mathcal{A}$ is not $\mathcal{L}$-complete.
$(ii) \implies (iv)$ ($\mathcal{A}$ approximable).

Clearly $\mathsf{Im}\ t_{\mathcal{A}} \supseteq \mathcal{K}(D)$, by 5.14(ii). Also, since $\mathcal{A}$ is approximable, we can apply the Characterisation Theorem to deduce that $t_{\mathcal{A}}$ is injective (modulo bisimulation). To show that $t_{\mathcal{A}}$ is a combinatory morphism, we argue as in 5.17. Application is preserved by $t_{\mathcal{A}}$ using (2) from the proof of $(ii) \Rightarrow (iii)$ and 5.17. The proof is completed by showing that $t_{\mathcal{A}}$ preserves denotations of λ-terms, i.e.

$$\forall M \in \mathbf{\Lambda}, \rho \in Env(\mathcal{A}).\, t_{\mathcal{A}}([\![M]\!]^{\mathcal{A}}_{\rho}) = [\![M]\!]^{D}_{t_{\mathcal{A}} \circ \rho}.$$

The proof is by induction on M. Since it is very similar to the corresponding part of the proof of 5.17, we omit it. The only non-trivial point is that in the case for abstraction we need:

$$\forall a \in A.\, a \models_{\mathcal{A}} \phi \implies M, \rho[x \mapsto a] \models_{\mathcal{A}} \psi$$

if and only if

$$M, \rho[x \mapsto a_\phi] \models_{\mathcal{A}} \psi,$$

which is proved similarly to (3) in $(ii) \Rightarrow (iii)$.
$(iv) \implies (v)$.

Assuming (iv), $\mathcal{A}$ is isomorphic (modulo bisimulation) to a substructure of D. Since formulas in HF are (equivalent to) universal (Π^0_1) sentences, this yields $\Im(D) \subseteq \Im(\mathcal{A})$. Since $\mathcal{K}(D) \subseteq \mathsf{Im} t_{\mathcal{A}}$, to prove the converse it is sufficient to show, for $H \in \mathsf{HF}$:

$$D, \rho \not\models H \implies \exists \rho_0 : \mathsf{Var} \to \mathcal{K}(D).\, D, \rho_0 \not\models H.$$

Let $H \equiv P \Rightarrow F$, where $P \equiv \bigwedge_{i \in I} M_i\Downarrow \wedge \bigwedge_{j \in J} N_j\Uparrow$. There are four cases, corresponding to the form of F.

Case 1: $F \equiv M \sqsubseteq N$. $D, \rho \not\models P \Rightarrow F$ implies $D, \rho \models P$ and $D, \rho \not\models M \sqsubseteq N$. Since D is algebraic, $D, \rho \not\models M \sqsubseteq N$ implies that for some $b \in \mathcal{K}(D)$, $b \sqsubseteq [\![M]\!]^D_\rho$ and $b \not\sqsubseteq [\![N]\!]^D_\rho$. Since the expression $[\![M]\!]^D_\rho$ is continuous in ρ, $b \sqsubseteq [\![M]\!]^D_\rho$ implies that for some $\rho_1 : \mathsf{Var} \to \mathcal{K}(D)$, $\rho_1 \sqsubseteq \rho$ and $b \sqsubseteq [\![M]\!]^D_{\rho_1}$. For all ρ' with $\rho_1 \sqsubseteq \rho' \sqsubseteq \rho$, $[\![N]\!]^D_{\rho'} \sqsubseteq [\![N]\!]^D_\rho$, and hence $b \not\sqsubseteq [\![N]\!]^D_{\rho'}$. Again, since D is algebraic,

$$D, \rho \models M_i \Downarrow \implies \exists \rho_i : \mathsf{Var} \to \mathcal{K}(D).\, \rho_i \sqsubseteq \rho \;\&\; D, \rho_i \models M_i \Downarrow.$$

Now let $\rho_0 \equiv \bigsqcup_{i \in I} \rho_i \sqcup \rho_1$. This is well-defined since D is a lattice. Moreover, $\rho_0 \sqsubseteq \rho$, and $\rho_0 : \mathsf{Var} \to \mathcal{K}(D)$. Since $\rho_0 \sqsupseteq \rho_i$ $(i \in I)$, $D, \rho_0 \models M_i \Downarrow$; while since $\rho_0 \sqsubseteq \rho$, $D, \rho_0 \models N_j \Uparrow$ $(j \in J)$. Since $\rho_1 \sqsubseteq \rho_0 \sqsubseteq \rho$, $b \sqsubseteq [\![M]\!]^D_{\rho_0}$ and $b \not\sqsubseteq [\![N]\!]^D_{\rho_0}$, and so $D, \rho_0 \not\models M \sqsubseteq N$. Thus $D, \rho_0 \not\models P \Rightarrow F$, as required.

The remaining cases are proved similarly.

$(v) \implies (i)$ ($\mathcal{A}$ sensible).

Consider the formula

$$H \equiv x\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow \wedge\, x(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow \;\Rightarrow\; x\mathbf{\Omega\Omega}\Downarrow.$$

It is easy to see that $\mathcal{A} \models H$ iff P is not definable in $\mathcal{A}$. Since P is definable in D, the result follows. ∎

We now turn to the question of when the bisimulation preorder on an lts can be characterised by means of a contextual equivalence, as in [Bar84, Plo77, Mil77].

