A Domain-theoretic Model for a Higher-order Process Calculus

Radhakrishnan Jagadeesan
Prakash Panangaden*

TR 89-1058
November 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This research was supported by NSF grant CCR-8818979.

A Domain-theoretic Model for a Higher-order Process Calculus

Radhakrishnan Jagadeesan and Prakash Panangaden*
Computer Science Department, Cornell University

December 20, 1989

Abstract

In this paper we study a *higher-order* process calculus, a restriction of one due to Boudol, and develop an abstract, model for it. By abstract we mean that the model is constructed domain-theoretically and reflects a certain conceptual viewpoint about observability. It is not constructed from the syntax of the calculus or from computation sequences. We describe a new powerdomain construction that can be given additional algebraic structure that allows one to model concurrent composition, in the same sense that Plotkin's powerdomain can have a continuous binary operation defined on it to model choice. We show that the model constructed this way is adequate with respect to the operational semantics. The model that we develop and our analysis of it is closely related to the work of Abramsky and Ong on the lazy lambda calculus.

1 Introduction

A fundamental problem in the semantics of parallel programming languages is integrating concurrency with abstraction. In this paper we study a *higher-order* process calculus, a restriction of one due to Boudol [4], and develop an abstract, mathematical model for it. The restrictions we make simplify certain aspects of the calculus, for example, deadlock is not possible, but it preserves much of the complexity. In particular both concurrency and nondeterminism still exist. By abstract we mean that the model is constructed domain theoretically and reflects a certain conceptual viewpoint about observability. It is not constructed from the syntax of the calculus or from computation sequences. We describe a new powerdomain construction that can be given additional algebraic structure that allows one to model concurrent composition, in the same spirit that Plotkin's powerdomain can have a continuous binary operation defined on it to model choice. We show that the model constructed this way is adequate with respect to the operational semantics. The model

*This research supported by NSF grant CCR-8818979

that we develop and our analysis of it is closely related to the work of Abramsky and Ong [2] on the lazy lambda calculus.

Kahn's pioneering work on static dataflow [10] is an example where concurrency meshes smoothly with an abstraction. More precisely, in Kahn's model one can abstract away the internal operational details of processes and view them as continuous stream-functions that compose as functions should. Feedback is modeled by a standard fixed-point iteration. This is a very pleasant application of Scott's semantic ideas.

In almost any elaboration of Kahn's model the situation becomes much more difficult. In particular, though process algebra has now reached a high degree of mathematical maturity and elegance, see, for example, the recent books by Milner and by Hennessy [7,14], it remains essentially an operational analysis of processes. The semantic models available are constructed from the computations of terms.

In the context of indeterminate dataflow, recent work by Kok, Jonsson and others has shown that one gets fully abstract models from the traces of computations [9,11,20]. Traces do not, however, give one the same level of abstraction that is provided by being able to think of processes as functions.

Powerdomains [18] do give one an elegant model of certain parallel languages. They were defined essentially to combine *indeterminacy* with abstraction. When one attempts to use the Plotkin powerdomain to model concurrency as well one finds that certain operational laws are violated in the model. Our model, though not fully abstract, describes the interplay between choice, lambda abstraction and concurrency in a smooth way.

Recent advances in process algebra include the study of higher-order process calculi [15, 22]. The studies cited, as well as Boudol's presentation of his calculus, focus entirely on the operational semantics. There are other related developments, most importantly the development of label passing calculi by Milner and his co-workers and, independently by Engberg and Neilsen [6]. The relationship between these calculi and the higher-order process calculi remains to be understood. Though these systems are theoretical there are other closely related systems, in varying stages of formal analysis, that are actually implemented and are being used in experiments. The most interesting of these is Reppy's calculus that incorporates events as first-class entities in Standard ML [19]. Though our work does not directly bear on these activities it does indicate that these ideas are ripe for an intensive study.

Following Boudol, we want the λ-calculus to be embedded in the process calculus. Our study of the restricted version of Boudol's calculus, henceforth called the γ calculus, is based on viewing the communication abilities of processes as the fundamental observables. This is, in some sense, a natural extension of the idea of making convergence the basic observable in the λ-calculus. A process that is diverging has no communication ability, a process that can accept a single input and then diverges has more communication ability. The connection with λ-calculus comes about by observing that the presence of an outer λ-abstraction signifies that a term has communication ability. Clearly, we should distinguish $\lambda x.\Omega$ from Ω, where Ω represents any divergent term such as $(\lambda x.xx)(\lambda x.xx)$, since they have different communication abilities. Thus we need our model to resemble the models of the lazy λ-calculus [2] rather than the models discussed by Scott and Wadsworth [23]. The new ingredient that we need to deal with is the fact that a term may or may not converge; thus we need two predicates to capture the convergence properties of term; these are "may converge" and "must converge".

The key differences between the γ-calculus and other process calculi, for example CCS [12] are the following.

1. The γ-calculus has no notion of *sequential* composition of output actions; a term may just *be* output, it cannot *produce* output and then go on and do something else.

2. In CCS the parallel composition operator expresses two effects, viz. juxtaposition and interaction. In the γ-calculus these are described by two different operators, written $|$ and $\odot$. The latter is *not* associative, unlike in CCS where parallel composition is defined to be associative.

It is not clear what the relative expressive power of CCS and the γ-calculus is. Very likely CCS is more expressive, it appears highly unlikely that one could emulate the behaviour of the CCS composition operator using the operators of the γ-calculus. This does not mean that the γ-calculus is very weak. One can fairly easily code up the standard concurrent programming examples, mutual exclusion, schedulers etc., using the γ-calculus. The notion of bisimulation that we use in our analysis is different from what one sees in process calculi and also different from the one used by Boudol in his analysis.

The paper is organised as follows. In section 2 we introduce the γ-calculus and discuss it informally through some examples. In section 3 we define a sub-calculus and its operational semantics and introduce the bisimulation relation and show that bisimulation is a congruence. In section 4 we give the powerdomain construction, in section 5 we analyze the structure of the powerdomain in terms of its finite approximants. In section 6 we prove adequacy and make some remarks about full abstraction. In the final section we discuss related work and directions for further study.

2 The γ Calculus

In this section we quickly review Boudol's γ-calculus and describe an example of a simple concurrent program expressed in it. The key contribution of this calculus is to provide a smooth integration of concurrent communication concepts with functional abstraction. Boudol's original work [4] describes the calculus and shows how the lazy λ-calculus is embedded in it. We will not reproduce his discussion of the embedding.

Let C be a set of channel names. Terms are generated by the grammar:

$$Terms ::= x \;||\; (\lambda_1 x_1 | \ldots | \lambda_k x_k).p \;||$$

$$p \odot q \;||\; p|q \;||\; \overline{\lambda}p.1 \;||1$$

where λ, $\lambda_1 \ldots \lambda_k$ are (not necessarily distinct) members of C. The novel constructs here are $(\lambda_1 x_1 | \ldots | \lambda_k x_k)$, $\odot$, $\overline{\lambda}$ and $p|q$. The term 1 represents the terminated process. It will turn out to be the identity for both $\odot$ and $|$. Roughly speaking, $(\lambda_1 x_1 | \ldots | \lambda_k x_k).p$ means that p waits *concurrently* for k unordered values. The combinator$|$ represents concurrency. The intuitive meaning of $p|q$ is that p and q are juxtaposed, *without any communication between them.* The term $\overline{\lambda}p$ represents a process that outputs p on channel λ and terminates. Finally, $p \odot q$ means that p and q communicate on *all* channels. The processes p and q cannot communicate with any other process until one of them terminates. The key points to note are that $|$ represents pure concurrency without any interaction while $\odot$ represents a very tight interaction between processes.

The following transition system presented informally, expresses these intuitive ideas. First we begin by defining a syntactic congruence that expresses the fact that 1 is the terminated process.

Definition 1. The syntactic relation $\equiv$ is the congruence (with respect to substitution) that is generated by the following equations:

- $p \odot 1 \;\equiv\; 1 \odot p \;\equiv\; p$
- $p|1 \;\equiv\; 1|p \;\equiv\; p$
- $p|(q|r) \equiv\; (p|q)|r$

Let λ_i be channel names.

- $(M_1|\ldots|(\lambda_1 x_1|\ldots|\lambda_k x_k).M|\ldots|M_n)$
  $\odot\;(N_1|\ldots|\overline{\lambda_i}N|\ldots|N_r)$
  $\longrightarrow (M_1|(\lambda_1 x_1|\ldots|\lambda_{i-1}x_{i-1}|\lambda_{i+1}x_{i+1}\ldots|\lambda_k x_k).[x_i \mapsto N]M|M_{k+1}\ldots|M_n)$
  $\odot\;(N_1|N_2|\ldots|N_s|N_{s+1}\ldots|N_r)$

- $(N_1|N_2|\ldots|N_s|\overline{\lambda_i}N|N_{s+1}\ldots|N_r)\odot$
  $(M_1|\ldots M_k|(\lambda_1 x_1|\ldots\lambda_k x_k).M)|M_{k+1}\ldots|M_n)$
  $\longrightarrow (N_1|N_2|\ldots|N_s|N_{s+1}\ldots|N_r)\odot$
  $(M_1|\ldots M_k|(\lambda_1 x_1|\ldots|\lambda_{i-1}x_{i-1}|\lambda_{i+1}x_{i+1}\ldots|\lambda_k x_k).[x_i \mapsto N]M|M_{k+1}\ldots M_n)$

-
  - $M \longrightarrow M' \Rightarrow M|N \longrightarrow M'|N$
  - $N \longrightarrow N' \Rightarrow M|N \longrightarrow M|N'$

-
  - $M \longrightarrow M' \Rightarrow M \odot N \longrightarrow M' \odot N$
  - $N \longrightarrow N' \Rightarrow M \odot N \longrightarrow M \odot N'$

In the above $[x \mapsto N]M$ is notation for substitution. The $\odot$ serves as a generalization of application. The communication is effected in the manner now customary in process algebras, one matches a name with its *dual* name. Note how there is no communication between processes that are combined with $|$. Finally there is no construct like $\overline{\lambda}p.M$ where

M represents a term. An output term cannot produce a value and go on to do something else.

The following simple term that appers in Boudol's original paper [4], illustrates some of the features of the γ-calculus.

$$A \simeq \lambda y.\alpha x.(\overline{\beta}z|(y \odot \overline{\lambda}y))$$

Now consider $A \odot \overline{\lambda}A$. This term reduces in one step to

$$\alpha x.(\overline{\beta}z|(A \odot \overline{\lambda}A)))$$

This last term has the property that it waits for a signal on α then outputs z on β and reproduces itself. It is a term that repeatedly offers communication to the outside.

3 Operational Semantics

In this section we define the restricted calculus. From the point of view of difficulty of modeling we have eliminated the possibility of deadlock but we have left in the indeterminacy as well as the concurrency. We do not allow $\odot$ in its unrestricted form; we force it to look like application. More precisely, the $\odot$ construct can only be used in the combination $\lambda x.M \odot \overline{\lambda}P$. Thus it cannot be introduced in a case where there is no communication possibility as in $\lambda x.x \odot \lambda x.x$. The bulk of this section is an analysis of our notion of bisimulation.

