# Fully Abstract Models of the Lazy Lambda Calculus

C.-H. Luke Ong

Imperial College, London SW7 2BZ

## Abstract

Much of what is known about the model theory and proof theory of the $\lambda$-calculus is *sensible* in nature, *i.e.* only *head normal forms* are semantically meaningful. However, most functional languages are *lazy*, *i.e.* programs are evaluated in normal order to *weak head normal forms*. In this paper, we seek to develop a theory of *lazy* or *strongly sensible* $\lambda$-calculus that corresponds to practice. A pure lazy language $\lambda\ell$ is defined in which the only computational observable is convergence to abstraction. $\lambda\ell$ is not fully abstract w.r.t. $D$, the initial solution of the domain equation $D \cong [D \to D]_\perp$ — the canonical model; however, $\lambda\ell_P$ which is $\lambda\ell$ augmented with a *parallel convergence* construct P is. Two more languages $\lambda\ell_C$ ($\lambda\ell$ with *convergence testing*) and $\lambda\ell_\omega$ ($\lambda\ell$ with *projections*) with expressive powers between those of $\lambda\ell$ and $\lambda\ell_P$ are introduced and their full abstraction properties w.r.t. $D$ are studied. A general method for constructing fully abstract models for a class of lazy languages, including $\lambda\ell_C$ and $\lambda\ell_\omega$, is illustrated. A new formal system $\lambda\beta$C ($\lambda\beta$-calculus with convergence testing C) is introduced and its properties investigated.

## 1 Introduction

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

- Much of what is known about the model theory and proof theory of the $\lambda$-calculus is *sensible* in nature, *i.e.* all unsolvables [Bar84] are identified. Crucially, $\lambda x.\bot = \bot$ where $\bot$ represents any divergent term (or program).
- In practice, however, most implementations of functional languages are *lazy* [HM76], *i.e.* programs are reduced in *normal order* to *weak head normal forms* (whnf) [PJ87], corresponding to a *call-by-name* semantics. Consequently, $\lambda x.\bot \neq \bot$, because all abstractions, being in whnf, are deemed to be legitimate and meaningful programs.

In this paper, we seek to develop a theory of *lazy* functional programming that corresponds to practice.

### 1.1 Applicative Bisimulation

We turn the pure, untyped $\lambda$-calculus into a paradigmatic functional language by admitting closed $\lambda$-terms as *programs* and (closed) abstractions as *values*. The lazy evaluation mechanism is then captured by a binary reduction relation $\Downarrow \subseteq \Lambda^o \times \Lambda^o$ ($\Lambda^o$ is the collection of all closed $\lambda$-terms) defined inductively as:

$$\overline{\lambda x.P \Downarrow \lambda x.P}$$

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

We read $M \Downarrow N$ as "$M$ reduces lazily to principal whnf $N$". NOTATION

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

$$M\Uparrow \stackrel{\text{def}}{=} \neg[M\Downarrow] \quad \text{"}M\text{ diverges"}.$$

The reduction $\Downarrow$ is *deterministic.* $\langle\Lambda^o, \Downarrow\rangle$ is known as the *pure lazy language.*

Under the reduction strategy $\Downarrow$, the possible "results" are of a particularly simple, indeed *atomic* kind. A term $M$ either converges to an abstraction (and according to this strategy, we have no clue as to the structure "under" the abstraction); or it diverges. In contrast to simply typed $\lambda$-calculi with ground constants, say Plotkin's PCF language [Plo77], the computational "observables" in our case is *convergence to abstraction.* As it stands, the relation $\Downarrow$ is too "shallow" to furnish enough information about the behaviour of the system.

Inspired by the work of Milner and Park on concurrency, Abramsky [Abr88] introduced an operational preorder on $\lambda$-terms called *applicative bisimulation* providing a tool which enables much deeper comparisons between the operational contents of terms to be made by using $\Downarrow$ as the basic "building blocks".

We prescribe a recursive specification of the applicative bisimulation preorder $M \sqsubseteq^B N$:

$$M \sqsubseteq^B N \iff$$
$$M \Downarrow \lambda x.P \Rightarrow \{ N \Downarrow \lambda x.Q \quad \& \quad \forall R \in \Lambda^o.P[x := R] \sqsubseteq^B Q[x := R] \}.$$

$\sqsubseteq^B$ may be defined as the conjunction of a family of inductively defined preorders $\{ \sqsubseteq^B_k : k \in \omega \}$ on $\Lambda^o$:

- $\forall M, N.M \sqsubseteq^B_0 N$.

  $M \sqsubseteq^B_{k+1} N \stackrel{\text{def}}{=}$

- $M \Downarrow \lambda x.P \Rightarrow \{ N \Downarrow \lambda x.Q \quad \&$

  $\forall R \in \Lambda^o.P[x := R] \sqsubseteq^B_k Q[x := R] \}$.

- $M \sqsubseteq^B N \stackrel{\text{def}}{=} \forall k \in \omega.M \sqsubseteq^B_k N$.

The definition is then extended to all $\lambda$-terms by considering closures: for $M \in \Lambda$, $M \sqsubseteq^B N \stackrel{\text{def}}{=} \forall \sigma : \mathsf{Var} \to \Lambda^o.M_\sigma \sqsubseteq^B N_\sigma$. We abbreviate $M \sqsubseteq^B N \;\&\; N \sqsubseteq^B M$ as $M \sim^B N$.

It is easy to see that for all $M, N \in \Lambda^o$,

$$M \sqsubseteq^B N \iff \forall \vec{P} \subseteq \Lambda^o.M\vec{P}\Downarrow \iff N\vec{P}\Downarrow;$$

where $\vec{P}$ denotes a sequence of $\lambda$-terms.

We define an (in)equational theory $\lambda\ell \stackrel{\text{def}}{=} \langle \Lambda^o, \sqsubseteq, = \rangle$ which we call *Abramsky lazy $\lambda$-theory* where:

$$\lambda\ell \vdash M \sqsubseteq N \stackrel{\text{def}}{=} M \sqsubseteq^B N,$$

$$\lambda\ell \vdash M = N \stackrel{\text{def}}{=} M \sim^B N.$$

## 1.2 Properties of $\lambda\ell$

The applicative bisimulation relation can be described as a "Morris-style contextual (pre)congruence" [Mor68]. Define a binary relation $\sqsubseteq^{C[]}$ on $\Lambda^o$:

$$M \sqsubseteq^{C[]} N \stackrel{\text{def}}{=} \forall C[\,].C[M]\Downarrow \Rightarrow C[N]\Downarrow;$$

where $C[\,]$ ranges over closed contexts. $\sqsubseteq^{C[]}$ is extended to $\Lambda$ in the same way as $\sqsubseteq^B$.