Definition 6.5 Let $\mathcal{A}$ be an lts, $X, Y \subseteq A$. Then X *separates* Y if:

$$\forall M, N \in \mathbf{\Lambda}^0(Y).\, \mathcal{A} \not\models M \sqsubseteq N \implies$$
$$\exists P_1, \ldots, P_k \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models MP_1 \ldots P_k \Downarrow \;\&\; \mathcal{A} \models NP_1 \ldots P_k \Uparrow.$$

In particular, if X separates A we say that it is a *separating set*. For example, A is always a separating set.

Proposition 6.6 *Let $\mathcal{A}$ be an approximable lts, and suppose X separates Y. Then*

$$\forall M, N \in \mathbf{\Lambda}^0(Y).\, \mathcal{A} \models M \sqsubseteq N \iff$$
$$\forall C[\cdot] \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models C[M]\Downarrow \;\Rightarrow\; \mathcal{A} \models C[N]\Downarrow.$$

Proof. Suppose $\mathcal{A} \not\models M \sqsubseteq N$. Then since X separates Y, for some $P_1, \ldots, P_k \in \mathbf{\Lambda}^0(X)$, $\mathcal{A} \models MP_1 \ldots P_k \Downarrow$ and $\mathcal{A} \models NP_1 \ldots P_k \Uparrow$. Let $C[\cdot] \equiv [\cdot]P_1 \cdots P_k$. For the converse, suppose $\mathcal{A} \models M \sqsubseteq N$ and $\mathcal{A} \models C[M]\Downarrow$. Since $\mathcal{A}$ is approximable and $\mathcal{A} \models C[M] = (\lambda x.C[x])M$, for some ϕ $\lambda x.C[x] \models_{\mathcal{A}} (\phi \to \lambda)_\perp$ and $M \models_{\mathcal{A}} \phi$. Since $\mathcal{A} \models M \sqsubseteq N$, by the Characterisation Theorem $N \models_{\mathcal{A}} \phi$, and so $\mathcal{A} \models C[N]\Downarrow$. ∎

As a first application of this Proposition, we have:

Proposition 6.7 *Let $\mathcal{A}$ be a sensible, approximable lts in which* C *and* P *are definable. Then* $\{\mathsf{C}, \mathsf{P}\}$ *is a separating set.*

PROOF. By the Full Abstraction Theorem, for each $\phi \in \mathcal{L}$ there is $M_\phi \in \mathbf{\Lambda}^0(\{\mathsf{C}, \mathsf{P}\})$ such that

$$M_\phi \models_{\mathcal{A}} \psi \iff \mathcal{L} \vdash \phi \leq \psi.$$

Now

$$\begin{array}{ll}
\bullet & \mathcal{A} \not\models M \sqsubseteq N \\
\Longrightarrow & \exists \phi.\, M \models_{\mathcal{A}} \phi \,\&\, N \not\models \phi, \ \text{ since A is approximable} \\
\Longrightarrow & \exists \phi_1, \ldots, \phi_k.\, M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
& \ \ \& \, N \not\models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
\Longrightarrow & M M_{\phi_1} \ldots M_{\phi_k}\Downarrow \,\&\, N M_{\phi_1} \ldots M_{\phi_k}\Uparrow. \ \blacksquare
\end{array}$$

The hypothesis of approximability has played a major part in our work. We now give a useful sufficient condition.

Definition 6.8 Let $\mathcal{A}$ be an lts, $X \subseteq A$. Then $\mathcal{A}$ is X-*sensible* if

$$\forall M \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models M\Downarrow \ \Rightarrow \ D \models M\Downarrow.$$

Here $[\![M]\!]^D$ is the denotation in D obtained by mapping each $a \in X$ to $t_{\mathcal{A}}(a)$. Note that if we extend our endogenous program logic to terms in $\mathbf{\Lambda}^0(X)$, with axioms

$$a, \Gamma \vdash \phi \ \ (\phi \in \mathcal{L}(a)),$$

then the Soundness and Completeness Theorems for D still hold, by a straightforward extension of the arguments used above.

Proposition 6.9 *Let $\mathcal{A}$ be an X-sensible lts. Then $\mathcal{A}$ is X-approximable, i.e.*

$$\begin{array}{c}
\forall M, N_1, \ldots, N_k \in \mathbf{\Lambda}^0(X).\, \mathcal{A} \models M N_1 \ldots N_k\Downarrow \ \Rightarrow \ \exists \phi_1, \ldots, \phi_k. \\
M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \,\&\, N_i \models_{\mathcal{A}} \phi_i, \ 1 \leq i \leq k.
\end{array}$$

PROOF.

$$\begin{array}{ll}
\bullet & \mathcal{A} \models M N_1 \ldots N_k\Downarrow \\
\Rightarrow & D \models M N_1 \ldots N_k\Downarrow \\
\Rightarrow & \exists \phi_1, \ldots, \phi_k.\, M \models_D (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
& \ \ \& \, N_i \models_D \phi_i, \ 1 \leq i \leq k, \ \text{since D is approximable} \\
\Rightarrow & \exists \phi_1, \ldots, \phi_k.\, M \vdash (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
& \ \ \& \, N_i \vdash \phi_i, \ 1 \leq i \leq k, \ \text{by extended Completeness} \\
\Rightarrow & \exists \phi_1, \ldots, \phi_k.\, M \models_{\mathcal{A}} (\phi_1 \to \cdots (\phi_k \to \lambda)_\perp \cdots)_\perp \\
& \ \ \& \, N_i \models_{\mathcal{A}} \phi_i, \ 1 \leq i \leq k, \ \text{by extended Soundness.} \ \blacksquare
\end{array}$$

In particular, if X generates $\mathcal{A}$ and $\mathcal{A}$ is X-sensible, then $\mathcal{A}$ is approximable. We now turn to a number of applications of these ideas to syntactically presented lts, i.e. "programming languages".