The terms are generated by the grammar

$$Terms ::= x \;||\; \lambda\langle x_1 \dots x_k\rangle.p \;||\; pq \;||\; p|q$$

we do not use the $\odot$ symbol explicitly. It is implicitly present in the applications. We use Λ_0 for the terms that do not have free variables, and call members of this set closed terms.

Definition 2. The syntactic equality $\equiv$ is the congruence generated by the equation:

$$p|(q|r) \equiv (p|q)|r$$

Define, by mutual recursion:
$Terms_1 ::= x \;||\; \lambda\langle x_1 \dots x_k\rangle.p \;||\; pq$
$Terms_2 ::= p \;||\; p|q$
where $p,\ q \in Terms_1 \bigcup Terms_2$. Note that $Terms_2 = Terms$. Intuitively, $Terms_1$ are the terms without a $|$ at 'outermost level', and $Terms_2$ are the terms of the form $t_1|\dots|t_n$, where the t_i are either abstractions or applications. The following definition is intended to capture the "number" of $t_i's$.

Define $len : Terms_2 \rightarrow Int$ as follows:

- $len(p) = 1$, if $p \in Terms_1$

- $len(p|q) = len(p) + len(q)$

It can be checked that this function is well-defined on the terms quotiented by the syntactic equality $\equiv$. The following definition is intended to capture the "position" of t_i in $t_1|\ldots t_n$. Define a partial function $index : \omega \times Terms_2 \rightarrow Terms_1$ as follows:

- $index(n,p) =$ undefined if $len(p) \leq n \wedge len(p) \neq n$
- $index(1,p) = p$, if $p \in Terms_1$
- $index(n,p|q) = index(n,p)$, if $n \leq len(p)$
- $index(n,p|q) = index(n - len(p), q)$, if $len(p) \leq n \wedge len(p) \neq n$

The reduction rules are as follows. In this presentation, we have introduced notation (as a subscript of $\longrightarrow$) to keep track of the redices, explicitly. Usually, we ignore these subscripts.

- $(\lambda\langle x_1 \ldots x_k\rangle.M)N \longrightarrow_{\langle 1,i\rangle} \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M$
  if $1 \leq i \leq k$
- $index(s,(M_1|\ldots|\lambda\langle x_1 \ldots x_k\rangle.M|\ldots|M_n)) = \lambda\langle x_1 \ldots x_k\rangle.M \Rightarrow$
  $(M_1|\ldots|\lambda\langle x_1 \ldots x_k\rangle.M|\ldots|M_n)N$
  $\longrightarrow_{\langle s,i\rangle}$
  $M_1|\ldots|\lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M|\ldots|M_n$, if $1 \leq i \leq k$
- $M \longrightarrow_\sigma M' \Rightarrow M|N \longrightarrow_\sigma M'|N$
- $N \longrightarrow_\sigma N' \Rightarrow M|N \longrightarrow_{\sigma'} M|N'$
  where $\sigma' = \langle first(\sigma) + len(M), second(\sigma)\rangle$
- $M \longrightarrow_\sigma M' \Rightarrow MN \longrightarrow_{\langle 1,\sigma\rangle} M'N$

3.1 Bisimulation

The key feature to note is the lazy evaluation, i.e. there are no reductions inside output terms. Let $\mathcal{K}$ be the term $\lambda\langle x_1, x_2\rangle.x_1$. Note that $(\mathcal{K}M)N$ corresponds to non-deterministic choice between M and N. The presence of non-determinism means that the theory of equality induced by $\xrightarrow{*}$ is not consistent, i.e. all terms get identified.

Since the intuitive meaning that was asigned to $\lambda x.M$ was the presence of a communication ability on port λ, we attempt to set up a theory that "measures" the communication ability of a term. The study of the lazy lambda calculus [2], proceeds on very similar lines. There the "definedness" of a term is measured by its outermost abstractions or, in other words, how many arguments it can accept. This is exactly what we do except that we need to confront the indeterminacy in the reduction relation. The study of the lazy

λ-calculus motivates the definition of a convergence predicate. Notice that the presence of non-determinism means that for a given term M, we might have both the following situations:

- $M \xrightarrow{*} \lambda\langle x_1 \dots x_k\rangle.M'$
- An infinite reduction sequence
  $M = M_0 \longrightarrow M_1 \longrightarrow M_2 \dots$

The above discussion motivates the definitions of the predicates $\Downarrow^{may}$ read as "may converge" and $\Downarrow^{must}$ read as "must converge". Let the closed terms be denoted by Λ_0.

Definition 3. 1. $\lambda\langle x_1 \dots x_k\rangle.M\Downarrow^{may}$, $(\forall 1 \leq k).\ (\forall \lambda\langle x_1 \dots x_k\rangle.M \in \Lambda_0)$

2. $M\Downarrow^{may} \vee N\Downarrow^{may} \Rightarrow M|N\Downarrow^{may}$

3. $(\exists M')\ [M \xrightarrow{*} M' \wedge M'\Downarrow^{may}] \Rightarrow M\Downarrow^{may}$

Definition 4. $M\Downarrow^{must}$ if there is no infinite reduction sequence $M = M_0 \longrightarrow M_1 \longrightarrow M_2 \dots$

The predicates $\Downarrow^{may}$ and $\Downarrow^{must}$ are the only observables in the calculus. As a first attempt at relating communication abilities of terms, one might define
$M \leq N$ if,

- $M\Downarrow^{may} \Rightarrow N\Downarrow^{may}$
- $M\Downarrow^{must} \Rightarrow N\Downarrow^{must}$

We need, however, to measure much more when relating two terms operationally. Intuitively, we need to measure the communication abilities of M after it has been applied to some arguments. This motivates the following definitions. Define (on closed terms Λ_0):

1. $M \preceq_0 N$ if
   - $M\Downarrow^{may} \Rightarrow N\Downarrow^{may}$
   - $M\Downarrow^{must} \Rightarrow N\Downarrow^{must}$
2. $M \preceq_{k+1} N$ if
   - $M \preceq_0 N$
   - $(\forall P \in \Lambda_0)\ [MP \preceq_k NP]$

Definition 5. $\preceq = \bigcap \preceq_k,\ k \in \omega$

The idea of $\preceq$ is extended to open terms in the usual way. Let M, N be terms such that the free variables of M and N are contained in $\{x_1 \dots x_n\}$. Then, $M \preceq N$ if for all possible substitutions $P_1 \dots P_n$ of closed terms for $\{x_1 \dots x_n\}$, we have
$[x_1 \mapsto P_1 \dots x_n \dots P_n]M \preceq [x_1 \mapsto P_1 \dots x_n \dots P_n]N$.

The following lemma is easy to prove. Let M, N be closed terms.

Lemma 1. $M \preceq N \Leftrightarrow (\forall n)\,(\forall P_1 \ldots P_n \in \Lambda_0)\,[MP_1 \ldots P_n \preceq_0 NP_1 \ldots P_n]$

In CCS one can define the bisimulation relation either as a fixed point of an appropriate operator on the lattice of relations or inductively as we have done [13]. The bisimulation obtained that way does not handle divergence properly, for example NIL and a completely divergent process are identified. A version of bisimulation that does handle divergence properly was defined by Walker [24]. The CCS situation is complicated by the presence of silent actions. In the γ-calculus the bisimulation that we have defined inductively is easily seen to be given as a greatest fixed point. The handling of non-determinism arising from internal silent actions resembles closely the ideas in testing equivalence [5].

Define a function F on the relations of closed terms by:
$M\ F(R)\ N$ if

- $M \preceq_0 N$
- $(\forall P \in \Lambda_0)\,[(MP,\ NP) \in R]$

Lemma 2. $\preceq$ is the maximum fixed point of F

Proof F is monotone on relations ordered by $\subseteq$. From Tarski's fixed point theorem, F has a maximum fixed point. It easily follows from the previous lemma that the closure ordinal of F is in fact ω. ∎

The most important fact about $\preceq$ is that operational extensionality holds, i.e. bisimulation is contextual. The rest of this section is a rather long proof of this fact. A preliminary step is to show that $|$ behaves monotonically with respect to bisimulation.

3.2 Monotonicity of |

In this subsection, the monotonicity of $|$ with respect to $\preceq$ is proved. This is a prelude to the major result of this section namely that bisimulation is operationally extensional. This section may be skipped on a first reading[1] The main point is that proving that $|$ is monotone with respect to bisimulation requires an analysis of the interleavings of the reductions in each component. The proofs are not hard but they do require a rather careful analysis of reduction.

Lemma 3. 1. $(Q_1|Q_2)\Downarrow^{may} \Leftrightarrow Q_1\Downarrow^{may} \vee Q_2\Downarrow^{may}$

2. $(Q_1|Q_2)\Downarrow^{must} \Leftrightarrow Q_1\Downarrow^{must} \wedge Q_2\Downarrow^{must}$

Proof:

1. This follows directly from the definition.

2. Note that the transition system has the rules

[1]And in all subsequent readings as well.

- $M \longrightarrow M' \Rightarrow M|N \longrightarrow M'|N$
- $N \longrightarrow N' \Rightarrow M|N \longrightarrow M|N'$

Also, all transitions of $M|N$ are of the above type. The result follows. ∎

The following lemma involves interleaving the reductions. The basic idea is best illustrated with a simple example. Consider $(Q_1|Q_2)P_1P_2$. It may converge if Q_2P_2 may converge; in order for this to happen, however, it has to be the case that Q_1 may converge in order for it to be possible for Q_1 to accept the argument P_1. Thus one has to describe the effects of partitioning the arguments to a parallel composition in the following fashion.