Since the computational behaviour of a program in our framework can only be described by observing convergence, the preorder $\sqsubseteq^{C[]}$ is just the usual notion of *operational precongruence* (for which more anon). Computationally, operational precongruence enunciates the *safety* criterion for the replacement of one program fragment by another. That is to say, if $M \sqsubseteq^{C[]} N$, then we can safely replace any occurrence of $M$ (as a subterm) in any program by $N$. Abramsky in *op. cit.* used the powerful machinery of the *Stone duality* between *domains* and their *logics of observable properties* to prove that applicative bisimulation is characterized by *observability under all contexts*; more precisely,

PROPOSITION **1.2.1** $\sqsubseteq^B = \sqsubseteq^{C[]}$. □

As a corollary, the pure lazy language satisfies the property of *operational extensionality* [Blo88], *i.e.* if two terms agree on all sequences of definable arguments, then they are operationally congruent.

In [Ong88, Chap 2], we studied the class of *fully lazy $\lambda$-theories* which are $\lambda$-theories [Bar84] that distinguish between two unsolvable terms iff they have different *orders.*[1] $\lambda\ell$ may be characterized as the *maximal* fully lazy $\lambda$-theory; equivalently, $\lambda\ell$ is *Hilbert-Post complete* w.r.t. fully lazy $\lambda$-theories [Bar84, pp 83]. More precisely, we have,

PROPOSITION **1.2.2** *Let $M, N$ be two unsolvables of orders $m$ and $n$ respectively. Then, $\lambda\ell \vdash M = N \iff m = n$. Furthermore, for any $P, Q$ such that $\lambda\ell \nvdash P = Q$, either $\lambda\ell + (P = Q)$ is inconsistent or it is not fully lazy.* □

## 1.3 Applicative Structures

The Abramsky lazy $\lambda$-theory $\lambda\ell$ is derived from a particular operational model — the transition system $\langle \Lambda^o, \Downarrow \rangle$. What is the general mathematical structure of which the previous transition system is an instance? More generally, how should a model of the lazy $\lambda$-calculus (call it *lazy $\lambda$-model*) look like? In the lazy regime, a clear distinction is made between terms that evaluate to *values* (=abstractions) and those that do not (*i.e.* the *strongly unsolvables*[2]). A natural way to reflect this dichotomy is to decree that the underlying applicative structure comes equipped with *divergent* elements. Moreover, normal order reduction entails an application operation which is left-strict but *not* right-strict. These lead to the following definitions.

A *quasi-applicative structure with divergence* (q-aswd) is a structure $\langle A, \cdot, \Uparrow \rangle$ such that $\langle A, \cdot \rangle$ is an applicative structure with a (non-empty) *divergence* predicate $\Uparrow \subseteq A$ satisfying $\forall x \in A.x\Uparrow \Rightarrow \forall y \in A.x \cdot y\Uparrow$. Define $x\Downarrow \stackrel{\text{def}}{=} \neg[x\Uparrow]$. The language $\langle \Lambda^o, \Downarrow \rangle$ is a q-aswd with $\Uparrow$ consisting of all closed strongly unsolvable terms.

Given a q-aswd $\langle A, \cdot, \Uparrow \rangle$, we define a *bisimulation preorder* $\sqsubseteq^A$ (by mimicking $\sqsubseteq^B$ in $\langle \Lambda^o, \Downarrow \rangle$) satisfying the following recursive specification:

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

$\sqsubseteq^A$ is defined as the conjunction of a sequence of inductively defined preorders in the same way as $\sqsubseteq^B$.

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

$$\forall a, b, c \in A.b \sqsubseteq^A c \Rightarrow a \cdot b \sqsubseteq^A a \cdot c.$$

## 1.4 Lazy $\lambda$-Models

We are now in a position to formalize the notion of lazy $\lambda$-

[2] A $\lambda$-term $M$ is *strongly unsolvable* if $M$ has order 0 and $\neg[\lambda\beta \vdash M = x\vec{N}]$, *i.e.* unsolvables of order 0. See [Ong88, Chap 1 & 2] for motivation

[1] The *order* of a $\lambda$-term $M$ is the largest $i$ such that $\exists N \in \Lambda.\lambda\beta \vdash M = \lambda x_1 \cdots x_i.N$.

model. An (environment) *lazy $\lambda$-model* $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle$ is a structure such that:

- $\langle A, \cdot, \Uparrow \rangle$ is a q-aswd.
- $[\![-]\!]_-$ is *homomorphic* w.r.t. application, i.e. $\forall M, N \in \Lambda(\underline{A}), \forall a \in A$,

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

- $\mathcal{A} \models (\beta)$, i.e. $\lambda\beta \vdash M = N \Rightarrow \mathcal{A} \models M = N$.
- $\mathcal{A} \models (\xi)$ where $\xi$ is:

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

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

Just as the classical $\lambda$-models can be presented equivalently in three different ways, namely, environment, functional or first order (combinatory) $\lambda$-models emphasizing their respective features (see [Koy84,Mey82,Bar84]); so may the lazy $\lambda$-models (see [Ong88, Chap 3]). It is well-known that $\lambda$-models can be characterized as reflexive objects which have enough points in Cartesian closed categories [Sco80,Koy84]. A similar presentation of lazy $\lambda$-models (and those in which convergence testing is definable) may be carried out in *partial Cartesian closed dominical categories* [Ong88, Chap 5] (see [RR88] for partial categories). A general account of lazy $\lambda$-models and partial categories will be the subject of a forth-coming paper.

### 1.5 Lambda Transition Systems

In [Ong88, Chap 3], we studied in some details the local structure of a class of lazy $\lambda$-models called *free lazy PSE-models* [Lon83]. In this paper, we will study another lazy $\lambda$-model $D$, the initial solution of the domain equation $D \cong [D \to D]_\perp$ in the category of cpos and continuous functions. $D$ satisfies a rather strong extensionality axiom ($\mathrm{Ext}_{\mathrm{bisim}}$):

$$\forall x, y \in D.x \sim^B y \Rightarrow x = y.$$

Lazy $\lambda$-models which satisfy the above axiom are called *lambda transition systems* (lts). An lts $\mathcal{A}$ is *adequate* if $\forall M \in \Lambda^o.M\Downarrow \iff \mathcal{A} \models M\Downarrow$. A prime example of an (adequate) lts is in fact a "syntactic structure" — the quotient $\langle \Lambda^o/{\sim^B}, \Downarrow \rangle$ (which is well-defined by an appeal to Proposition 1.2.1) henceforth referred to as $\lambda\ell$ by abuse of notation.