Firstly, we consider the lts $\ell = (\mathbf{\Lambda}^0, eval)$ defined in section 3 (and studied previously in section 2). Since ℓ is $\varnothing$-sensible by 3.11, and it is generated by $\varnothing$, it is approximable by 6.9. Since $\varnothing$ is a separating set for $\mathbf{\Lambda}^0$, we can apply 6.6 to obtain Theorem 2.5.

Next, we consider extensions of ℓ.

Definition 6.10 (i) ℓ_{C} is the extension of ℓ defined by

$$\ell_{\mathsf{C}} = (\mathbf{\Lambda}(\{\mathsf{C}\}), \_\Downarrow\_)$$

where $\Downarrow$ is the extension of the relation defined in 2.2 with the following rules:

$$\bullet\ \mathsf{C}\Downarrow\mathsf{C} \qquad \bullet\ \frac{M\Downarrow}{\mathsf{C}M\Downarrow\mathbf{I}}$$

(ii) ℓ_{P} is the extension $(\mathbf{\Lambda}(\{\mathsf{C}\}), \_\Downarrow\_)$ of ℓ with the rules

$$\bullet\ \mathsf{P}\Downarrow\mathsf{P} \qquad \bullet\ \mathsf{P}M\Downarrow\mathsf{P}M \qquad \bullet\ \frac{M\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}} \qquad \bullet\ \frac{N\Downarrow}{\mathsf{P}MN\Downarrow\mathbf{I}}$$

It is easy to see that the relation $\_\Downarrow\_$ as defined in both ℓ_{C} and ℓ_{P} is a partial function. Moreover, with these definitions the C and P combinators have the properties required by 6.3; while C is definable in ℓ_{P}, by

$$\mathsf{C}M \equiv \mathsf{P}MM.$$

Since ℓ_{C} is generated by $\{\mathsf{C}\}$, and ℓ_{P} by $\{\mathsf{P}\}$, these are separating sets. Thus to apply Theorem 6.6, we need only check that ℓ_{C} is C-sensible, and ℓ_{P} P-sensible.

To do this for ℓ_{C}, we proceed as follows. Define

$$c \equiv \{(\lambda \to (\phi \to \phi)_\perp)_\perp \mid \phi \in \mathcal{L}\}^\dagger \in \mathsf{Filt}\,\mathcal{L}.$$

Then it is easy to see that $c \subseteq t_{\mathcal{A}}(\mathsf{C})$, and by monotonicity and the Soundness Theorem,

$$[\![M[c/\mathsf{C}]]\!]^D \subseteq [\![M]\!]^D$$

for $M \in \mathbf{\Lambda}^0(\{\mathsf{C}\})$. Thus

$$(\star)\quad D \models M[c/\mathsf{C}]\Downarrow \implies D \models M\Downarrow.$$

Now we prove

$$\begin{aligned} (\star\star)\quad & \forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{C}\}). \\ & M\Downarrow N \implies [\![M[c/\mathsf{C}]]\!]^D = [\![N[c/\mathsf{C}]]\!]^D \;\&\; D \models N[c/\mathsf{C}]\Downarrow, \end{aligned}$$

which by $(\star)$ yields $\ell_{\mathsf{C}} \models M\Downarrow \;\Rightarrow\; D \models M\Downarrow$, as required. $(\star\star)$ is proved by a straightforward induction on the length of the proof that $M\Downarrow N$.

The argument for ℓ_{P} is similar, using

$$p \equiv \{(\lambda \to (t \to (\phi \to \phi)_\perp)_\perp)_\perp \wedge (t \to (\lambda \to (\psi \to \psi)_\perp)_\perp)_\perp : \phi, \psi \in \mathcal{L}\}^\dagger.$$

Altogether, we have shown

Theorem 6.11 (Contextual Equivalence) *(i)* $\forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{C}\})$:

$$\ell_{\mathsf{C}} \models M \sqsubseteq N \iff \forall C[\cdot] \in \mathbf{\Lambda}^0(\{\mathsf{C}\}).\, \ell_{\mathsf{C}} \models C[M]\Downarrow \;\Rightarrow\; \ell_{\mathsf{C}} \models C[N]\Downarrow.$$

(ii) $\forall M, N \in \mathbf{\Lambda}^0(\{\mathsf{P}\})$:

$$\ell_{\mathsf{P}} \models M \sqsubseteq N \iff \forall C[\cdot] \in \mathbf{\Lambda}^0(\{\mathsf{P}\}).\, \ell_{\mathsf{P}} \models C[M]\Downarrow \;\Rightarrow\; \ell_{\mathsf{P}} \models C[N]\Downarrow.$$