Lemma 4. Let $0 \leq n$. Then $(Q_1|Q_2)P_1 \dots P_n \Downarrow^{may} \Leftrightarrow$
$(\exists \langle i_1 \dots i_k \rangle, \langle j_1 \dots j_l \rangle$ such that

- $\langle i_1 \dots i_k \rangle$ and $\langle j_1 \dots j_l \rangle$ are (possibly empty) strictly increasing sequences of integers from $\{1 \dots n\}$
- $k + l = n$
- At least one of the following hold:
    1. $(Q_1P_{i_1} \dots P_{i_k})\Downarrow^{may} \wedge (Q_2P_{j_1} \dots P_{j_{l-1}})\Downarrow^{may}$, or
    2. $(Q_1P_{i_1} \dots P_{i_{k-1}})\Downarrow^{may} \wedge (Q_2P_{j_1} \dots P_{j_l})\Downarrow^{may}$

Proof:

1. (Reverse direction) Let $0 \leq n$. Assume that
$(\exists \langle i_1 \dots i_k \rangle, \langle j_1 \dots j_l \rangle$ such that
    - $\langle i_1 \dots i_k \rangle$ and $\langle j_1 \dots j_l \rangle$ are (possibly empty) strictly increasing sequences of integers from $\{1 \dots n\}$
    - $k + l = n$
    - $(Q_1P_{i_1} \dots P_{i_k})\Downarrow^{may} \wedge (Q_2P_{j_1} \dots P_{j_{l-1}})\Downarrow^{may}$

   (The case when $(Q_1P_{i_1} \dots P_{i_{k-1}})\Downarrow^{may} \wedge (Q_2P_{j_1} \dots P_{j_l})\Downarrow^{may}$ is proved by a similar argument).
   The proof is by induction on n. Base case, $n = 0$ follows from part 1 of lemma 3. Assume result for $n = s$. Let $n = s + 1$. Consider $(Q_1|Q_2)P_1 \dots P_n$. Note that we have $i_1 = 1$ or $j_1 = 1$. Without loss of generality, assume that $i_1 = 1$. (The other case can be proved by an argument similar to the one below). From the assumption that $(Q_1P_{i_1} \dots P_{i_{k-1}})\Downarrow^{may}$, it can be deduced that there is a reduction $Q_1 \xrightarrow{*} Q_1'$, where Q_1' is of form
   $(M_1| \dots |\lambda\langle x_1 \dots x_g \rangle.M| \dots |M_t)$, for some $0 \leq t$, such that
   $(M_1| \dots |\lambda\langle x_1 \dots x_{h-1}, x_{h+1} \dots x_g \rangle.[x_i \mapsto P_{i_1}]M| \dots |M_t)P_{i_2} \dots P_{i_{k-1}}\Downarrow^{may}$.
   Note that we have,

- $(Q_1|Q_2)P_1 \ldots P_n \xrightarrow{*}$
  $(M_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_g\rangle.[x_h \mapsto P_{i_1}]M|\ldots|M_t|Q_2)P_2 \ldots P_n$
- $(M_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_g\rangle.[x_h \mapsto P_{i_1}]M|\ldots|M_t)P_{i_2} \ldots P_{i_k}\Downarrow^{may}$
- $(Q_2 P_{j_1} \ldots P_{j_{l-1}})\Downarrow^{may}$
- $k-1+l = n-1$
- $\langle i_2 \ldots i_{k-1}\rangle$ and $\langle j_1 \ldots j_l\rangle$ are (possibly empty) strictly increasing sequences of integers from in $\{2 \ldots n\}$

So, the inductive hypothesis can be used. i.e
$(M_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_g\rangle.[x_h \mapsto P_{i_1}]M|\ldots|M_t|Q_2)P_2 \ldots P_n\Downarrow^{may}$. Thus, we have
$(Q_1|Q_2)P_1 \ldots P_n\Downarrow^{may}$.

2. (Forward direction)
The proof proceeds by induction on r, where r is the length of the reduction
$(Q_1|Q_2)P_1 \ldots P_n \xrightarrow{*} M$, such that M is of form
$(M_1|\ldots|\lambda\langle x_1 \ldots x_s\rangle.M|\ldots|M_t)$. Note that the base case ($r = 0$) follows immediately. Let $(Q_1|Q_2)P_1 \ldots P_n \longrightarrow M'$ be the first step of the reduction sequence
$(Q_1|Q_2)P_1 \ldots P_n \xrightarrow{*} M$. We have the following (mutually exclusive) cases depending on the reduction
$(Q_1|Q_2)P_1 \ldots P_n \longrightarrow M'$.

   - $Q_1 \longrightarrow Q_1'$, and $M' = (Q_1'|Q_2)P_1 \ldots P_n$. Result follows by the induction hypothesis on $(Q_1'|Q_2)P_1 \ldots P_n$
   - $Q_2 \longrightarrow Q_2'$, and $M' = (Q_1|Q_2')P_1 \ldots P_n$. Result follows by the induction hypothesis on $(Q_1|Q_2')P_1 \ldots P_n$.
   - The first step is a β reduction that involves P_1. Without loss of generality, assume that Q_1 has form
     $(N_1|\ldots|\lambda\langle x_1 \ldots x_s\rangle.N|\ldots|N_g)$, for some $0 \leq g$, and the first step is
     $(N_1|\ldots|\lambda\langle x_1 \ldots x_s\rangle.N|\ldots|N_g|Q_2)P_1 \ldots P_n \longrightarrow$
     $(N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g|Q_2)P_2 \ldots P_n$, where $1 \leq h \leq s$. (The case in which Q_2 has this form can be handled similarly). Notice that this term satisfies the induction hypothesis. Let
     $Q_1' = (N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g)$.
     So, we have
     $(\exists\langle i_1 \ldots i_k\rangle, \langle j_1 \ldots j_l\rangle$ such that
     - $\langle i_1 \ldots i_k\rangle$ and $\langle j_1 \ldots j_l\rangle$ are (possibly empty) strictly increasing sequences of integers from $\{2 \ldots n\}$
     - $k + l = n - 1$
     - At least one of the following hold:

(a) $(Q_1' P_{i_1} \dots P_{i_k}) \Downarrow^{may} \wedge (Q_2 P_{j_1} \dots P_{j_{l-1}}) \Downarrow^{may}$, or
(b) $(Q_1' P_{i_1} \dots P_{i_{k-1}}) \Downarrow^{may} \wedge (Q_2 P_{j_1} \dots P_{j_l}) \Downarrow^{may}$

. The result follows for $(Q_1|Q_2)P_1 \dots P_n$ by setting

- The sequence for Q_1 is the sequence got by adding 1 to the sequence for Q_1'
- The sequence for Q_2 is the same sequence as that obtained from the induction hypothesis. ∎

The must converge situation is rather like the may converge situation but is more natural to state.

Lemma 5. Let $0 \leq n$. Then $(Q_1|Q_2)P_1 \dots P_n \Downarrow^{must} \Leftrightarrow$
$(\forall \langle i_1 \dots i_k \rangle, \langle j_1 \dots j_l \rangle$ such that

- $\langle i_1 \dots i_k \rangle$ and $\langle j_1 \dots j_l \rangle$ are (possibly empty) strictly increasing sequences of integers from $\{1 \dots n\}$
- $k + l = n$
- Both of the following hold:
  1. $(Q_1 P_{i_1} \dots P_{i_k}) \Downarrow^{must} \wedge (Q_2 P_{j_1} \dots P_{j_l}) \Downarrow^{must}$
  2. $(Q_1 P_{i_1} \dots P_{i_k}) \Downarrow^{must} \wedge (Q_2 P_{j_1} \dots P_{j_l}) \Downarrow^{must}$

Proof:

- (Forward implication)
  Proof is by induction on n. Part 2 of lemma 3 proves the base case. Assume the result for $n = s$. Consider $n = s + 1$. Let
  - $P_1 \dots P_n \in \Lambda_0$
  - $\langle i_1 \dots i_k \rangle$, $\langle j_1 \dots j_l \rangle$ be such that
    * $(Q_1|Q_2)P_1 \dots P_n \Downarrow^{must}$
    * $\langle i_1 \dots i_k \rangle$ and $\langle j_1 \dots j_l \rangle$ are (possibly empty) strictly increasing sequences of integers from $\{1 \dots n\}$
    * $k + l = n$

  Without loss of generality, assume that $i_1 = 1$. (The case in which $j_1 = 1$ can be handled similarly.) Note that
  $(Q_1|Q_2)P_1 \dots P_n \Downarrow^{must} \Rightarrow Q_1 \Downarrow^{must} \wedge Q_2 \Downarrow^{must}$. Consider any reduction sequence
  $Q_1 \overset{*}{\longrightarrow} N_1 | \dots | \lambda \langle x_1 \dots x_s \rangle . N | \dots | N_g$
  Consider any possible reduction $Q_1 P_{i_1} \dots P_{i_k} \longrightarrow$
  $(N_1 | \dots | \lambda \langle x_1 \dots x_{h-1}, x_{h+1} \dots x_s \rangle . [x_h \mapsto P_1] N | \dots | N_g) P_{i_2} \dots P_{i_k}$.
  Note that

- $(Q_1|Q_2)P_1 \ldots P_n \xrightarrow{*}$
  $(N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g|Q_2)P_2 \ldots P_n$
- $(N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g|Q_2)P_2 \ldots P_n {\Downarrow}^{must}$

From the induction hypothesis,

- $(Q_2 P_{j_1} \ldots P_{j_l}){\Downarrow}^{must}$
- $(N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g)P_{i_2} \ldots P_{i_k} {\Downarrow}^{must}$

The result follows, since the above argument holds for *ALL* possible reduction sequences of Q_1.

- (Reverse implication)
  Proof is by induction on n. Part 2 of this lemma proves the base case. Note that we have
  $(\forall\langle i_1 \ldots i_k\rangle,\ \langle j_1 \ldots j_l\rangle$ such that

  - $\langle i_1 \ldots i_k\rangle$ and $\langle j_1 \ldots j_l\rangle$ are (possibly empty) strictly increasing sequences of integers from $\{1 \ldots n\}$
  - $k + l = n$

  is true, both of the following hold:

  1. $(Q_1 P_{i_1} \ldots P_{i_k}){\Downarrow}^{must} \ \wedge (Q_2 P_{j_1} \ldots P_{j_l}){\Downarrow}^{must}$
  2. $(Q_1 P_{i_1} \ldots P_{i_k}){\Downarrow}^{must} \ \wedge (Q_2 P_{j_1} \ldots P_{j_l}){\Downarrow}^{must}$

  Hence, we deduce $Q_1{\Downarrow}^{must} \ \wedge \ Q_2{\Downarrow}^{must}$. Consider any reduction sequence r of $(Q_1|Q_2)P_1 \ldots P_n$. From the above remark, we note that there is an initial segment of the above reduction sequence such that $(Q_1|Q_2)P_1 \ldots P_n \xrightarrow{*}$
  $(N_1|\ldots|\lambda\langle x_1 \ldots x_s\rangle.N|\ldots|N_g)P_1 \ldots P_n$, and the next term in the reduction sequence is
  $(N_1|\ldots|\lambda\langle x_1 \ldots x_{h-1}, x_{h+1} \ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g)P_2 \ldots P_n$.
  Without loss of generality, assume that

  - $Q_1 \xrightarrow{*} (N_1|\ldots|\lambda\langle x_1 \ldots x_s\rangle.N|\ldots|N_f)$
  - $Q_2 \xrightarrow{*} (N_{f+1}|\ldots N_g)$

  (The symmetric case with the roles of Q_1 and Q_2 reversed can be handled similarly). From assumption of part of lemma that is being proved, we get $(\forall\langle i_1 \ldots i_k\rangle,\ \langle j_1 \ldots j_l\rangle$ such that

  - $\langle i_1 \ldots i_k\rangle$ and $\langle j_1 \ldots j_l\rangle$ are (possibly empty) strictly increasing sequences of integers from $\{2 \ldots n\}$
  - $k + l = n - 1$

is true, both of the following hold,

1. $(N_1|\ldots|\lambda\langle x_1\ldots x_{h-1}, x_{h+1}\ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_f)P_{i_1}\ldots P_{i_k})\Downarrow^{must}$
2. $(N_{f+1}|\ldots N_g)P_{j_1}\ldots P_{j_l})\Downarrow^{must}$

Using the induction hypothesis, we have
$(N_1|\ldots|\lambda\langle x_1\ldots x_{h-1}, x_{h+1}\ldots x_s\rangle.[x_h \mapsto P_1]N|\ldots|N_g)P_2\ldots P_n\Downarrow^{must}$. In particular, r terminates. ∎

Lemma 6. $M\preceq M' \wedge N\preceq N' \Rightarrow M|N\preceq M'|N'$

Proof: Follows easily from lemma 1 and the above lemma. ∎

3.3 Operational Extensionality

The idea of the proof is quite simple but the details are a little complicated since one has to keep track of redices carefully. The basic idea is as follows. Suppose that $N\preceq M$, we want to show that for any context, $C[]$, $C[N]\preceq C[M]$. Given any terms $P_1,\ldots,P_j$ we need to show that if $C[N]P_1,\ldots,P_j\Downarrow^{may}$ then $C[M]P_1,\ldots,P_j\Downarrow^{may}$ as well; there is an analogous condition with $\Downarrow^{must}$. If the reduction occur only inside the context the result is immediate. Thus what we need to keep track of is when terms are inserted into the "functional" position in a context. The structure of the proof resembles the structure of the corresponding proof for the lazy lambda calculus.