With respect to the inherent preorder, i.e. the bisimulation preorder, any lts $\mathcal{A} = \langle A, \cdot, \Uparrow, [\![-]\!]_- \rangle$ has unique least and greatest elements. They are the interpretations of the strongly unsolvables and $\mathbf{PO}_\infty$-terms[3] respectively. If an lts $\mathcal{A}$ is *adequate*, then it is a *fully lazy $\lambda$-model*, i.e. for unsolvables $M, N$ of orders $m, n \in \omega + 1$ respectively,

$$\mathcal{A} \models M \sqsubseteq^A N \iff m \leqslant n.$$

## 2 Convergence Testing

Is there a closed $\lambda$-term $X$ that discriminates between the convergent and divergent $\lambda$-terms? *i.e.* $\forall M \in \Lambda^o$,

$$\begin{cases} XM = \mathbf{I} & \text{if } M\Downarrow, \\ XM\Uparrow & \text{if } M\Uparrow; \end{cases}$$

where $\mathbf{I}$ is the identity. A case analysis of the possible orders of $X$ shows that *no* such convergence discriminatory function is *internally* definable in $\langle \Lambda^o, \Downarrow \rangle$. More generally, we say that *convergence testing is definable* in a q-aswd $\mathcal{A} = \langle A, \cdot, \Uparrow \rangle$ if $\exists c \in A$ such that for $x \in A$, $\mathcal{A}$ satisfies the following:

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

### 2.1 The lts $\lambda\ell_{\mathsf{C}}$

Define an augmented language $\langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{c}} \rangle$ (not superfluous since convergence testing is not definable in $\langle \Lambda^o, \Downarrow \rangle$), where $\mathsf{C}$ is a formal constant called *convergence testing* and $\Downarrow_{\mathsf{c}}$ is a reduction relation defined on $\Lambda(\mathsf{C})^o$ by

$$\frac{}{\mathsf{C} \Downarrow_{\mathsf{c}} \mathsf{C}} \quad \frac{M\Downarrow_{\mathsf{c}}}{\mathsf{C}M \Downarrow_{\mathsf{c}} \mathbf{I}} \quad \frac{}{\lambda x.P \Downarrow_{\mathsf{c}} \lambda x.P}$$

$$\frac{M \Downarrow_{\mathsf{c}} \lambda x.P \qquad P[x := Q] \Downarrow_{\mathsf{c}} N}{MQ \Downarrow_{\mathsf{c}} N}.$$

$\langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{c}} \rangle$ is a q-aswd; denote the associated bisimulation preorder as $\sqsubseteq^{\mathsf{C}}$ and the induced equivalence as $\sim^{\mathsf{C}}$. Just as $\sqsubseteq^B$, $\sqsubseteq^{\mathsf{C}}$ can be characterized by *observability under all contexts* i.e. for $M, N \in \Lambda(\mathsf{C})^o$,

$$M \sqsubseteq^{\mathsf{C}} N \iff \forall C[\,].C[M]\Downarrow_{\mathsf{c}} \Rightarrow C[N]\Downarrow_{\mathsf{c}}.$$

An immediate corollary is that $\sqsubseteq^{\mathsf{C}}$ is a *pre-congruence i.e.*

$$M \sqsubseteq^{\mathsf{C}} N \Rightarrow \forall C[\,].C[M] \sqsubseteq^{\mathsf{C}} C[N].$$

In the same way as $\lambda\ell$, we define an (in)equational theory $\lambda\ell_{\mathsf{C}}$. The q-aswd $\langle \Lambda(\mathsf{C})^o, \Downarrow_{\mathsf{c}} \rangle$ is an lts which we refer to as $\lambda\ell_{\mathsf{C}}$ by abuse of notation.

### 2.2 Properties of $\lambda\beta\mathsf{C}$

Just as $\lambda\ell$ is a $\lambda$-theory, *i.e.* a consistent extension of the formal system $\lambda\beta$, so $\lambda\ell_{\mathsf{C}}$ is a consistent extension of the

[3] A $\mathbf{PO}_\infty$-term $M$ is one whose order is unbounded i.e. $M$ is (convertible to) an infinitely deeply-nested abstraction; for example $\mathbf{YK} = \lambda x_1 \cdots x_n \mathbf{YK}$ for any $n \in \omega$ or $(\lambda xy.xx)(\lambda xy.xx)$.

formal system $\lambda\beta\mathsf{C}$ defined on the language $\mathbf{\Lambda}(\mathsf{C})$ by extending the rules of $\lambda\beta$ by the axiom scheme $\mathsf{C}M = \mathbf{I}$ provided $M\Downarrow_{\mathsf{c}}$. By abuse of notation, we "overload" the symbol $\Downarrow_{\mathsf{c}}$ by using it to denote the (new) reduction relation on $\mathbf{\Lambda}(\mathsf{C})$ (*i.e.* possibly open $\lambda\mathsf{C}$-terms), defined by the same rules as the previous $\Downarrow_{\mathsf{c}}$. Provability in $\lambda\beta\mathsf{C}$ is denoted $\lambda\beta\mathsf{C} \vdash$. Define an associated proof system with formulae of the form $M \geqslant N$ as: $\lambda\beta\mathsf{C} \vdash M \geqslant N$ if $\lambda\beta\mathsf{C} \vdash M = N$ *without* using the symmetry rule. The *one-step $\beta\mathsf{C}$-reduction* is the compatible closure of the union of the relation schema: $\langle(\lambda x.P)Q, P[x := Q]\rangle$, $\langle\mathsf{CC}, \mathbf{I}\rangle$ and $\langle\mathsf{C}(\lambda x.P), \mathbf{I}\rangle$.

THEOREM **2.2.1 (Church-Rosser)** *The proof system $\lambda\beta\mathsf{C}$ is Church-Rosser, i.e. $\lambda\beta\mathsf{C} \vdash M \geqslant M_i$ for $i = 1,2 \Rightarrow \exists N.\lambda\beta\mathsf{C} \vdash M_i \geqslant N$.* □

$\lambda\beta\mathsf{C}$ satisfies a *standardization theorem.* First, a definition. Define *one-step lazy $\beta\mathsf{C}$ reduction* $\rightarrow_{\mathsf{l}}$ on $\mathbf{\Lambda}(\mathsf{C})$ by

$$\frac{}{\mathsf{CC} \rightarrow_{\mathsf{l}} \mathbf{I}} \qquad \frac{}{\mathsf{C}(\lambda x.P) \rightarrow_{\mathsf{l}} \mathbf{I}}$$

$$\frac{}{(\lambda x.P)Q \rightarrow_{\mathsf{l}} P[x := Q]}$$

$$\frac{M \rightarrow_{\mathsf{l}} M'}{\mathsf{C}M \rightarrow_{\mathsf{l}} \mathsf{C}M'} \qquad \frac{M \rightarrow_{\mathsf{l}} M'}{MN \rightarrow_{\mathsf{l}} M'N}.$$