As a further application of these ideas, we have

Proposition 6.12 (Soundness of D) *If $\mathcal{A}$ is X-sensible, and X separates X in $\mathcal{A}$, then:*

$$\Im^0(D, X) \subseteq \Im^0(\mathcal{A}, X).$$

PROOF.

$$\begin{array}{ll}
\bullet & D \models M \sqsubseteq N \\
\Longrightarrow & \forall C[\cdot] \in \mathbf{\Lambda}^0(X).\, D \models C[M] \sqsubseteq C[N] \\
\Longrightarrow & D \models C[M]\Downarrow \;\Rightarrow\; D \models C[N]\Downarrow \\
\Longrightarrow & \mathcal{A} \models C[M]\Downarrow \;\Rightarrow\; \mathcal{A} \models C[N]\Downarrow \\
\Longrightarrow & \mathcal{A} \models M \sqsubseteq N.
\end{array}$$

The argument for formulae of other forms is similar. ∎

As an immediate corollary of this Proposition,

Proposition 6.13 *The denotational semantics of each of our languages is* sound *with respect to the operational semantics:*

$$\begin{array}{rl}
(i) & \Im^0(D) \subseteq \Im^0(\ell) \\
(ii) & \Im^0(D, \{\mathsf{C}\}) \subseteq \Im^0(\ell_{\mathsf{C}}, \{\mathsf{C}\}) \\
(iii) & \Im^0(D, \{\mathsf{P}\}) \subseteq \Im^0(\ell_{\mathsf{P}}, \{\mathsf{P}\}).
\end{array}$$

We now turn to the question of full abstraction for these languages. Since, as we have seen, ℓ_{P} is P-sensible, and hence sensible and approximable, and C and P are definable, we can apply the Full Abstraction Theorem to obtain

Proposition 6.14 *D is fully abstract for ℓ_{P}.*

We now use the sequential nature of ℓ and ℓ_{C} to obtain negative full abstraction results for these languages. This will require a few preliminary notions.

Definition 6.15 The *one-step reduction* relation $>$ over terms in $\mathbf{\Lambda}$ is the least satisfying the following axioms and rules:

$$\bullet\ (\lambda x.M)N > M[N/x] \qquad \bullet\ \frac{M > M'}{MN > M'N}$$

This is then extended to $\mathbf{\Lambda}(\{\mathsf{C}\})$ with the additional rules

$$\bullet \ \ \mathsf{C}(\lambda x.M) > \mathbf{I} \quad \bullet \ \ \mathsf{CC} > \mathbf{I} \quad \bullet \ \ \frac{M > M'}{\mathsf{C}M > \mathsf{C}M'}$$

We then define

$$\begin{array}{lrcl} \bullet & \gg & \equiv & \text{the reflexive, transitive closure of} > \\ \bullet & M\uparrow & \equiv & \exists\{M_n\}.\, M = M_0 \,\&\, \forall n.\, M_n > M_{n+1} \\ \bullet & M\not> & \equiv & M \notin \mathsf{dom}> \\ \bullet & M{\downarrow}N & \equiv & M \gg N \,\&\, N \not> . \end{array}$$

It is clear that $>$ is a partial function. Note that these relations are being defined over *all* terms, not just closed ones. For closed terms, these new notions are related to the evaluation predicate $\_\Downarrow\_$ as follows:

Proposition 6.16 *For* $M, N \in \mathbf{\Lambda}^0$ $(\mathbf{\Lambda}^0(\{\mathsf{C}\}))$*:*

$$\begin{array}{rrcl} (i) & M{\Downarrow}N & \Longleftrightarrow & M{\downarrow}N \\ (ii) & M{\Uparrow} & \Longrightarrow & M{\uparrow}. \end{array}$$

We omit the straightforward proof. The following proposition is basic; it says that "reduction commutes with substitution".

Proposition 6.17 $M \gg N \ \Rightarrow\ M[P/x] \gg N[P/x]$.

Proof. Clearly, it is sufficient to show:

$$M > N \ \Rightarrow\ M[P/x] > N[P/x].$$

This is proved by induction on M, and cases on why $M > N$. We give one case for illustration:

$$M \equiv (\lambda y.M_1)M_2 > N \equiv M_1[M_2/y].$$

We assume $x \neq y$; the other sub-case is simpler.

$$\begin{array}{rcll} M[P/x] & = & (\lambda y.M_1[P/x])M_2[P/x] & \\ & > & M_1[P/x][M_2[P/x]/y] & \\ & = & M_1[M_2/y][P/x] & \text{by [Bar84, 2.1.16]} \\ & = & N[P/x]. & \blacksquare \end{array}$$

Now we come to the basic sequentiality property of ℓ from which various non-definability results can be deduced.