Two reductions $M\longrightarrow_{\sigma'}M'$, and $M\longrightarrow_{\sigma''}M''$ are different if $\sigma' \neq \sigma''$. Note that there are only finitely many different reductions. Let $M \in \Lambda_0$. Construct a tree with labelled edges corresponding to M denoted by $T(M)$ as follows. Let $M\longrightarrow_{\sigma_i}M_i$, be all the possible different one step reductions from M. Then, the root has an edge for each label σ_i. The subtree at the node at the other end of the edge with label σ_i is the one obtained by doing the construction for M_i. Also, by a Konig's lemma argument, we deduce that $M\Downarrow^{must} \Rightarrow T(M)$ is finite.

Definition 6. Let $\langle D, \leq\rangle$ be the domain of labelled, finitely-branching trees of finite depth, where the ordering relation $\leq$ is the subtree ordering.

Note that $\langle D, \leq\rangle$ is well-founded. Furthermore, if $M\Downarrow^{must}$ and $M\longrightarrow M'$, then $T(M') \leq T(M)$.

Define the contexts C[] with holes by the following grammar:
$C[] ::= x \mid\mid [] \mid\mid C_1[]|C_2[] \mid\mid \lambda\langle x_1\ldots x_k\rangle.C_1[] \mid\mid (C_1[])(C_2[])$
The following definition is intended to capture the idea of a hole occurring in a "functional" position.

Definition 7. • [] occurs functionally in []

• [] occurs functionally in $C_1[]|C_2[]$ if at least one of the following hold:

  - $[]$ occurs functionally in $C_1[]$
  - $[]$ occurs functionally in $C_2[]$

- $[]$ occurs functionally in $(C_1[])(C_2[])$ if
  $[]$ occurs functionally in $C_1[]$

Lemma 7. The contexts $D[]$ such that $[]$ does not occur functionally in $D[]$ are generated by the following grammar:
$D[] ::= x \mid\mid D_1[] | D_2[] \mid\mid \lambda\langle x_1 \ldots x_k\rangle C[] \mid\mid (D[])(C[])$
where $C[]$ is ANY context at all

Proof: Structural induction ∎

Let $C[]$ be any context with a hole. Let M be any term. Then
$[[] \mapsto M]C[]$ is the term got by substituting M for $[]$ in $C[]$, and is usually denoted by $C[M]$.

Let M be any term. We define the notion of *substituting M for the functional occurrences of* $[]$ in $C[]$, denoted by $[[] \mapsto_f M]C[]$ by structural induction on $C[]$.

- $[[] \mapsto_f M]x = x$
- $[[] \mapsto_f M][] = M$
- $[[] \mapsto_f M](C_1[]|C_2[]) = ([[] \mapsto_f M]C_1[])|([[] \mapsto_f M]C_2[])$
- $[[] \mapsto_f M]\lambda\langle x_1 \ldots x_k\rangle.C_1[] = \lambda\langle x_1 \ldots x_k\rangle.C_1[]$
- $[[] \mapsto_f M](C_1[])(C_2[]) = ([[] \mapsto_f M]C_1[])(C_2[])$

Note that $[]$ does not occur functionally in $[[] \mapsto_f M]C[]$.

Lemma 8. Let $(\forall P \in Terms)[([[] \mapsto_f P]D[]) \equiv ([[] \mapsto P]D[]) \in Terms]$. Then,
$[(M \preceq N) \Rightarrow [[] \mapsto M]D[] \preceq [[] \mapsto M]D[]]$

Proof: Note that the hypothesis of the lemma means that $D[]$ is a member of the contexts generated by the grammar:
$E[] ::= x \mid\mid [] \mid\mid (E_1[]|E_2[]) \mid\mid \lambda\langle x_1 \ldots x_k\rangle.P \mid\mid (C_1[])P$
where P is any term of the calculus. Proof now follows by structural induction. (Monotonicity of $|$ is used in a case) ∎

A symmetric notion of *substituting M for the non-functional occurrences of* $[]$ in $C[]$, denoted by $[[] \mapsto_{nf} M]C[]$ is defined by structural induction on $C[]$.

- $[[] \mapsto_{nf} M]x = x$
- $[[] \mapsto_{nf} M][] = []$
- $[[] \mapsto_{nf} M](C_1[]|C_2[]) = ([[] \mapsto_{nf} M]C_1[])|([[] \mapsto_{nf} M]C_2[])$

- $[[] \mapsto_{nf} M]\lambda\langle x_1 \ldots x_k\rangle.C_1[] = \lambda\langle x_1 \ldots x_k\rangle.[[] \mapsto M]C_1[]$
- $[[] \mapsto_{nf} M](C_1[])(C_2[]) = ([[] \mapsto_{nf} M]C_1[])([[] \mapsto M]C_2[])$

Note that
$C[M] = [[] \mapsto_f M]([[] \mapsto_{nf} M]C[]) = [[] \mapsto_{nf} M]([[] \mapsto_f M]C[])$.
Define a syntactic equality on contexts as follows:

Definition 8. The syntactic equality $\equiv$ is the congruence (with respect to substitution) that is generated by the equation:
$p|(q|r) \equiv\ (p|q)|r$

Define a reduction relation on contexts as follows:

- $(\lambda\langle x_1 \ldots x_k\rangle.C_1[])C_2[] \longrightarrow \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto C_2[]]C_1[]$
  if $1 \leq i \leq k$
- $(C_1[]| \ldots |\lambda\langle x_1 \ldots x_k\rangle.C[]| \ldots |C_n[])C'[] \longrightarrow$
  $C_1| \ldots |\lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto C'[]]C[]| \ldots |C_n[]$, if $1 \leq i \leq k$
- $C_1[] \longrightarrow C_1'[] \Rightarrow C_1[]|C_2[] \longrightarrow C_1'[]|C_2[]$
- $C_2[] \longrightarrow C_2'[] \Rightarrow C_1[]|C_2[] \longrightarrow C_1[]|C_2'[]$
- $C_1[] \longrightarrow C_1'[] \Rightarrow (C_1[])(C_2[]) \longrightarrow (C_1'[])(C_2[])$

Lemma 9. $[]$ does not occur functionally in $C[] \Rightarrow$
$(C[M] \longrightarrow T \Rightarrow C[] \longrightarrow D[] \wedge\ D[M] \equiv\ T)$

Proof: Structural induction, and the characterisation of contexts of hypothesis of lemma, as in lemma 7 . ∎

Lemma 10. $P \preceq Q \Rightarrow [(\forall C[])\ [C[P]\Downarrow^{may} \Rightarrow\ C[Q]\Downarrow^{may}]]$

Proof: Proof proceeds by induction on the length n of the reduction
$C[P] \xrightarrow{*} (N_1| \ldots |\lambda\langle x_1 \ldots x_s\rangle.N| \ldots |N_f)$. Note that the case $n = 0$ is immediate. Assume the result for $n = s$. We have the following two (mutually exclusive) cases.

- ($[]$ does not occur functionally in $C[]$).
  Then, we have $C[M] \longrightarrow T \Rightarrow C[] \longrightarrow D[] \wedge\ D[M] \equiv\ T$, from lemma 9. Since $D[M]\Downarrow^{may}$ in s steps, from the induction hypothesis, $D[N]\Downarrow^{may}$. Since $C[N] \longrightarrow D[N]$, $C[N]\Downarrow^{may}$.
- ($[]$ occurs functionally in $C[]$)
  Define $D[] = [[] \mapsto_f M]C[]$. Note that $D[M] = C[M]$. Since, $D[M]\Downarrow^{may}$, from case the above $D[N]\Downarrow^{may}$. Note that
  $D[N] =\ [[] \mapsto_{nf} N]([[] \mapsto_f M]C[]) =\ [[] \mapsto_f M]([[] \mapsto_{nf} N]C[])$
  From lemma 8, we deduce that
  $C[N] =\ [[] \mapsto_f N]([[] \mapsto_{nf} N]C[])\Downarrow^{may}$ ∎.

Lemma 11. $P \preceq Q \Rightarrow [(\forall C[])\ [C[P]\Downarrow^{must} \Rightarrow C[Q]\Downarrow^{must}]]$

Proof: Proof proceeds by induction on $T(C[M])$. The base case is immediate. For the induction step, we have the following two (mutually exclusive) cases.

- ($[]$ does not occur functionally in $C[]$).
  Then, we have $C[N] \longrightarrow T \Rightarrow C[] \longrightarrow D[] \wedge D[N] \equiv T$ from lemma 9. So $C[M] \longrightarrow D[M]$. Since $D[M]\Downarrow^{must}$ and $T(D[M]) \leq T(C[M])$, the induction hypothesis can be used to deduce $D[N]\Downarrow^{must}$. This is true for any reduction of $C[N]$. Hence, $C[N]\Downarrow^{must}$.

- ($[]$ occurs functionally in $C[]$
  Define $D[] = [[] \mapsto_f M]C[]$. Note that $D[M] = C[M]$. Since, $D[M]\Downarrow^{must}$, from case the above $D[N]\Downarrow^{must}$. Note that
  $D[N] = [[] \mapsto_{nf} N]([[] \mapsto_f M]C[]) = [[] \mapsto_f M]([[] \mapsto_{nf} N]C[])$
  From lemma 8, we deduce that
  $C[N] = [[] \mapsto_f N]([[] \mapsto_{nf} N]C[])\Downarrow^{must}$ ∎.

With these lemmas in hand the proof of operational extensionality is complete.

Theorem 1. (Operational extensionality) $M \preceq N \Leftrightarrow (\forall C[.])\ [C[M] \preceq C[N]]$

Proof: From lemmas 10 and 11 we immediately get $M \preceq N \Leftrightarrow (\forall C[.])\ [C[M] \preceq_0 C[N]]$. Clearly, any terms that we wish to use as arguments to $C[]$ can be absorbed into another context $D[]$. Thus $M \preceq N \Leftrightarrow (\forall C[.])\ [C[M] \preceq C[N]]$. ∎

4 The Powerdomain Construction

In this section we define the powerdomain construction that we use. We introduce it as a functor in a certain category of nondeterministic continuous algebras. We obtain a model of the γ-calculus by constructing a solution to a recursive domain equation in the usual way [21]. We show how one can define the product structure, needed to model the $|$ construct of the γ-calculus. inductively in the iterates that arise in the inductive construction of the initial solution. Finally, we describe in detail certain index calculations that are needed in the adequacy proof.