Define *standard reduction sequence* on $\mathbf{\Lambda}(\mathsf{C})$ inductively:

$$\frac{}{\langle x\rangle} \qquad \frac{}{\langle\mathsf{C}\rangle} \qquad \frac{\langle N_2, \cdots, N_n\rangle \quad N_1 \rightarrow_{\mathsf{l}} N_2}{\langle N_1, N_2, \cdots, N_n\rangle}$$

$$\frac{\langle N_1, \cdots, N_n\rangle}{\langle \lambda x.N_1, \cdots, \lambda x.N_n\rangle}$$

$$\frac{\langle M_1, \cdots, M_m\rangle \quad \langle N_1, \cdots, N_n\rangle}{\langle M_1N_1, \cdots, M_mN_1, M_mN_2, \cdots, M_mN_n\rangle}.$$

THEOREM **2.2.2 (Standardization)** *Let $M, N \in \mathbf{\Lambda}(\mathsf{C})$. Then, $\lambda\beta\mathsf{C} \vdash M \geqslant N \iff \exists\vec{M}.M_1 \equiv M \,\&\, M_m \equiv N \,\&\, \langle M_1, \cdots, M_m\rangle$.* □

The proofs of the two previous Theorems employ *parallel reduction* technique á la Plotkin [Plo75], Martin-Löf and Tait.

### 2.3 Call-by-Value Simulation

The introduction of convergence testing in $\langle\mathbf{\Lambda}(\mathsf{C})^{o}, \Downarrow_{\mathsf{c}}\rangle$ enables an application operation which is both left and right strict to be simulated. This corresponds to call-by-value evaluation. Define a *call-by-value* language $\langle\mathbf{\Lambda}^{o}, \Downarrow_{\mathsf{v}}\rangle$ where $\Downarrow_{\mathsf{v}}$ is a reduction relation on $\mathbf{\Lambda}^{o}$ defined as follows:

$$\frac{}{\lambda x.M \Downarrow_{\mathsf{v}} \lambda x.M}$$

$$\frac{M \Downarrow_{\mathsf{v}} \lambda x.P \quad N \Downarrow_{\mathsf{v}} Q \quad P[x := Q] \Downarrow_{\mathsf{v}} L}{MN \Downarrow_{\mathsf{v}} L}$$

The associated convergence predicate $\Downarrow_{\mathsf{v}}$ and divergence predicate $\Uparrow_{\mathsf{v}}$ are defined in the usual way.

We define a translation $\overline{(\ )} : \mathbf{\Lambda} \rightarrow \mathbf{\Lambda}(\mathsf{C})$ by structural induction as follows:

$$\begin{array}{lcl} \overline{x} & \stackrel{\text{def}}{=} & x, \\ \overline{\lambda x.M} & \stackrel{\text{def}}{=} & \lambda x.\overline{M}, \\ \overline{MN} & \stackrel{\text{def}}{=} & \mathsf{C}\overline{N}((\overline{M})(\overline{N})). \end{array}$$

THEOREM **2.3.1 (Simulation)** *Let $M \in \mathbf{\Lambda}^{o}$. Then,*

$$M\Downarrow_{\mathsf{v}} \iff \overline{M}\Downarrow_{\mathsf{c}}.$$

PROOF See Appendix. □

## 3 Canonical Model $\mathcal{D}$

In the *sensible* theory (in which all unsolvables are identified [Bar84]), $\lambda$-calculus may be regarded as being characterized by the type equation $D = [D \rightarrow D]$ — every element of $D$ may be *unfolded* into a continuous function from $D$ to $D$ — which has *no* non-trivial initial solution in, say, the category of cpos and continuous functions. In the lazy regime, the equation needs to be modified to $D = [D \rightarrow D]_{\perp}$, where $(-)_{\perp}$ is the standard *lifting* operation [Plo81], to reflect the sharp distinction between convergent and divergent elements: only convergent elements *unfold* to functions from $D$ to $D$, the divergent element $\perp$ in $D$, devoid of any *functional* (or *operator* as opposed to *operand*) content, "unfolds" naturally to the adjoint $\perp$.

We regard the initial solution to the equation $D \cong [D \rightarrow D]_{\perp}$ in the category of cpos and continuous functions (which is non-trivial) as the canonical model of the pure lazy language (see [Abr88] for a domain logic justification). The construction of the initial solution is standard. We refer the reader to [Plo81] and [SP82] for a detailed account. As usual, we regard each canonical approximant $D_n$ for $n \in \omega$ as a subset of $D$. The isomorphism pair is denoted as:

$$D \xrightarrow{\mathsf{Fun}} [D \rightarrow D]_{\perp} \xrightarrow{\mathsf{Gr}} D.$$

Recall the category-theoretic characterization of *lifting* as the left adjoint to the forgetful functor $U$:

$$\mathsf{CPO} \xrightarrow{(-)_{\perp}} \mathsf{CPO}_{\perp} \xrightarrow{U} \mathsf{CPO}$$

where $\mathsf{CPO}_{\perp}$ is the sub-category of strict functions with:

- A natural transformation:

$$\mathsf{up} : I_{\mathsf{CPO}} \rightarrow U \circ (-)_{\perp}.$$

- For each continuous function $f : D \rightarrow UE$, its adjoint $\mathsf{lift}(f) : (D)_{\perp} \rightarrow_{\perp} E$.

Concretely, we have, for $x, y \in D$:

$$(D)_{\perp} \qquad \stackrel{\text{def}}{=} \quad \{\perp\} \cup \{\langle 0, d\rangle \mid d \in D\},$$

$$
\begin{aligned}
x \sqsubseteq y \quad &\stackrel{\text{def}}{=} \quad x = \bot \quad \text{or}\, [x = \langle 0, d\rangle \,\& \\
&\qquad y = \langle 0, d'\rangle \,\&\, d \sqsubseteq_D d'], \\
\mathsf{up}_D(d) \quad &\stackrel{\text{def}}{=} \quad \langle 0, d\rangle, \\
\mathsf{lift}(f)(\bot) \quad &\stackrel{\text{def}}{=} \quad \bot_E, \\
\mathsf{lift}(f)\langle 0, d\rangle \quad &\stackrel{\text{def}}{=} \quad f(d).
\end{aligned}
$$

$D$ is a lts with $\Uparrow = \{\bot\}$ and application is defined as:

$$
d \cdot e \stackrel{\text{def}}{=} \begin{cases} f(d) & \text{if } \mathsf{Fun}(d) = \langle 0, f\rangle; \\ \bot & \text{if } \mathsf{Fun}(d) = \bot. \end{cases}
$$

Abstractions have denotation

$$
[\![\lambda x.M]\!]_\rho \stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}(\lambda d.[\![M]\!]_{\rho[x:=d]})).
$$

The interpretation of the rest of the $\lambda$-terms is standard.