Proposition 6.18 *For $M \in \mathbf{\Lambda}$, exactly one of the following holds:*

$$\begin{array}{rl} (i) & M\uparrow \\ (ii) & M \gg \lambda x.N \\ (iii) & M \gg xN_1 \ldots N_k \ (k \geq 0). \end{array}$$

PROOF. Since $>$ is a partial function, the computation sequence beginning with M is uniquely determined. Either it is infinite, yielding (i); or it terminates in a term N with $N \not>$, which must be in one of the forms (ii) or (iii). ∎

As a consequence of this proposition, we obtain

Theorem 6.19 C *is not definable in ℓ. Moreover, D is not fully abstract for ℓ.*

PROOF. We shall show that ℓ satisfies

$$(\star) \ \ x = \mathbf{I} \ \text{ or } \ [x\mathbf{\Omega}\Downarrow \iff x(\mathbf{K\Omega})\Downarrow].$$

Indeed, consider any term $M \in \mathbf{\Lambda}^0$. Either $M\Uparrow$, in which case $M\mathbf{\Omega}\Uparrow$ and $M(\mathbf{K\Omega})\Uparrow$, or $M\Downarrow$. In the latter case, by $(\Downarrow\eta)$ we have $\lambda\ell \models M = \lambda x.Mx$. Thus without loss of generality we may take M to be of the form $\lambda x.M'$, with $FV(M) \subseteq \{x\}$. Now applying the three previous propositions to M', we see that in case (i) of 6.18, $(\lambda x.M')\mathbf{\Omega}\Uparrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Uparrow$; in case (ii), $(\lambda x.M')\mathbf{\Omega}\Downarrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Downarrow$; finally in case (iii), if $k = 0$, $\lambda x.M' = \mathbf{I}$; while if $k > 0$, $(\lambda x.M')\mathbf{\Omega}\Uparrow$ and $(\lambda x.M')(\mathbf{K\Omega})\Uparrow$. Since $\mathsf{C} \neq \mathbf{I}$, $\mathsf{C}\mathbf{\Omega}\Uparrow$ and $\mathsf{C}(\mathbf{K\Omega})\Downarrow$, this shows that C is not definable. Moreover, $(\star)$ implies

$$(\star\star) \ \ x\mathbf{\Omega}\Uparrow \,\&\, x(\mathbf{K\Omega})\Downarrow \ \Rightarrow \ x = \mathbf{I}$$

which is not satisfied by D, since C is definable in D, and taking $x = \mathsf{C}$ refutes $(\star\star)$; hence D is not fully abstract for ℓ. ∎

Note that since C is not definable in ℓ, we could not apply the Full Abstraction Theorem. By contrast, to show that D is not fully abstract for ℓ_{C}, it suffices to show that P is not definable. For this purpose, we prove a result analogous to 6.18.

Proposition 6.20 *For $M \in \mathbf{\Lambda}(\{\mathsf{C}\})$, exactly one of the following conditions holds:*

$$\begin{array}{rl} (i) & M\uparrow \\ (ii) & M \gg \lambda x.N \\ (iii) & M \gg \mathsf{C} \\ (iv) & M \gg \underbrace{\mathsf{C}(\mathsf{C}\ldots(\mathsf{C}}_{n}(xN_1 \ldots N_k)\ldots)\ldots)P_1 \ldots P_m \ \ (n, k, m \geq 0) \end{array}$$

PROOF. Similar to 6.18. ∎

Theorem 6.21 P *is not definable in* ℓ_{C}*; hence* D *is not fully abstract for* ℓ_{C}*.*

PROOF. We show that ℓ_{C} satisfies

$$x(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow \,\&\, x\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow \;\Rightarrow\; x\mathbf{\Omega\Omega}\Downarrow,$$

and hence, as in the proof of the Full Abstraction Theorem, P is not definable in ℓ_{C}. As in the proof of 6.19, without loss of generality we consider closed terms of the form $\lambda y_1.\lambda y_2.M$. Assume $(\lambda y_1.\lambda y_2.M)(\mathbf{K\Omega})\mathbf{\Omega}\Downarrow$ and $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega}(\mathbf{K\Omega})\Downarrow$. Applying 6.20, we see that case (*i*) is impossible; cases (*ii*) and (*iii*) imply that $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega\Omega}\Downarrow$; while in case (*iv*), if $x = y_1$, then $(\lambda y_1.\lambda y_2.M)\mathbf{\Omega}(\mathbf{K\Omega})\Uparrow$, *contra hypothesis*; and if $x = y_2$, $(\lambda y_1.\lambda y_2.M)(\mathbf{K\Omega})\mathbf{\Omega}\Uparrow$, also *contra hypothesis*. Thus case (*iv*) is impossible, and the proof is complete. ∎

For our final non-definability result, we shall consider a different style of extension of ℓ, to incorporate *ground data*. We shall consider the simplest possible such extension, where a single atom is added. This corresponds to the domain equation

$$D_\star = \mathbf{1} + [D_\star \to D_\star]$$

(where $+$ is separated sum), which is indeed an extension of our original domain, in the sense that D is a retract of $D_\star$. $D_\star$ is still a Scott domain (indeed, a coherent algebraic cpo), but it is no longer a lattice; we have introduced *inconsistency* via the sum.