Many of the ideas are the same as in the analysis of the lazy λ-calculus but the details are somewhat more complicated. The product structure is new and the powerdomain construction itself is new. Before we begin with the mathematical details we discuss some motivational issues. As the adequacy proof shows, semantic equality in our model is at least as fine as bisimulation. We are almost certain that one could construct an adequate model for the fragment of the γ-calculus that we consider using the Plotkin powerdomain [18]. Why, then, did we choose to use this powerdomain rather than Plotkin's?

Our model is probably not fully abstract but it is, in some sense, "closer" to being fully abstract than a model based on the Plotkin powerdomain would be. In order to say what we mean more clearly we discuss an example. We abbreviate the term $\lambda\langle x, y\rangle.x$ as *or* and write it in infix form for readability. One easily checks that $\lambda x.[\Omega \ or\ \lambda y.\Omega]$ is bisimular to $\lambda xy.\Omega \ or\ \lambda x.\Omega$. In our model these terms are equated but in a model based on the Plotkin powerdomain they would not be. From our point of view, these terms should be equated since they offer the same communication abilities. The key property that distinguishes our powerdomain from the Plotkin powerdomain is the following:

$$f \sqsubseteq g \Leftrightarrow \begin{cases} \perp \in g \Rightarrow \perp \in f \\ f \neq \{\perp\} \Rightarrow g \neq \{\perp\} \\ \forall x. f \diamond x \sqsubseteq g \diamond x \end{cases}$$

The preceding holds in our powerdomain but not in the Plotkin powerdomain. In the Plotkin powerdomain, only the left to right implication holds.

Basic notation

All domains in this section are SFP objects. We use $B(D)$ as notation for the basis of D. We follow the notation of the work of S. Abramsky and L. Ong on the lazy lambda calculus [2]. Recall the definition of lifting as the left adjoint of the forgetful functor from $CPO_\perp$ to CPO where $CPO_\perp$ is the subcategory of strict functions. Let D, E objects of CPO. Let $f \in D \to E$.

- $D_\perp$ is the cpo defined as follows:
  - $|D_\perp| = \{\perp\} \bigcup \{\langle 0, d\rangle | d \in D\}$
  - Let y, $z \in D_\perp$. Then
    $y \sqsubseteq z \Leftrightarrow y = \perp \ \vee \ [y = \langle 0, d_1\rangle \ \wedge \ z = \langle 0, d_2\rangle \wedge \ d_1 \sqsubseteq_D d_2$
- If $d \in D$, define $up(d) = \langle 0, d\rangle$
- Define $lift(f) \in D_\perp \to E$ by:
  - $lift(f)\ (\perp) = \perp_E$
  - $lift(f)\ (\langle 0, d\rangle) = f\ (d)$
- Let $dn_D = lift(id_D)$
- $\diamond : (D_1 \to D_2)_\perp \times D_1 \to D_2$ is defined by
  - $\perp \diamond x = \perp$
  - $up(f) \diamond x = f(x)$, where $f \in D_1 \to D_2$

Definition 9. $\langle D, \star\rangle$ is a continuous algebra if $\star$ is a continuous function in $D \times D \to D$, satisfying upper semi-lattice axioms

Definition 10. Let $\langle D_1, \star_1\rangle$ and $\langle D_2, \star_2\rangle$ be continuous algebras. Let $f \in D_1 \rightarrow D_2$. f is said to be **linear** if
$(\forall\{x_1, x_2\} \subseteq D_1)\,[f(x_1 \star_1 x_2) = f(x_1) \star_2 f(x_2)]$

Definition 11. Let $\langle D_1, \star_1\rangle$ and $\langle D_2, \star_2\rangle$ be continuous algebras. Then, (e,p) is a linear embedding-projection pair if the following hold:

- $p \circ e = 1_{D_1}$
- $e \circ p \sqsubseteq 1_{D_2}$
- e is linear
- p is linear

Given any continuous algebra D, If $s = \{f_1 \ldots f_n\}$ where $(\forall i)$ $[1 \leq i \leq n]$ $[f_i \in (D \rightarrow D)_{\perp}]$, and $x \in D$ then $s \diamond x$ is notation for $f_1 \diamond x \star f_2 \diamond x \ldots \star f_n \diamond x$.

4.1 The Powerdomain Functor

In this subsection we define the powerdomain functor and show that it is continuous on a category of algebras very closely related to the bifinites (SFP).

Definition 12. Let $\langle D, \star\rangle$ be a continuous algebra. Then $P(D)$ is a preorder defined as follows:

- $|P(D)| = \{s | s \in P_{fin}(B((D \rightarrow D)_{\perp}))$
- $s_1 \sqsubseteq s_2 \Leftrightarrow$
  1. $\perp \in s_2 \Rightarrow \perp \in s_1$
  2. $s_1 \neq \{\perp\} \Rightarrow s_2 \neq \{\perp\}$
  3. $(\forall x \in D)[s_1 \diamond x \sqsubseteq s_2 \diamond x]$

$\overline{P}(D)$ is the ideal completion of $P(D)$. We now define a union operation on $\overline{P}(D)$ to make it a continuous algebra. Define $\uplus$ from $P(D) \times P(D)$ to $P(D)$ by $s_1 \uplus s_2 = s_1 \bigcup s_2$

Lemma 12. $\uplus$ is monotone in each argument.

Proof: Conditions 1 and 2 in the definition 12 of the preordering relation are easy to check. Condition 3 follows by noting that $(s_1 \uplus s_2) \diamond x = (s_1 \diamond x) \star (s_2 \diamond x)$, and from the monotonicity of $\star$. ∎

So $\uplus$ can be extended to a continuous function from $\overline{P}(D) \times \overline{P}(D)$ to $\overline{P}(D)$. The upper semi-lattice axioms are easy to check for the members of $P(D)$ and the continuity of $\uplus$ enables us to verify the laws for members of $\overline{P}(D)$.

We hope to solve the recursive domain equation $D \simeq \overline{P}(D)$. So, we need to establish a suitable category in which the above construction generalises to a functor preserving colimits of ω- chains. Define the category $NSFP$ as follows:

- Objects:
  The objects are continuous algebras expressible as the colimits of ω-chains of finite continuous algebras, where the arrows of the chain are linear embedding-projection pairs.

- Arrows:
  The arrows are linear embedding projection pairs.

In particular, all the objects are SFP objects. The above category can be viewed intuitively as that obtained by adding colimits of countable directed diagrams of finite continuous algebras, where the arrows of the diagram are linear embedding-projection pairs. Also note that the category is a subcategory of SFP^{ep} that contains the image of the Plotkin-powerdomain functor acting on SFP^{ep}, where SFP^{ep} is the category of SFP objects with arrows embedding-projection pairs.

Lemma 13. (Existence of colimits)

- $NSFP$ is closed under colimits of countable directed diagrams
- The one element domain is the initial object

The recursive domain equation $D \simeq \overline{P}(D)$ can be solved in $NSFP$ if we can prove that $\overline{P}(.)$ is a functor on $NSFP$ that preserves colimits of ω-chains. We now define the action of $\overline{P}(.)$ on linear embedding projection pairs. Let $\langle D_1, \star_1 \rangle$ and $\langle D_2, \star_2 \rangle$ be continuous algebras. Let (e, p), be a linear embedding-projection pair. Define e' as follows:

- Define $e' : P(D_1) \rightarrow \overline{P}(D_2)$ by:
  - $e'(\bot) = \bot$
  - $e'(\langle 0, f \rangle) = up\,(e \circ dn\,(f) \circ p)$
  - $e'\{F_1 \ldots F_n\} = \{e'(F_1) \ldots e'(F_n)\}$

  We need to show that e' is well-defined and monotone.

Lemma 14. Let $s_1 = \{F_1 \ldots F_n\}$ and $s_2 = \{H_1 \ldots H_m\}$ be elements of $P(D_1)$, such that $s_1 \sqsubseteq s_2$. Then $e'(s_1) \sqsubseteq e'(s_2)$.

Proof: We check the three conditions of Definition 12.

1. $\bot \in e'(s_2)$
   $\Rightarrow \bot \in s_2$ [from definition of e']
   $\Rightarrow \bot \in s_1$ [as $s_1 \sqsubseteq s_2$]
   $\Rightarrow \bot \in e'(s_1)$

2. $\{\bot\} \neq e'(s_1)$
   $\Rightarrow \{\bot\} \neq s_1$ [from definition of e']
   $\Rightarrow \{\bot\} \neq s_2$ [as $s_1 \sqsubseteq s_2$]
   $\Rightarrow \{\bot\} \neq e'(s_2)$

3. Let $x \in D_2$. Using the linearity of e, it follows that $e'(s_1) \diamond x = e(s_1 \diamond p(x))$. Similarly, we can deduce $e'(s_2) \diamond x = e(s_2 \diamond p(x))$. Since $s_1 \sqsubseteq s_2$, we have $s_1 \diamond p(x) \sqsubseteq s_2 \diamond p(x)$. The result follows from the monotonicity of e. ∎

Thus e' is well-defined, monotone and extends uniquely to a continuous function from $\overline{P}(D_1)$ to $\overline{P}(D_2)$. Furthermore, it follows immediately from the definition of e' that $e'(s_1 \uplus s_2) = e'(s_1) \uplus e'(s_2)$, for $s_1, s_2 \in P(D_1)$. The result, for arbitrary elements of $\overline{P}(D_1)$, follows from the continuity of all the functions involved.

The situation for p' is almost identical. Define p' as follows:

- $p'(\bot) = \bot$
- $p'(\langle 0, g \rangle) = up\,(p \circ dn\,(g) \circ e)$
- $p'\{F_1 \ldots F_n\} = \{p'(F_1) \ldots p'(F_n)\}$

Lemma 15. Let t_1 and t_2 be elements of $P(D_2)$, such that $t_1 \sqsubseteq t_2$. Then $p'(t_1) \sqsubseteq p'(t_2)$.

Proof: Similar to the previous lemma. ∎

Thus p' is also well-defined, monotone and extends uniquely to a continuous function from $\overline{P}(D_2)$ to $\overline{P}(D_1)$. Also, it follows immediately from the definition of p' that $p'(t_1 \uplus t_2) = p'(t_1) \uplus p'(t_2)$, for $t_1, t_2 \in P(D_2)$. The result, for arbitrary elements of $\overline{P}(D_2)$, follows from the continuity of all functions involved.

Since e', p' are linear and continuous, the proof that (e', p') is a linear embedding projection pair reduces to the following lemma.