## 3.1 Properties of $D$

$D$ does not satisfy the *strong extensionality* principle (see [Bar84]) — just consider $\bot$ and $\bot_1 \stackrel{\text{def}}{=} \mathsf{Gr}(\mathsf{up}(\lambda x \in D.\bot))$ which is the least convergent element; but it satisfies a weaker property which we call *conditional strong extensionality*: for $d, e \in D$,

$$
d\Downarrow \,\&\, e\Downarrow \Rightarrow [\forall x \in D. d \cdot x \sqsubseteq e \cdot x \Rightarrow d \sqsubseteq e].
$$

As a corollary, $D$ is *internally fully abstract*:

$$
\forall d, e \in D. d \sqsubseteq e \iff d \underset{\sim}{\sqsubset}^B e.
$$

Convergence testing is definable in $D$ as $\mathsf{Gr}(\mathsf{up}(\mathsf{f}_{\bot_1, i}))$, call it $c$, with $i \equiv [\![\lambda x.x]\!]$ and $\mathsf{f}_{d,e}$ being the standard *step function* defined as

$$
\mathsf{f}_{d,e}(x) \stackrel{\text{def}}{=} \begin{cases} e & \text{if } d \sqsubseteq x, \\ \bot & \text{else.} \end{cases}
$$

$\langle \psi_n \rangle_{n \in \omega}$, the canonical projection functions from $D$ to $D_n$, are not $\lambda$-definable but they are $\lambda\mathsf{C}$-*definable*. That is to say, for each $n \in \omega$, $\exists \mathbf{\Psi}_n \in \mathbf{\Lambda}(\mathsf{C})^o. \forall d \in D. \psi_n(d) \equiv d_n = [\![\mathbf{\Psi}_n]\!] \cdot d$ where $\mathsf{C}$ is interpreted in $D$ as the convergence testing $c$. The $\lambda\mathsf{C}$-terms $\langle \mathbf{\Psi}_n \rangle_{n \in \omega}$ are defined inductively as:

$$
\begin{aligned}
\mathbf{\Psi}_0 \quad &\stackrel{\text{def}}{=} \quad \lambda x.\bot, \\
\mathbf{\Psi}_{n+1} \quad &\stackrel{\text{def}}{=} \quad \lambda x.\mathsf{C}x(\lambda y.\mathbf{\Psi}_n(x(\mathbf{\Psi}_n y))).
\end{aligned}
$$

$D$ is a $\omega$-algebraic complete lattice with an application operation that *left-preserves arbitrary joins*, *i.e.*

$$
\forall X \subseteq D. \forall d \in D. (\bigsqcup X) \cdot d = \bigsqcup_{x \in X} x \cdot d.
$$

This is a consequence of the coincidence of *representable* [Bar84] and the continuous functions of $D$.

# 4 The Full Abstraction Problem

The *full abstraction* problem was first studied by Gordon Plotkin in the seminal paper [Plo77], and shortly after by Robin Milner [Mil77]; see also [Sto88]. Informally stated, it is concerned with the problem of finding a denotational semantic definition for a programming language which is not "over-generous" w.r.t. a natural notion of operational equivalence defined by *observational indistinguishability*. Let $M, N$ be two program fragments (or terms) of a language $\mathcal{L}$. We define the notion of *operational precongruence* as follows. We say that $M$ *safely approximates* $N$ (in all contexts), denoted $M \underset{\sim}{\sqsubset}^{C[\,]} N$, if under all program contexts $C[\,]$, all that can be observed about the computational outcome of $C[M]$ can also be observed about $C[N]$. (In an optimizing compiler, for example, to preserve correctness, we will only want to replace $M$ by $N$ if $M \underset{\sim}{\sqsubset}^{C[\,]} N$.) A denotational semantics (*i.e.* the semantic function $[\![-]\!]_-$ with an associated domain $D$) is *fully abstract* w.r.t. the language $\mathcal{L}$ if for all terms $M, N$,

$$
M \underset{\sim}{\sqsubset}^{C[\,]} N \iff [\![M]\!] \sqsubseteq [\![N]\!].
$$

In the pure lazy language $\lambda\ell$, only convergence is observable. By an appeal to Proposition 1.2.1, the full abstraction criterion may be recast as:

$$
M \underset{\sim}{\sqsubset}^B N \iff [\![M]\!] \sqsubseteq [\![N]\!];
$$

similarly for $\lambda\ell_{\mathsf{c}}$.

## 4.1 Non Full Abstraction

THEOREM **4.1.1** $\exists M, N \in \mathbf{\Lambda}. M \sim^B N \,\&\, D \not\models M = N.$

PROOF Define $M \equiv x(\lambda y.x\mathbf{\Xi}\mathbf{\Omega}y)\mathbf{\Xi}, N \equiv x(x\mathbf{\Xi}\mathbf{\Omega})\mathbf{\Xi}$ where $\mathbf{\Omega}$ is any strongly unsolvable term and $\mathbf{\Xi}$ a $\mathbf{PO}_\infty$-term. $M \sim^B N$ may be shown by a case analysis of the possible orders of the interpretation of $x$ in $\mathbf{\Lambda}^o$. For $\rho$ which maps $x$ to $c$, $[\![M]\!]_\rho \neq [\![N]\!]_\rho$. □

As an immediate corollary, $\lambda\ell$ is *not* fully abstract w.r.t. $D$. This strengthens Abramsky's result in [Abr87, Theorem 6.6.19].

Given that convergence testing is definable in $D$ and that in the construction of the previous counter-example, convergence testing features so pivotally; it is at least plausible that $\lambda\ell_{\mathsf{c}}$ might be fully abstract with respect to $D$. This turns out *not* to be the case. This result was first obtained by Abramsky in [Abr87, Chap 6], and later independently by the author; and is a corollary of the following:

THEOREM **4.1.2** $\exists M, N \in \mathbf{\Lambda}(\mathsf{C}). M \sim^{\mathsf{c}} N \,\&\, D \not\models M = N.$

PROOF The proof depends on the *non-definability of parallel convergence* in $\langle \mathbf{\Lambda}(\mathsf{C})^o, \Downarrow_{\mathsf{c}} \rangle$. Generally, we say that *parallel convergence is definable* in a q-awsd $\mathcal{A} = \langle A, \cdot, \Uparrow \rangle$

if $\exists p \in A$ and for $x, y \in A$,

- $p\Downarrow$, $px\Downarrow$,
- $x\Downarrow \Rightarrow pxy\Downarrow$ & $pyx\Downarrow$,
- $x\Uparrow$ & $y\Uparrow \Rightarrow pxy\Uparrow$.