This extension is reflected on the syntactic level by two constants, $\star$ and C. We define

$$\ell_\star = (\mathbf{\Lambda}^0(\{\star, \mathsf{C}\}), \_\Downarrow\_)$$

with $\_\Downarrow\_$ extending the definition for ℓ as follows:

$$\bullet\ \star\Downarrow\star \qquad \bullet\ \mathsf{C}\Downarrow\mathsf{C} \qquad \bullet\ \frac{M\Downarrow\lambda x.N}{\mathsf{C}M\Downarrow\mathsf{T}} \qquad \bullet\ \frac{M\Downarrow\mathsf{C}}{\mathsf{C}M\Downarrow\mathsf{T}} \qquad \bullet\ \frac{M\Downarrow\star}{\mathsf{C}M\Downarrow\mathsf{F}}$$

where $\mathsf{T} \equiv \lambda x.\lambda y.x$, $\mathsf{F} \equiv \lambda x.\lambda y.y$. We see that the C combinator introduced here is a natural generalisation (not strictly an extension) of the C defined previously in the pure case. Of course, C corresponds to *case selection*, which in the unary case — lifting being unary separated sum — is just convergence testing.

A theory can be developed for $\ell_\star$ which runs parallel to what we have done for the pure lazy λ-calculus. Some of the technical details are more complicated because of the presence of inconsistency, but the ideas and results are essentially the same. Our reasons for mentioning this extension are twofold:

1. To show how the ideas we have developed can be put in a broader context. In particular, with the extension to $\ell_\star$ the reader should be able to see, at least in outline, how our work can be applied to systems such as Martin-Löf's Type Theory under its Domain Interpretation [DNPS83], and (the analogues of) our results in this section can be used to settle most of the questions and conjectures raised in [DNPS83].

2. To prove an interesting result which clarifies a point about which there seems to be some confusion in the literature; namely, *what is parallel or*?

The *locus classicus* for parallel or in the setting of typed λ-calculus is [Plo77]. But what of untyped λ-calculus? In [Bar84, p. 375], we find the following definition:

$$FMN = \begin{cases} \mathbf{I} & \text{if M or N is solvable,} \\ \text{unsolvable} & \text{otherwise} \end{cases}$$

which (modulo the difference between the standard and lazy theories) corresponds to our parallel convergence combinator P. The point we wish to make is this: in the pure λ-calculus, where (in domain terms) there are no inconsistent data values (since everything is a function), i.e. we have a lattice, parallel convergence does indeed play the role of parallel or, as the Full Abstraction Theorem shows. However, when we introduce ground data, and hence inconsistency, a distinction reappears between parallel convergence and parallel or, and it is definitely *wrong* to conflate them. To substantiate this claim, we shall prove the following result: even if parallel convergence is added to $\ell_\star$, parallel or is still not definable. This result is also of interest from the point of view of the fine structure of definability; it shows that parallelism is not all or nothing even in the simple, deterministic setting of $\ell_\star$.

Definition 6.22 $\ell_{\star\mathsf{P}}$ is the extension of $\ell_\star$ with a constant P and the rules

$$\bullet\ \mathsf{P}{\Downarrow}\mathsf{P} \quad \bullet\ \mathsf{P}M{\Downarrow}\mathsf{P}M \quad \bullet\ \frac{M\Downarrow}{\mathsf{P}MN{\Downarrow}\mathbf{I}} \quad \bullet\ \frac{N\Downarrow}{\mathsf{P}MN{\Downarrow}\mathbf{I}}$$

Definition 6.23 Let ℓ' be an extension of $\ell_\star$. We say that *parallel or is definable in* ℓ' if for some term M

$$\begin{array}{ll} (i) & M(\mathbf{K\Omega})\mathbf{\Omega}, M\mathbf{\Omega}(\mathbf{K\Omega}) \ \text{converge to abstractions} \\ (ii) & M\star\star{\Downarrow}\star\,. \end{array}$$

Theorem 6.24 *Parallel or is not definable in* $\ell_{\star\mathsf{P}}$.

Proof. We proceed along similar lines to our previous non-definability results. Firstly, we extend our definition of $>$ as follows:

- $\mathsf{constructor}(M) \equiv M$ is an abstraction, P, C or $\star$
- $\mathsf{constructor}(M)\ \&\ M \not\equiv \star \ \Rightarrow\ \mathsf{C}M > \mathsf{T}$
- $\mathsf{C}\star > \mathsf{F}$
- $\dfrac{M > M'}{\mathsf{C}M > \mathsf{C}M'}$
- $\mathsf{constructor}(M)$ or $\mathsf{constructor}(N) \ \Rightarrow\ \mathsf{P}MN > \mathbf{I}$

$$\bullet \quad \frac{M > M' \quad N > N'}{\mathsf{P}MN > \mathsf{P}M'N'}$$

With these extensions, $>$ is still a partial function, and 6.16, 6.17 still hold. For each $M \in \mathbf{\Lambda}(\{\star, \mathsf{C}, \mathsf{P}\})$, one of the following two disjoint conditions must hold:

- $M\uparrow$
- $M \gg N \,\&\, N \not> .$

We now define $\mathcal{T}$ to be the set of all terms M in $\mathbf{\Lambda}(\{\star, \mathsf{C}, \mathsf{P}, \bot\})$, where $\bot$ is a new constant, such that:

- $FV(M) \subseteq \{y_1, y_2\}$
- M contains no $>$-redex.

Note that $\mathcal{T}$ is closed under sub-terms.