Lemma 16. Let $F \in P(D_1)$, $G \in P(D_2)$ be singleton sets.So, $F \in B((D_1 \rightarrow D_1)_\bot)$, $G \in B((D_2 \rightarrow D_2)_\bot$. Then,

- $p' \circ e'(F) = F$
- $e' \circ p'(G) \sqsubseteq G$

Proof:

1. Proving that $p' \circ e'(F) = F$:

   - $p' \circ e'(\bot) = \bot$
   - $p' \circ e'(\{\langle 0, f \rangle\}) = p'(\{up\,(p \circ f \circ e)\}$
     $= up\,(p \circ e \circ f \circ p \circ e) = \langle 0, f \rangle$ [as $p \circ e = id_{D_1}$]

2. Proving that $e' \circ p'(G) \sqsubseteq G$:
   Similar to above but using $p \circ e \sqsubseteq id_{D_2}$ ∎

Now we have the machinery to define the action of the functor on the morphisms of the category $NSFP$. Define $\overline{P}((e,p)) = (e',p')$. It is easy to check that

- $\overline{P}((id_D, id_D)) = (id_D, id_D)$, for any continuous algebra $\langle D, \star \rangle$
- Let (e_1, p_1) be a linear embedding projection pair between $\langle D_1, \star_1 \rangle$ and $\langle D_2, \star_2 \rangle$. Let (e_2, p_2) be a linear embedding projection pair between $\langle D_2, \star_2 \rangle$ and $\langle D_3, \star_3 \rangle$. Then $(e_1 \circ e_2, p_2 \circ p_1)$ is a linear embedding projection pair between $\langle D_1, \star_1 \rangle$ and $\langle D_3, \star_3 \rangle$ and we have
  $\overline{P}(e_2 \circ e_1, p_1 \circ p_2) = \overline{P}((e_2, p_2)) \circ \overline{P}((e_1, p_1))$

The final lemma establishes that this functor is continuous and thus one can solve recursive domain equations using it.

Lemma 17. Let $\Delta = \langle D_m, \langle f_{mn}, f_{nm} \rangle \rangle$ be a chain of linear embedding projection pairs. Let $\langle D, \rho \rangle = \underrightarrow{Colim}\ \Delta$. Then, $\underrightarrow{Colim}\ \overline{P}(\Delta) \simeq \overline{P}(\underrightarrow{Colim}\ \Delta)$, where $\underrightarrow{Colim}\ \overline{P}(\Delta)$ means $\underrightarrow{Colim}\ \langle \overline{P}(D_m), \overline{P}(\langle f_{mn}, f_{nm} \rangle) \rangle$.

Proof: It suffices (lemma 2, [17][ch 4, page 11]) to check that $\bigsqcup \overline{P}(\rho_n) \circ \overline{P}(\rho_n^R) = id_{\overline{P}(D)}$. From the linearity and continuity of $\overline{P}(\rho_n)$ and $\overline{P}(\rho_n^R)$ it suffices to check $\bigsqcup \overline{P}(\rho_n) \circ \overline{P}(\rho_n^R)(s) = s$, for singleton sets of $P(D)$. When we look at the singleton sets, however, it is clear that we can mimic the standard verifications of this fact [21]. ∎

5 The Model and its Basic Properties

In this section we define the model and prove some basic properties of the model. The properties are essentially tools that show how one can define structures on the domain by induction using the iterates and also how one can use the projections onto the iterates to get a handle on the finite approximants to the elements. They are similar in spirit to the "index" calculations outlined in Wadsworth's discussion of D_∞, and to the "index" calculations outlined in the discussion of the lazy lambda-calculus [2].

5.1 What is a Model of the γ-calculus?

Before constructing the initial solution we sketch how this is used to provide a model of our subset of the γ-calculus. The recursive domain equation that we solve is

$$D = \overline{P}(D).$$

We solve this equation in the category NSFP.

From an algebraic point of view we have a cpo with three continuous operations, application $\diamond$, union $\star$ and product, $\times$. These operations obey the following laws:

1. $\bot \diamond x = \bot$
2. $(d \star e) \diamond x = (d \diamond x) \star (e \diamond x)$
3. $\times$ is associative
4. $\times$ is commutative
5. $d \times \bot = d \star \bot$
6. $(d \times e) \diamond x = ((d \diamond x) \times e) \star (d \times (e \diamond x))$
7. $d \times (e \star f) = (d \times e) \star (d \times f)$.

We now have enough structure to give semantics to the fragment of the language that we are considering. The following definition uses the familiar environment mechanism. The functions Gr and Fun map between $\overline{P}(D)$ and D.

- $[\![x]\!]\,\rho \;= \rho(x)$
- $[\![\lambda x.M]\!]\,\rho \;= Gr(d \mapsto [\![M]\!]\,\rho[x \mapsto d]\;)$
- $[\![\lambda\langle x_1, x_2\rangle M]\!]\,\rho \;=$
  $Gr(\star[(d_1 \mapsto Gr(d_2 \mapsto [\![M]\!]\,\rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\;)),$
  $(d_1 \mapsto Gr(d_2 \mapsto [\![M]\!]\,\rho[x_2 \mapsto d_1, x_1 \mapsto d_2]\;))])$
- $[\![MN]\!]\,\rho \;= [\![M]\!]\,\rho \diamond [\![N]\!]\,\rho$
- $[\![M|N]\!]\,\rho \;= [\![M]\!]\,\rho \times [\![N]\!]\,\rho$

5.2 Construction of the Initial Solution

Let $\langle D_1, \star_1\rangle$ and $\langle D_2, \star_2\rangle$ be continuous algebras. Let (e, p) be a linear embedding-projection pair. Define e' as follows:

- Define $e' : B(\overline{P}(D_1)) \to \overline{P}(D_2)$ by:
  - $e'\,(\bot) = \bot$
  - $e'\,(\langle 0, f\rangle) = up\,(e \circ dn\,(f) \circ p)$
  - $e'\{F_1 \ldots F_n\} = \{e'(F_1) \ldots e'(F_n)\}$

$e^\star$ is the unique continuous extension of e'.

Define $p' : B(\overline{P}(D_2)) \to \overline{P}(D_1)$ as follows:

- $p'\,(\bot) = \bot$
- $p'\,(\langle 0, g\rangle) = up\,(p \circ dn\,(g) \circ e)$

- $p'\{f_1 \dots f_n\} = \{p'(f_1) \dots p'(f_n)\}$

$p^\star$ is the unique continuous extension of p'.

Let D_0 be the the one point continuous algebra, and let $D_1 = \overline{P}(D)$. Let $i_0 : D_0 \rightarrow D_1$ be defined by $i_0(\bot_0) = \bot_1$. Let $j_0 : D_1 \rightarrow D_0$ be defined by $j_0(x) = \bot_0$. Define inductively:

- $D_{n+1} = \overline{P}(D_n)$
- $\langle i_{n+1}, j_{n+1}\rangle = \langle i_n^\star, j_n^\star\rangle$

Note that $(\forall n)$ $[\langle i_n, j_n\rangle$ is a linear ep pair]. Then $\langle D_n, j_n\rangle_{n\in\omega}$ is an inverse system of finite continuous algebras. Define, standardly, $\phi_{m,n} : D_n \rightarrow D_m$ by

- $\phi_{n,n} = 1_{D_n}$
- $\phi_{m+1,n} = \phi_{m,n} \circ j_m$, if $(n \leq m,\ n \neq m)$
- $\phi_{m,n+1} = i_n \circ \phi_{m,n}$, if $(m \leq n,\ n \neq m)$

Note that
$(\forall n, m)$ $[m \leq n \Rightarrow \langle \phi_{m,n}, \phi_{n,m}\rangle$ is a linear ep pair]. Identify the initial solution $D \subseteq \Pi_{n\in\omega} D_n$, as $D = \{\langle x_n\rangle_{n\in\omega} |\ x_n \in D_n \wedge j_n(x_{n+1}) = x_n\}$
ordered pointwise. Note that we have the linear ep pairs $\langle \phi_{m,\infty}, \phi_{\infty,m}\rangle$. Also, note that $\langle D, \star\rangle$ is a continuous algebra, where $\star$ is defined by
$\langle x_n\rangle_{n\in\omega} \star \langle y_n\rangle_{n\in\omega} = \langle x_n \star_n y_n\rangle_{n\in\omega}$
We write x_m for $\phi_{\infty,m}(x)$. Henceforth, we identify the element x of D_m with $\phi_{m,\infty}(x)$.

5.3 Index calculations

The proof of the following two lemmas is standaard and is omitted.

Lemma 18. Let $x \in D$. Then,

1. $x \in D_n \Rightarrow x_n = x$
2. $x \in D_n \Rightarrow i_n(x) = x$
3. $x \in D_{n+1} \Rightarrow j_n(x) \sqsubseteq x$

Lemma 19. Let $x \in D$. Then,

1. $(x_n)_m = x_{min(n,m)}$
2. $n \leq m \Rightarrow x_n \sqsubseteq x_m \sqsubseteq x$
3. $x = \bigsqcup_n x_n$
4. $x \in D_n \Rightarrow (\forall m \geq n)$ $[x_m = x]$
5. $\bot_n$ is the least element of D_n
6. $\bot_n = \bot$

5.4 Applicative behaviour

Define $App_n : (D_n \to D_n)_\perp \times D_n \to D_n$ as,

- $App_n(\perp, x) = \perp$
- $App_n(up(f), x) = f(x)$, where $f \in D_n \to D_n$

Define $App_n : D_{n+1} \times D_n \to D_n$ as the left linear extension of the above. More formally, let $s = \{f_1 \dots f_n\} \in D_{n+1}$. Then,
$App_n(s, x) = App_n(f_1, x) \star_n App_n(f_2, x) \dots \star_n App_n(f_n, x)$. It follows from the definitions that $App_n : D_{n+1} \times D_n \to D_n$ is a monotone (and hence continuous) function.