$\mathbf{\Omega}$ is as before and $\mathbf{\Omega}_1 \equiv \lambda x.\mathbf{\Omega}$. Let $M \equiv C[(x\mathbf{\Omega}\mathbf{\Omega})]$, $N \equiv C[(x\mathbf{\Omega}\mathbf{\Omega}_1)]$ where $C[\,] \stackrel{\text{def}}{=} \mathsf{C}(\mathsf{C}(x\mathbf{\Omega}_1\mathbf{\Omega})[\,])$. Then $M \sim^{\mathsf{c}} N$ & $D \not\models M = N$. □

### 4.2 Full Abstraction and the lts $\lambda\ell_{\mathsf{P}}$

Full abstraction is attained if all the compact elements of the (algebraic) semantic domains are *definable* in the language. Given a denotational semantics which is not fully abstract, then, there are generally two natural directions in which to achieve full abstraction:

- The *expansive* approach consists in enriching the language, as in the introduction of parallel or to PCF [Plo77], thereby enabling all *finite* semantic information to be represented syntactically as program phrases.
- The *restrictive* approach is to "cut down" (as in "quotienting out" by an appropriate equivalence relation) the existing "over-generous" semantic domain to an appropriate sub-structure that "fits" the prescribed language [Mil77,Mul86].

Abramsky showed that full abstraction is attained if *parallel convergence* $\mathsf{P}$ (which is not definable in $\lambda\ell$) is introduced to $\lambda\ell$ — an *expansive* approach. Define $\langle \mathbf{\Lambda}(\mathsf{P})^o, \Downarrow_{\mathsf{P}}\rangle$ by augmenting the rules for $\Downarrow$ with

$$\frac{}{\mathsf{P} \Downarrow_{\mathsf{P}} \mathsf{P}} \qquad \frac{}{\mathsf{P}M \Downarrow_{\mathsf{P}} \mathsf{P}M}$$

$$\frac{M\Downarrow_{\mathsf{P}}}{\mathsf{P}MN \Downarrow_{\mathsf{P}} \mathbf{I}} \qquad \frac{M\Downarrow_{\mathsf{P}}}{\mathsf{P}NM \Downarrow_{\mathsf{P}} \mathbf{I}}.$$

Call the augmented language $\lambda\ell_{\mathsf{P}}$. Let the associated bisimulation preorder be $\sqsubseteq^{\mathsf{P}}$.

THEOREM **4.2.1 (Abramsky)** *Let* $M, N \in \mathbf{\Lambda}(\mathsf{P})^o$. *Then,* $M \sqsubseteq^{\mathsf{P}} N \iff D \models M \sqsubseteq N$. □

# 5 Fully Abstract Models

### 5.1 The Problem

Let $\mathcal{K} = \langle K, \cdot^{\mathcal{K}}, \Downarrow_{\mathcal{K}}\rangle$ be a *fully-adequate* lts *i.e.* $\forall M \in \mathbf{\Lambda}(K)^o.M\Downarrow_{\mathcal{K}} \Rightarrow D \models M\Downarrow$ given an interpretation of $K$ in $D$. We aim to construct $Q^{\mathcal{K}}$, a *retract* of $D$, which is *fully abstract* with respect to $\mathcal{K}$ by the *restrictive* approach. That is to say $Q^{\mathcal{K}} \stackrel{\phi^{\mathcal{K}}}{\to} D \stackrel{\psi^{\mathcal{K}}}{\to} Q^{\mathcal{K}}$ with $\psi^{\mathcal{K}} \circ \phi^{\mathcal{K}} = \mathrm{id}$ and

$$\forall M \in \mathbf{\Lambda}(K)^o.[\![M]\!]^{\mathcal{K}} \stackrel{\text{def}}{=} \psi^{\mathcal{K}}([\![M]\!]);$$

satisfying $\forall M, N \in \mathbf{\Lambda}(K)^o$,

- $[\![MN]\!]^{\mathcal{K}} = [\![M]\!]^{\mathcal{K}} \cdot^{\mathcal{K}} [\![N]\!]^{\mathcal{K}}$;
- $M \sqsubseteq^{\mathcal{K}} N \iff Q^{\mathcal{K}} \models M \sqsubseteq N$.

In the following, we present a sketch of the general strategy we shall adopt to construct such $Q^{\mathcal{K}}$ for any fully-adequate lts $\mathcal{K}$. However, we are only able to *prove* that $Q^{\mathcal{K}}$ is fully abstract for $\mathcal{K}$ for a restricted class of lts's, which includes $\lambda\ell_{\mathsf{C}}$.

### 5.2 Construction of $Q^{\mathcal{K}}$

The construction relies on a *bisimulation logical relation*, $\lessdot^{\mathcal{K}}$, between $D$ and $\mathcal{K}$ which captures the extent to which an element $d$ of $D$ *bisimulates* an element $M$ of $\mathbf{\Lambda}(K)^o$ with respect to a suite of *tests* consisting of elements of $\mathbf{\Lambda}(K)^o$. $\lessdot^{\mathcal{K}} \subseteq D \times \mathbf{\Lambda}(K)^o$ satisfies the following recursive specification: $d \lessdot^{\mathcal{K}} M$ iff

- $\forall \vec{P} \subseteq \mathbf{\Lambda}(K)^o.D \models d\vec{P}\Downarrow \Rightarrow \mathcal{K} \models M\vec{P}\Downarrow_{\mathcal{K}}$ &
- $\forall e, N.[e \lessdot^{\mathcal{K}} N \Rightarrow de \lessdot^{\mathcal{K}} MN]$.

Thus, $\lessdot^{\mathcal{K}}$ may be seen as a natural extention of the by now familiar notion of bisimulation to one between two *different* lts's. That $\lessdot^{\mathcal{K}}$ is a *logical relation* [Plo73,Sta85] is an extrapolation of the notion of a precongruence. Intuitively, $d \lessdot^{\mathcal{K}} M$ if "all that can be observed about $d$ by applying it to terms in $\mathbf{\Lambda}(K)^o$ can equally be observed about $M$". $\lessdot^{\mathcal{K}}$ satisfies the property of *arbitrary join inclusiveness*, *i.e.* for any $X \subseteq D$

$$[\forall x \in X.x \lessdot^{\mathcal{K}} M] \Rightarrow (\bigsqcup X) \lessdot^{\mathcal{K}} M.$$

Define a preorder $\lesssim^{\mathcal{K}}$ on $D$ as $d \lesssim^{\mathcal{K}} e \stackrel{\text{def}}{=}$

$$\forall M \in \mathbf{\Lambda}(K)^o.e \lessdot^{\mathcal{K}} M \Rightarrow d \lessdot^{\mathcal{K}} M.$$

$\lesssim^{\mathcal{K}}$ compares the extent to which any two elements in $D$ *bisimulate* elements of $\mathbf{\Lambda}(K)^o$. Finally, $Q^{\mathcal{K}}$ is obtained by taking the respective supremums of the equivalence classes induced by the preoder $\lesssim^{\mathcal{K}}$.