Lemma A

For all $M \in \mathcal{T}$:

$$M[\mathbf{K\Omega}/y_1, \mathbf{\Omega}/y_2]\!\downarrow\! a \;\&\; M[\mathbf{\Omega}/y_1, \mathbf{K\Omega}/y_2]\!\downarrow\! b \;\&\; M[\star/y_1, \star/y_2]\!\downarrow\! c$$
$$\Rightarrow\; a = b = c = \star \text{ or } \star \notin \{a, b, c\}.$$

Proof. By induction on M. Since terms in $\mathcal{T}$ contain no $>$-redexes, M must have one of the following forms:

$$
\begin{array}{rl}
(i) & xN_1 \ldots N_k \;\; (x \in \{y_1, y_2\}, k \geq 0) \\
(ii) & \star N_1 \ldots N_k \;\; (k \geq 0) \\
(iii) & \lambda x.N \\
(iv) & \mathsf{C} \;\; (v)\ \mathsf{P} \;\; (vi)\ \mathsf{P}N \\
(vii) & \mathsf{C}NN_1 \ldots N_k \;\; (k \geq 0) \\
(viii) & \mathsf{P}M_1M_2N_1 \ldots N_k \;\; (k \geq 0) \\
(ix) & \bot N_1 \ldots N_k \;\; (k \geq 0)
\end{array}
$$

Most of these cases can be disposed of directly; we deal with the two which use the induction hypothesis.

(vii). Firstly, we can apply the induction hypothesis to N to conclude that $N[c_1/y_1, c_2/y_2]$ converges to the same result (i.e. either an abstraction or $\star$) for all three argument combinations c_1, c_2; we can then apply the induction hypothesis to either $N_1N_3 \ldots N_k$ or $N_2N_3 \ldots N_k$.

$(viii)$. Under the hypothesis of the Lemma, we must have

$$(\mathsf{P}M_1M_2)[c_1/y_1, c_2/y_2]\Downarrow\mathbf{I}$$

for all three argument combinations c_1, c_2; hence we can apply the induction hypothesis to $N_1 \ldots N_k$. ∎

Lemma B

Let $M \in \mathbf{\Lambda}\ (\{\star, \mathsf{C}, \mathsf{P}\})$, with $FV(M) \subseteq \{y_1, y_2\}$. Then for some $M' \in \mathcal{T}$, for all $P, Q \in \mathbf{\Lambda}^0(\{\star, \mathsf{C}, \mathsf{P}\})$:

$$M[P/y_1, Q/y_2]{\downarrow}\star \iff M'[P/y_1, Q/y_2]{\downarrow}\star.$$

PROOF. Given M, we obtain M' as follows; working in an inside-out fashion, we replace each sub-term N by:

$$\begin{cases} N' & \text{if } N{\downarrow}N' \\ \perp & \text{if } N{\uparrow}. \end{cases} \blacksquare$$

Now suppose that we are given a putative term in $\mathbf{\Lambda}^0(\{\star, \mathsf{C}, \mathsf{P}\})$ defining parallel or. As in the proof of 6.21, we may take this term to have the form $\lambda y_1.\lambda y_2.M$. Applying Lemma B, we can obtain $M' \in \mathcal{T}$ from M; but then applying Lemma A, we see that $\lambda y_1.\lambda y_2.M'$ cannot define parallel or. Applying Lemma B again, we conclude that $\lambda y_1.\lambda y_2.M$ cannot define parallel or either. $\blacksquare$

7 Variations

Throughout this Chapter, we have focussed on the lazy λ-calculus. We round off our treatment by briefly considering the varieties of function space.

1. The Scott function space

$[D \to E]$, the standard function space of all continuous functions from D to E. In terms of our domain logic $\mathcal{L}$, we can obtain this construction by adding the axiom

$$(1) \quad t \le (t \to t).$$

Note that with (1), $\mathcal{L}$ collapses to a single equivalence class (corresponding to the trivial one-point solution of $D = [D \to D]$). For this reason, Coppo *et al.* have to introduce atoms in their work on Extended Applicative Type Structures [CDHL84].

2. The strict function space

$[D \to_\perp E]$, all *strict* continuous functions. This satisfies (1), and also

$$(2) \quad (t \to_\perp \phi) \le f \quad (\phi \ne t).$$

3. The lazy function space

$[D \to E]_\perp$, which satisfies neither (1) nor (2). This has of course been our object of study in this Chapter.

4. The Landin-Plotkin function space

$[D \to_\perp E]_\perp$, the lifted strict function space. This satisfies (2) but not (1). The reason for our nomenclature is that this construction in the category of domains and strict continuous functions corresponds to Plotkin's $[D \rightharpoonup E]$ construction in his (equivalent) category of predomains and partial functions [Plo85]. Moreover, this may be regarded as the formalisation of Landin's applicative-order λ-calculus, with abstraction used to protect expressions from evaluation, as illustrated extensively in [Lan64, Lan65, Bur75].

The intriguing point about these four constructions is that (1) and (2) are *mathematically* natural, yielding cartesian closure and monoidal closure in e.g. **CPO** and $\mathbf{CPO}_\perp$ respectively (the latter being analogous to partial functions over sets); while (3) and (4) are *computationally* natural, as argued extensively for (3) in this Chapter, and as demonstrated convincingly for (4) by Plotkin in his work on predomains [Plo85]. Much current work is aimed at providing good categorical descriptions of generalisations of (4) [Ros86, RR87, Mog86, Mog87, Mog]; a similar programme is being carried out for (3) by Chih-Hao Ong.