Lemma 20. Let $n \leq k$. Then,

1. $App_n(x_{n+1}, y_n) \sqsubseteq App_k(x_{k+1}, y_k)$
2. $App_n(x_{n+1}, y_n) = App_k((x_{n+1})_{k+1}, y_k)$
3. $App_n(x_{n+1}, y_n) = [App_k(x_{k+1}, (y_n)_k]_n$

Proof:

1. (Proof is by induction on k)
   Consider $k = n + 1$. Depending on the structure of x_{n+2}, there are two cases:

   - x_{n+2} is a singleton, i.e $x_{n+2} = \{f\}$, for some $f \in (D_n \to D_n)_\perp$. This splits up further into two cases.
     - $f = \perp$. Then $x_{n+1} = \{\perp\}$, and result follows.
     - $f = up(g)$, for some $g \in D_n \to D_n$. Then, note that $x_{n+1} = up(g')$, for some $g' \in D_{n-1} \to D_{n-1}$. So, we have

$$\begin{aligned} App_n(x_{n+1}, y_n) &= App_n(j_{n+1}(x_{n+2}), j_n(y_{n+1})) \\ &= App_n(up(j_n \circ dn(x_{n+2}) \circ i_n), j_n(y_{n+1})) \\ &= j_n \circ dn(x_{n+2}) \circ i_n \circ j_n(y_{n+1}) \\ &\sqsubseteq j_n \circ dn(x_{n+2})(y_{n+1}) \\ &\sqsubseteq dn(x_{n+2})(y_{n+1}) \\ &= App_{n+1}(x_{n+2}, y_{n+1}) \end{aligned}$$

   - $x_{n+2} = \{f_1 \dots f_m\}$. Note that
     $j_{n+1}(x_{n+2}) = \{j_{n+1}(f_1) \dots j_{n+1}(f_m)\} = \star_{n+1}\{j_{n+1}(f_1) \dots j_{n+1}(f_m)\}$.

$$\begin{aligned} App_n(x_{n+1}, y_n) &= App_n(j_{n+1}(x_{n+2}), j_n(y_{n+1})) \\ &= App_n(\star_{n+1}\{j_{n+1}(f_1) \dots j_{n+1}(f_m)\}, j_n(y_{n+1})) \\ &= \star_n\{App_n(j_{n+1}(f_1), j_n(y_{n+1})), \dots App_n(j_{n+1}(f_m), j_n(y_{n+1}))\} \end{aligned}$$

   But, from the preceding case, $App_n((f_i)_{n+1}, y_n) \sqsubseteq App_n(f_i, y_{n+1})$. Result now follows from monotonicity of $\star_n$

2. (Proof is by induction on k)
Assume result for k. Consider $k+1$. Depending on the structure of $(x_{n+1})_{k+2}$, there are two cases:

   - $(x_{n+1})_{k+2}$ is a singleton, i.e $(x_{n+1})_{k+2} = \{f\}$, for some $f \in (D_{k+1} \rightarrow D_{k+1})_{\perp}$. This splits up further into two cases.
     - $f = \perp$. Then $x_{n+1} = \{\perp\}$, and result follows.
     - $f = up(g)$, for some $g \in D_{k+1} \rightarrow D_{k+1}$. We have

$$\begin{aligned} App_{k+1}((x_{n+1})_{k+2}, y_{k+1}) &= App_{k+1}(i_{k+1}(x_{n+1})_{k+1}, y_{k+1}) \\ &= App_{k+1}(up(i_k \circ dn(x_{n+1})_{k+1} \circ j_k), y_{k+1}) \\ &= i_k \circ dn(x_{n+1})_{k+1} \circ j_k(y_{k+1}) \\ &= i_k \circ dn(x_{n+1})_{k+1}(y_k) \\ &= i_k(App_k((x_{n+1})_{k+1}, y_k)) \\ &= i_k(App_n(x_{n+1}, y_n)) \\ &= App_n(x_{n+1}, y_n) \end{aligned}$$

   - $(x_{n+1})_{k+2} = \{(f_1)_{k+2} \ldots (f_m)_{k+2}\}$. We have
$App_{k+1}((x_{n+1})_{k+2}, y_{k+1}) = \star_{k+1}\{App_{k+1}((f_1)_{k+2}, y_{k+1}), \ldots App_{k+1}((f_m)_{k+2}, y_{k+1})\}$.

     From previous cases, we have
$(\forall 1 \leq i \leq m)\ [App_{k+1}((f_1)_{k+2}, y_{k+1}) = App_n((f_i)_{n+1}, y_n)$
So, we have

$$\begin{aligned} App_{k+1}((x_{n+1})_{k+2}, y_{k+1}) &= \star_{k+1}\{App_{k+1}((f_1)_{k+2}, y_{k+1}), \ldots App_{k+1}((f_m)_{k+2}, y_{k+1})\} \\ &= \star_{k+1}\{App_n((f_1)_{n+1}, y_n), \ldots App_n((f_m)_{n+1}, y_n)\} \\ &= \star_n\{App_n((f_1)_{n+1}, y_n), \ldots App_n((f_m)_{n+1}, y_n)\} \\ &= App_n(\{(f_1)_{n+1} \ldots (f_m)_{n+1}\}, y_n) \\ &= App_n(x_{n+1}, y_n) \qquad \blacksquare \end{aligned}$$

3. (Proof by induction on k)
Assume result for k. Consider $k+1$. Depending on the structure of x_{k+2}, there are two cases:

   - x_{k+2} is a singleton, i.e $x_{k+2} = \{f\}$, for some $f \in (D_{k+1} \rightarrow D_{k+1})_{\perp}$. This splits up further into two cases.
     - $f = \perp$. Then $x_{n+1} = \{\perp\}$, and result follows.
     - $f = up(g)$, for some $g \in D_{k+1} \rightarrow D_{k+1}$. We have

$$\begin{aligned} [App_{k+1}(x_{k+2}, (y_n)_{k+1})]_n &= \Phi_{k+1,n}[App_{k+1}(x_{k+2}, (y_n)_{k+1})] \\ &= \Phi_{k,n} \circ j_k[App_{k+1}(x_{k+2}, (y_n)_{k+1})] \end{aligned}$$

$$
\begin{aligned}
&= \Phi_{k,n} \circ j_k \circ dn(x_{k+2}) \circ i_k((y_n)_k) \\
&= \Phi_{k,n}[App_k(j_{k+1}(x_{k+2}), (y_n)_k)] \\
&= [App_k(j_{k+1}(x_{k+2}), (y_n)_k)]_n \\
&= [App_k(x_{k+1}, (y_n)_k)]_n \\
&= App_n(x_{n+1}, y_n)
\end{aligned}
$$

- $(x_{n+1})_{k+2} = \{(f_1)_{k+2} \ldots (f_m)_{k+2}\}$.

$$
\begin{aligned}
[App_{k+1}((x_{k+2}), (y_n)_{k+1})]_n &= \Phi_{k+1,n}[App_{k+1}(x_{k+2}, (y_n)_{k+1})] \\
&= [\star_{k+1}\{App_{k+1}((f_1)_{k+2}, y_{k+1}) \ldots App_{k+1}((f_m)_{k+2}, (y_n)_{k+1})\}], \\
&= \star_n\{[App_{k+1}((f_1)_{k+2}, y_{k+1})]_n \ldots [App_{k+1}((f_m)_{k+2}, (y_n)_{k+1})]_n \\
&= \star_n[\{App_n((f_1)_{n+1}, y_n) \ldots App_n((f_m)_{n+1}, y_n)\}] \\
&= App_n(\{(f_1)_{n+1}, \ldots (f_m)_{n+1}\}, y_n) \\
&= App_n(x_{n+1}, y_n) \qquad \blacksquare
\end{aligned}
$$

Define $\diamond : D \times D \to D$ by $x \diamond y = \bigsqcup_{n\in\omega}[Ap_n(x_{n+1}, y_n)]$. Lemma 20.1 proves that the terms whose $\bigsqcup$ is being taken do form a chain.

Lemma 21. (Coherence) Let $x \in D_{n+1}, y \in D_n$. Then,
$x \diamond y = Ap_n(x_{n+1}, y_n)$

Proof:

$$
\begin{aligned}
x \diamond y &= x_{n+1} \diamond y_n \\
&= \textstyle\bigsqcup_{i\in\omega}[Ap_i((x_{n+1})_{i+1}, (y_n)_i)] \text{ [from 20 .2]} \\
&= \textstyle\bigsqcup_{i\leq n}[Ap_i((x_{n+1})_{i+1}, (y_n)_i)] \\
&= Ap_n(x_{n+1}, y_n) \qquad \blacksquare
\end{aligned}
$$

It is easy to check that $\diamond$ is a continuous function.

Lemma 22. Let $x, y \in D$. Then,

1. $x_{n+1} \diamond y = x_{n+1} \diamond y_n = [x \diamond y_n]_n$
2. $x_1 \diamond y = \perp$
3. $x_0 \diamond y = \perp$

Proof: Proofs of 2, 3 are omitted. 1 is proved below.

$$
\begin{aligned}
x_{n+1} \diamond y &= \textstyle\bigsqcup_{i\in\omega}[App_i((x_{n+1})_{i+1}, y_i)] \text{ use 20 2} \\
&= \textstyle\bigsqcup_{i\leq n}[App_i((x_{n+1})_{i+1}, y_i)] \text{ use 20 1} \\
&= App_n(x_{n+1}, y_n)
\end{aligned}
$$

$$\begin{aligned}[x \diamond y_n]_n &= [\textstyle\bigsqcup_{i\in\omega}[App_i((x_{n+1})_{i+1}, y_i)]]_n \\ &= \textstyle\bigsqcup_{i\in\omega}[App_i((x_{n+1})_{i+1}, y_i)]_n \text{ from } 20\ 3 \\ &= \textstyle\bigsqcup_{i\leq n}[App_i((x_{n+1})_{i+1}, y_i)]_n \\ &= \textstyle\bigsqcup_{i\leq n}[App_i(x_{i+1}, y_i)]_n \\ &= \textstyle\bigsqcup_{i\leq n}[App_i(x_{i+1}, y_i)]_n \text{ from } 20\ 1 \\ &= App_n(x_{n+1}, y_n) \\ &= x_{n+1} \diamond y_n \qquad \blacksquare\end{aligned}$$

5.5 Isomorphism between D and $\overline{P}(D)$

$\langle D, \star\rangle$ is a continuous algebra. Define $App' : (D \to D)_\perp \times D \to D$ as,

- $App'(\perp, x) = \perp$
- $App'(up(f), x) = f(x)$, where $f \in D \to D$

Define $App : P(D) \times D \to D$ as the left linear extension of App'. Formally, let $s = \{f_1 \dots f_n\} \in D_{n+1}$. Then, $App(s, x) = App(f_1, x) \star App(f_2, x) \dots \star App(f_n, x)$. Recall that $P(D)$ was defined as:

- $|P(D)| = \{s | s \in P_{fin}(B((D \to D)_\perp))$
- $s_1 \sqsubseteq s_2 \Leftrightarrow$
  1. $\perp \in s_2 \Rightarrow \perp \in s_1$
  2. $s_1 \neq \{\perp\} \Rightarrow s_2 \neq \{\perp\}$
  3. $(\forall x \in D)[App(s_1, x) \sqsubseteq App(s_2, x)]$

It follows from the definitions that $App : P(D) \times D \to D$ is a monotone function. Note that $B(\overline{P}(D)) = P(D)$. Extend App continuously to the whole of $\overline{P}(D)$.

Define $[rep()]_n : P(D) \to D_{n+1}$ as follows. Let $s \in P(D)$.

- s is a singleton. This splits up into two cases.
  - $s = \{\perp\}$. Define $[rep(s)]_n = \perp_{n+1}$.
  - $s = \{up(f)\}$, where $f \in B(D \to D)$. Define $[rep(s)]_n = up(\lambda y \in D_n.[App(s, y)]_n)$
- $s = \{f_1 \dots f_m\}$, where $f_i \in B((D \to D)_\perp)$. Define $[rep(s)]_n = \star\{[rep(f_1)]_n \dots [rep(f_m)]_n\}$.