### 5.3 Proof of Full Abstraction

This we secure by a technique first employed in [Mil77], see also [Mul86]. Construct for the model $Q^{\mathcal{K}}$ and the language $\mathcal{K}$ respectively a chain of *approximants* such that, roughly speaking, both the model $Q^{\mathcal{K}}$ and the language $\mathcal{K}$ are appropriate *completions* of their respective chains.

We assume that the canonical projections $\langle\psi_n\rangle_{n\in\omega}$ are definable in the language $\mathcal{K}$ by $\langle\mathbf{\Psi}_n\rangle_{n\in\omega}$ with

$$\mathcal{K} \models \forall n \in \omega.(\mathbf{\Psi}_{n+1}M)N = \mathbf{\Psi}_n(M(\mathbf{\Psi}_n N))$$

and that $\mathcal{K}$ is *reflexive*, *i.e.*

$$\forall M \in \mathbf{\Lambda}(K)^o.[\![M]\!] \lessdot^{\mathcal{K}} M;$$

conditions which $\lambda\ell_{\mathsf{C}}$ satisfies. For each $i \in \omega$, define

$\mathbf{\Lambda}(K)_i^o$ as the smallest subset of $\mathbf{\Lambda}(K)^o$ containing $\{\,\mathbf{\Psi}_i M : M \in \mathbf{\Lambda}(K)^o\}$ closed under application and $\sim^K$. $K_i = \langle \mathbf{\Lambda}(K)_i^o, \Downarrow_K\rangle$, the *i-th approximant of the language* $K$, is a well-defined q-aswd and denote the associated bisimulation ordering as $\sqsubseteq_i^K$.

Define for each $i \in \omega$, $Q_i^K$, *i-th approximant of the model,* consisting of the respective supremums of the intersection of $D_i$ and the equivalence classes induced by $\lesssim^K$.

Full abstraction of the *completion, i.e.* $\forall M, N \in \mathbf{\Lambda}(K)^o$,

$$M \sqsubseteq^K N \iff Q^K \models M \sqsubseteq N;$$

then follows from the full abstraction of the approximants, *i.e.* $\forall M, N \in \mathbf{\Lambda}(K)_i^o$

$$M \sqsubseteq_i^K N \iff Q_i^K \models M \sqsubseteq N$$

by a continuity argument. To summarize, the problem posed in §5.1 is solved for a class of lts as follows:

THEOREM **5.3.1 (Full Abstraction)** *Let $K$ be a fully-adequate, reflexive lts in which the projection functions $\langle\psi_n\rangle_{n\in\omega}$ are internally definable in the above sense. Then,* $\forall M, N \in \mathbf{\Lambda}(K)^o$

$$M \sqsubseteq^K N \iff Q^K \models M \sqsubseteq N.$$

□

$\lambda\ell_C$ satisfies the premises of the Theorem, hence a fully abstract model which is a retract of $D$ exists (and can be constructed) for it.

## 5.4 Complementarity of C

The convergence testing constant C introduced to $\lambda\ell$ enables the projection functions $\langle\psi_n\rangle_{n\in\omega}$ to be *internally definable* thereby making it possible to enunciate finite information of the domain within the language $\lambda\ell_C$. The domain-theoretic role C plays is clear: the *lifted* space $D_\perp$ is just unitary *separated sum* and C constitutes the corresponding *discriminatory function* [Plo81] i.e. the "elimination" operation concomitant to the "introduction" operation $\mathsf{up}_D$. That convergence testing *complements* lazy $\lambda$-calculus is reinforced further from a category-theoretic perspective. In [Ong88, Chap 5], we introduce a formal proof system $\lambda_L$ based on Scott's logic of existence [Sco79] which is *correct* (see [Plo75] for definition) w.r.t. $\lambda\ell$ and may be given a sound interpretation in partial categories. The interpretation is *complete* only for the subclass of $\lambda_L$ in which convergence testing is definable. These results lead us to conclude that a foundational treatment of lazy functional programming in the framework of the pure untyped $\lambda$-calculus should include as fundamental a device for testing convergence. We propose $\lambda\ell_C$ as such a framework.

## 5.5 The lts $\lambda\ell_\omega$ and Conjecture

We are not able to apply the above Theorem to $\lambda\ell$ because the projection functions $\langle\psi_n\rangle_{n\in\omega}$ are not internally definable, even though the construction of the retract $Q^{\lambda\ell}$ is well-defined. Can this be circumvented?

We define a new q-aswd $\lambda\ell_\omega = \langle(\mathbf{\Lambda}^\omega)^o, \Downarrow_\omega\rangle$ with $\mathbf{\Lambda}^\omega \stackrel{\text{def}}{=} \mathbf{\Lambda}(\langle\mathbf{\Psi}_n : n \in \omega\rangle)$ which is essentially $\lambda\ell$ augmented with the *formal* projection *constants*. The binary reduction relation $\Downarrow_\omega \subseteq (\mathbf{\Lambda}^\omega)^o \times (\mathbf{\Lambda}^\omega)^o$ is defined inductively as follows:

$$\frac{}{\mathbf{\Psi}_n \Downarrow_\omega \mathbf{\Psi}_n} \qquad \frac{}{\lambda x.P \Downarrow_\omega \lambda x.P}$$

$$\frac{M \Downarrow_\omega \mathbf{\Psi}_{n+1} \quad N\Downarrow_\omega}{MN \Downarrow_\omega \lambda y.\mathbf{\Psi}_n(N(\mathbf{\Psi}_n y))}$$

$$\frac{M \Downarrow_\omega \lambda x.P \quad P[x := Q] \Downarrow_\omega N}{MQ \Downarrow_\omega N}.$$

Define the bisimulation preorder $\sqsubseteq^\omega$ and the induced equivalence $\sim^\omega$ accordingly.

$\langle(\mathbf{\Lambda}^\omega)^o, \Downarrow_\omega\rangle$ is a fully-adequate reflexive lts with bisimulation ordering $\sqsubseteq^\omega$ in which the projection functions $\langle\psi_n\rangle_{n\in\omega}$ are trivially internally definable, hence the Full Abstraction Theorem applies.