8 Further Directions

Our development of the lazy λ-calculus represents no more than a beginning. An extensive study is being undertaken by Chih-Hao Ong; anyone interested in pursuing the subject further is strongly recommended to read his forthcoming thesis (Imperial College, University of London; expected 1988). His results include: a syntactic characterisation of the local structure of lazy PSE models; a construction of a fully abstract model for ℓ_{C}; and a category-theoretic characterisation of the lazy λ-calculus.

Acknowledgements

I would like to thank Henk Barendregt, Peter Dybjer, Per Martin-Löf, Chih-Hao Ong, Gordon Plotkin and Jan Smith for inspiring discussions on these topics. Per Martin-Löf's ideas on the Domain Interpretation of his Type Theory have been particularly influential. The first version of this material was developed during and shortly after a most enjoyable visit to the Programming Methodology Group, Chalmers University Göteborg, in March 1984; and presented at the Seminar on Concurrency at CMU in July 1984. The present version crytallised during and between visits to the University of Nijmegen in March–April and August 1986, and was presented in seminars there in August 1986, and at the Institute of Declarative Programming at Austin in August 1987. I would like to thank my hosts on all these occasions for their hospitality, and particularly Bengt Nördstrom, Per Martin-Löf, Raymond Boute and David Turner. Finally, my thanks to the U.K. Science and Engineering Research Council and the Alvey Programme for financial support.

References

[Abr87a] S. Abramsky. A domain equation for bisimulation. 1987. Department of Computing, Imperial College.

[Abr87b] S. Abramsky. Domain theory in logical form. 1987. Submitted for publication.

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

[Aug84] L. Augustsson. A compiler for lazy ML. In *ACM Symposium on Lisp and Functional Programming, Austin*, pages 218–227, August 1984.

[Bar84] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[Bur75] W. H. Burge. *Recursive programming techniques*. Addison Wesley, Reading, Mass., 1975.

[BvL86] H. Barendregt and M. van Leeuwen. *Functional programming and the language TALE*. Technical Report 412, University of Utrecht Dept. of Mathematics, 1986.

[CDHL84] M. Coppo, M. Dezani-Ciancaglini, Furio Honsell, and G. Longo. Extended type structure and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquim '82*, pages 241–262, Elsevier Science Publishers B.V. (North-Holland), 1984.

[DNPS83] P. Dybjer, B. Nordström, K. Petersson, and J. Smith, editors. *Workshop on Semantics of Programming Languages*, Programming Methodology Group, Chalmers University of Technology, Göteborg, Sweden, August 1983.

[Fai85] J. Fairbairn. *Design and implementation of a simple typed language based on the lambda calculus*. PhD thesis, University of Cambridge, 1985.

[Hen80] P. Henderson. *Functional Programming: Applications and Implementation*. Prentice Hall, 1980.

[Koy82] Christiaan Peter Jozef Koymans. Models of the lambda calculus. *Information and Control*, 52:206–332, 1982.

[Lan64] P. J. Landin. The mechanical evaluation of expressions. *Computer Journal*, 6:308–320, 1964.

[Lan65] P. J. Landin. A correspondence between ALGOL 60 and Church's lambda notation. *Communications of the ACM*, 8:89–101,158–165, 1965.

[Lon83] Giuseppe Longo. Set-theoretical models of lambda calculus: theories, expansions and isomophisms. *Annals of Pure and Applied Logic*, 24:153–188, 1983.

[LS86] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic. Cambridge Studies in Advanced Mathematics Vol. 7*, Cambridge University Press, 1986.

[Mil77] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

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

[Mog] Eugenio Moggi. Partial cartesian closed categories of effective objects. To Appear in *Information and Computation.*

[Mog86] Eugenio Moggi. Categories of partial morphisms and the $lambda_p$-calculus. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Category Theory and Computer Programming*, pages 242–251, Springer-Verlag, 1986. *LNCS* Vol. 240.

[Mog87] Eugenio Moggi. *Partial Lambda Calculus.* PhD thesis, University of Edinburgh, 1987.

[Mor68] J. H. Morris. *Lambda Calculus Models of Programming Languages.* PhD thesis, Massachusets Institute of Technology, 1968.

[Mos74] Y. Moschovakis. *Elementary Induction on Abstract Structures.* North Holland, 1974.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Plo85] G. D. Plotkin. Lectures on predomains and partial functions. 1985. Notes for a course given at the Center for the Study of Language and Information, Stanford 1985.

[Ros86] Giuseppe Rosolini. *Continuity and Effectiveness in Topoi.* PhD thesis, Carnegie-Mellon University, 1986.

[RR87] E. Robinson and G. Rosolini. Categories of partial maps. 1987. Computing laboratory and DPMMS, Cambridge University.

[Sco80] D. S. Scott. Relating theories of lambda calculus. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*, pages 403–450, Academic Press, 1980.

[Tur85] D. A. Turner. Miranda—a non-strict functional language with polymorphic types. In J.P. Jouannaud, editor, *Functional programming languages and Computer Architectures*, Springer-Verlag, Berlin, 1985. Lecture Notes in Computer Science Vol. 201.

[Wad85] P. Wadler. *Introduction to ORWELL.* Technical Report, Oxford University Programming Research Group, 1985.