The solubility of the full abstraction problem posed earlier for $\lambda\ell$ is then reduced to the validity of the following Conjecture:

CONJECTURE **5.5.1** *Let $M, N \in \mathbf{\Lambda}^o$. Then,*

$$M \sqsubseteq^B N \Rightarrow M \sqsubseteq^\omega N.$$

□

## 5.6 Summary

We summarize the full abstraction results obtained as follows:

| **Full Abstraction Results** | | |
|---|---|---|
| *Languages* | *Fully Abstract Models* | *Fully Abstract w.r.t. D* |
| $\lambda\ell_P$ | $D$ | Yes (4.2.1) |
| $\lambda\ell_C$ | $Q^{\lambda\ell_C}$ (5.3.1) | No (4.1.2) |
| $\lambda\ell_\omega$ | $Q^{\lambda\ell_\omega}$ (5.3.1) | No |
| $\lambda\ell$ | ? (5.5.1) | No (4.1.1) |

## Appendix

The proof of the Theorem consists of the following steps. First, we define a *one-step call-by-value reduction* $\rightarrow_{\mathsf{v}} \subseteq \Lambda^o \times \Lambda^o$

$$\frac{}{(\lambda x.P)(\lambda y.Q) \rightarrow_{\mathsf{v}} P[x := (\lambda y.Q)]}$$
$$\frac{M \rightarrow_{\mathsf{v}} M'}{MN \rightarrow_{\mathsf{v}} M'N}$$
$$\frac{M \rightarrow_{\mathsf{v}} M'}{(\lambda x.P)M \rightarrow_{\mathsf{v}} (\lambda x.P)M'}.$$

LEMMA **5.6.1** *$M \Downarrow_{\mathsf{v}} N$ iff the* deterministic *sequence of one-step call-by-value reductions starting from $M$ terminates at $N$.* □

Next, a (deterministic) one-step *parallel* $\beta$C-reduction $\rightarrow_{\mathsf{o}}$ is defined on $\Lambda(\mathsf{C})^o$ which *simulates* the one-step call-by-value reduction in a *step-wise fashion i.e.*

$$M \rightarrow_{\mathsf{v}} N \iff \overline{M} \rightarrow_{\mathsf{o}} \overline{N};$$

thus transforming the termination problem of $M$ under $\rightarrow_{\mathsf{v}}$ to one of $\overline{M}$ under $\rightarrow_{\mathsf{o}}$.

$\rightarrow_{\mathsf{o}}$ is defined by

$$\frac{}{\mathsf{C}(\lambda y.Q)((\lambda x.P)(\lambda y.Q)) \rightarrow_{\mathsf{o}} P[x := \lambda y.Q]}$$
$$\frac{P \rightarrow_{\mathsf{o}} P'}{\mathsf{C}P((\lambda x.Q)P) \rightarrow_{\mathsf{o}} \mathsf{C}P'((\lambda x.Q)P')}$$
$$\frac{P \rightarrow_{\mathsf{o}} P'}{\mathsf{C}Q(PQ) \rightarrow_{\mathsf{o}} \mathsf{C}Q(P'Q)}.$$

Now, for $\overline{M} \in \Lambda(\mathsf{C})^o$, $\overline{M}\Uparrow_{\mathsf{C}}$ iff $\overline{M}$ has an *infinite quasi lazy reduction i.e.* an infinite sequence of one-step $\beta$C-reductions containing an *infinite subsequence* of one-step *lazy* $\beta$C-reductions. Hence, the following completes the argument:

PROPOSITION **5.6.2** *Let $M \in \Lambda^o$. If there is an infinite sequence of $\rightarrow_{\mathsf{o}}$ reduction starting from $\overline{M}$, then $\overline{M}$ has an infinite quasi-lazy reduction.* □

The Proposition is proved by a tedious case analysis according to the last rule used in proving each of the one-step $\rightarrow_{\mathsf{o}}$ reduction in the given infinite sequence.

## References


[Abr87] Samson Abramsky.
*Domain Theory and the Logic of Observable Properties.*
PhD thesis, University of London, 1987.

[Abr88] Samson Abramsky.
The Lazy Lambda Calculus.
In David Turner, editor, *Declarative Programming*, Addison-Wesley, 1988.
To Appear.

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

[Blo88] Bard Bloom.
Can LCF be Topped? Flat Lattice Models of Typed Lambda Calculus.
In *Proceedings of the third Symposium on LICS*, Computer Society Press, 1988.

[HM76] P. Henderson and J. H. Morris.
A Lazy Evaluator.
In *Third ACM Symposium on The Principles of Programming Languages, Atlanta, GA*, 1976.

[Koy84] C. P. J. Koymans.
*Models of the Lambda Calculus.*
PhD thesis, University of Utrecht, 1984.

[Lon83] Giuseppe Longo.
Set-Theoretical Models of Lambda Calculus: Theories, Expansions and Isomorphisms.
*Annals of Pure and Applied Logic*, 24:153–188, 1983.

[Mey82] Albert Meyer.
What is a Model of the Lambda Calculus?
*Information and Control*, 52:87–122, 1982.

[Mil77] Robin Milner.
Fully Abstract Models of Typed Lambda-Calculus.
*Theoretical Computer Science*, 4:1–22, 1977.

[Mor68] J.-H. Morris.
*Lambda Calculus Models of Programming Languages.*
PhD thesis, M.I.T., 1968.

[Mul86] Ketan Mulmuley.
Fully Abstract Submodels of Typed Lambda Calculus.
*Journal of Computer and System Sciences*, 33:2–46, 1986.

[Ong88] C.-H. Luke Ong.
*The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming.*
PhD thesis, University of London, 1988.
To appear.

[PJ87] Simon L. Peyton Jones.
*The Implementation of Functional Programming Languages.*
Prentice-Hall, 1987.

[Plo73] G. D. Plotkin.
*Lambda-Definability and Logical Relations.*
Technical Report SAI-RM-4, School of A.I., Univ. of Edinburgh, 1973.

[Plo75] Gordon D. Plotkin.
Call-by-Name, Call-by-Value and the Lambda Calculus.
*Theoretical Computer Science*, 1:125–159, 1975.

[Plo77] Gordon D. Plotkin.
LCF as a Programming Language.
*Theoretical Computer Science*, 5:223–255, 1977.

[Plo81] Gordon D. Plotkin.
Post-Graduate Lecture Notes in Advanced Domain Theory.
1981.
Dept. of Computer Science, Univ. of Edinburgh.

[RR88] E. Robinson and G. Rosolini.
Categories of Partial Maps.
*J. Symbolic Logic*, 1988.
To appear.

[Sco79] Dana S. Scott.
Identity and Existence in Intuitionistic Logic.
In M. P. Fourman, C. J. Mulvey, and D. S. Scott, editors, *Applications of Sheaves*, pages 660–696, Springer-Verlag, 1979.

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

[SP82] M. B. Smyth and G. D. Plotkin.
The Category-Theoretic Solution of Recursive Domain Equations.
*SIAM J. Computing*, 11:761–783, 1982.

[Sta85] R. Statman.
Logical Relations and the Typed $\lambda$-Calculus.
*Information and Control*, 65:85–97, 1985.

[Sto88] Allen Stoughton.
*Fully Abstract Models of Programming Languages.*
Pitman, 1988.
Research Notes in Theoretical Computer Science.