Lemma 23. $(\forall y \in D_n)\ [[rep(s)]_n \diamond y = [App(s, y)]_n]$

Proof: Proof is by cases depending on the structure of s. Let $y \in D_n$

- s is a singleton. This splits up into two cases.
  - $s = \{\bot\}$. Result is immediate.
  - $s = \{up(f)\}$, where $f \in B(D \to D)$. Then

$$\begin{aligned}[rep(s)]_n \diamond y &= Ap_n[up(\lambda y \in D_n.[App(s,y)]_n), y] \\ &= [App(s,y)]_n\end{aligned}$$

- $s = \{f_1 \ldots f_m\}$, where $f_i \in B((D \to D)_\bot)$.

$$\begin{aligned}[rep(s)]_n \diamond y &= \star\{[rep(f_1)]_n \ldots [rep(f_m)]_n\} \diamond y \\ &= \star\{[rep(f_1)]_n \diamond y \ldots [rep(f_m)]_n \diamond y\} \\ &= \star\{[App(f_1,y)]_n \ldots [App(f_m,y)]_n\} \\ &= [\star\{App(f_1,y) \ldots App(f_m,y)\}]_n \\ &= [App(\{f_1, \ldots f_m\}, y)]_n \quad \blacksquare\end{aligned}$$

Lemma 24. $s_1 \sqsubseteq s_2 \Rightarrow [rep(s_1)]_n \sqsubseteq [rep(s_2)]_n$

Proof: Let $s_1 = \{f_1 \ldots f_m\}$, $s_2 = \{g_1 \ldots g_n\}$.

-

$$\begin{aligned}\bot \in [rep(s_2)]_n &\Rightarrow \bot \in s_2 \\ &\Rightarrow \bot \in s_1 \\ &\Rightarrow \bot \in [rep(s_1)]_n\end{aligned}$$

- $[rep(s_1)]_n \neq \{\bot\} \Rightarrow [rep(s_2)]_n \neq \{\bot\}$ is proved similarly.
- Let $y \in D_n$. Then,

$$\begin{aligned}[rep(s_1)]_n \diamond y &= [App(s_1,y)]_n \\ &\sqsubseteq [App(s_2,y)]_n \\ &= [rep(s_2)]_n \diamond y \quad \blacksquare\end{aligned}$$

As a corollary, we get that $[rep()]_n$ is well-defined on the equivalence classes of the preorder $P(D)$, and hence on $B(\overline{P}(D))$. Extend $[rep()]_n$ continuously to the whole of $\overline{P}(D)$. Continuity of the functions involved means that lemma 23 holds for the continuous extensions so defined.

Lemma 25. $j_{n+1}([rep(s)]_{n+1}) = [rep(s)]_n$

Proof: From linearity of j_n, $[rep()]_n$, suffices to prove the result for singletons i.e s is of form, $s = \{f\}$, where $f \in B((D \to D)_\bot)$. This splits up into the following two cases:

- $f = \bot$. Result follows immediately.
- $f = up(g)$, for some $g \in D \to D$.

$$\begin{array}{rcl} j_{n+1}([rep(s)]_{n+1}) & = & j_{n+1}(\lambda y \in D_{n+1}.[App(s,y)]_{n+1}) \\ & = & up(\lambda y \in D_n.j_n \circ [\lambda y \in D_{n+1}.[App(s,y)]_{n+1}] \circ i_n(y)) \\ & = & up(\lambda y \in D_n.j_n([App(s,i_n(y))]_{n+1}) \\ & = & up(\lambda y \in D_n.[App(s,i_n(y))]_n)\ [y \in D_n \Rightarrow i_n(y) = y] \\ & = & up(\lambda y \in D_n.[App(s,y)]_n) \\ & = & [rep(s)]_n \quad \blacksquare \end{array}$$

Define $rep() : \overline{P}(D) \to D$ as
$rep(s) = \bigsqcup_{n\in\omega}[rep(s)]_n$. The above lemma shows that the lub is well-defined. Furthermore, we have the following lemma.

Lemma 26. Let $y \in D$. Then, $rep(s) \diamond y = App(s,y)$

Proof: Let $y \in D$.

$$\begin{array}{rcl} rep(s) \diamond y & = & \bigsqcup_{i\in\omega} App_i((rep(s))_{i+1}, y_i) \\ & = & \bigsqcup_{i\in\omega}[rep(s) \diamond y_i]_i \\ & = & \bigsqcup_{i\in\omega}[\bigsqcup_{n\in\omega}[rep(s)]_n \diamond y_i]_i \\ & = & \bigsqcup_{i,n\in\omega}[[rep(s)]_n \diamond y_i]_i \\ & = & \bigsqcup_{j\in\omega}[[rep(s)]_j \diamond y_j]_j \\ & = & \bigsqcup_{j\in\omega}[App(s,y_j)]_j \\ & = & \bigsqcup_{n\in\omega}\bigsqcup_{i\in\omega}[App(s,y_i)]_n \\ & = & \bigsqcup_{n\in\omega}[App(s,y)]_n \\ & = & App(s,y) \quad \blacksquare \end{array}$$

Define $Fun : D \to \overline{P}(D)$ as follows. We first define it on $B(D)$. Note that $B(D) = \bigcup_{n\in\omega} D_n$. Let $s \in B(D)$. This definition is done by cases.

- $s = \{\bot\}$. Define $Fun(s) = \{\bot\}$.
- $s = \{up(g)\}$, for some $g \in D_n \to D_n$. Define $Fun(s) = up(x \mapsto s \diamond x)$.
- $s = \{f_1 \dots f_m\}$. Define $Fun(s) = \{Fun(f_1), \dots, Fun(f_m)\}$

Lemma 27. $App(Fun(s), y) = s \diamond y$

Proof: From linearity of App and Fun suffices to prove the result for singletons s. For singletons, result follows directly from the definition. $\blacksquare$

Lemma 28. *Fun* is monotone.

Proof: Let s_1, $s_2 \in D_n$ and $s_1 \sqsubseteq s_2$. Let $s_1 = \{f_1 \ldots f_m\}$. Let $s_2 = \{g_1 \ldots g_n\}$.

-
$$\begin{aligned} \bot \in Fun(s_2) &\Rightarrow \bot \in s_2 \\ &\Rightarrow \bot \in s_1 \\ &\Rightarrow \bot \in Fun(s_1) \end{aligned}$$

- $Fun(s_1) \neq \{\bot\} \Rightarrow Fun(s_2) \neq \{\bot\}$ is proved similarly.
- Let $y \in D$. Then,

$$\begin{aligned} App(Fun(s_1), y) &= s_1 \diamond y \\ &\sqsubseteq s_2 \diamond y \\ &= App(Fun(s_2), y) \quad \blacksquare \end{aligned}$$

Extend Fun to a continuous function in $D \to \overline{P}(D)$. From continuity of all functions involved we get $App(Fun(s), y) = s \diamond y$

Lemma 29. $Fun \circ rep = id_{\overline{P}(D)}$

Proof: Since Fun, rep are linear and continuous, it suffices to check the $Fun \circ rep(s) = s$, for $s = \{f\}$, where $f \in B((D \to D)_\bot)$. We have the following two cases.

- $f = \bot$. Result follows from definitions.
- $f = up(g)$, for some $g \in B(D \to D)$.

$$\begin{aligned} Fun(rep(s)) &= \{up(x \mapsto rep(s) \diamond x)\} \\ &= \{up(x \mapsto App(s, x))\} \\ &= \{up(x \mapsto g(x))\} \\ &= s \quad \blacksquare \end{aligned}$$

Lemma 30. $rep \circ Fun = id_D$

Proof: Since Fun, rep are linear and continuous, it suffices to check the $rep \circ Fun(s) = s$, for $s = \{f\}$, $s \in D_{n+1}$, for some n. We have the following two cases.

- $f = \bot$. Result follows from definitions.

- $f = up(g)$, for some $g \in D_n \to D_n$).

$$\begin{aligned}
rep(Fun(s)) &= rep(up(x \mapsto s \diamond x)) \\
&= \bigsqcup_{n\in\omega}[[rep(up(x \mapsto s \diamond x))]_n] \\
&= \bigsqcup_{n\in\omega}[up(\lambda y \in D_n.[App(up(x \mapsto s \diamond x), y)]_n)] \\
&= \bigsqcup_{n\in\omega}[up(\lambda y \in D_n.[s \diamond y]_n)] \\
&= \bigsqcup_{n\in\omega}[up(\lambda y \in D_n.[s_{n+1} \diamond y]))] \\
&= \bigsqcup_{n\in\omega} s_{n+1} \\
&= s \quad \blacksquare
\end{aligned}$$

Lemma 31. Let $s_1, s_2 \in D_n$, $1 \leq n$. Then, the following are equivalent:

- $(s_1)_1 \sqsubseteq (s_2)_1$
- $[\bot \in s_2 \Rightarrow \bot \in s_1] \wedge [s_1 \neq \{\bot\} \Rightarrow s_2 \neq \{\bot\}]$

Proof: By induction on n, where $1 \leq n$. $\blacksquare$

Lemma 32. (Conditional Strong extensionality) Let $d, e \in D$. Then $d \sqsubseteq e \Leftrightarrow d_1 \sqsubseteq e_1 \wedge (\forall x \in D)\ [d \diamond x \sqsubseteq e \diamond x]$

Proof: Forward implication is immediate. For the reverse implication, note that $(\forall x \in D_n) d_{n+1} \diamond x = [d \diamond x]_n \sqsubseteq [e \diamond x]_n = e_{n+1} \diamond x$. This along with the previous lemma shows that $d_n \sqsubseteq e_n$. $\blacksquare$

6 Product structure on D

The | constructor is modelled by a continuous function $\times : D \times D \to D$. In this section we use $[]_1$ as shorthand for the projection map onto D_1. The following lemma is used implicitly in the following proofs.

Lemma 33. Let $f \in D_s$. Then,

- $\bot \notin f \Leftrightarrow \{\lambda x.\bot_0\} \sqsubseteq [f]_1$
- $\bot = f \Leftrightarrow \bot = [f]_1$

Proof: Induction on s. $\blacksquare$

Let D_s be the iterates in the solution of the recursive domain equation $D \simeq \overline{P}(D)$. Define a family of functions $\times_{(s,t)} : D_s \times D_t \to D_{(s+t)}$, by induction on $s + t$ as follows. Let $f \in D_s$, $g \in D_t$.

- $(s + t = 0)$. $f \times_{(0,0)} g = \bot_{(0,0)}$

- $(s+t \neq 0)$. Assume f, g are singleton sets. Then, define by cases on $[f]_1$, $[g]_1$ as follows.
  - $\{\lambda x.\bot_0\} = [f]_1$, $\bot_0 = [g]_1$. Then,
    $f \times_{(s,t)} g = \bot_{(s+t)} \star up[x \in D_{s+t-1} \mapsto (f \diamond x) \times_{(s-1,t)} \bot_t]$
  - $\{\lambda x.\bot_0\} = [g]_1$, $\bot_0 = [f]_1$. Then,
    $f \times_{(s,t)} g = \bot_{(s+t)} \star [up[x \in D_{s+t-1} \mapsto \bot_s \times_{(s,t-1)} (g \diamond x)]$
  - $\{\lambda x.\bot_0\} = [f]_1$, $\{\lambda x.\bot_0\} = [g]_1$,. Then,
    $f \times_{(s,t)} g = up[x \in D_{s+t-1} \mapsto ([f \times_{(s,t-1)} (g \diamond x)] \star [(f \diamond x) \times_{(s-1,t)} g])$
  - $\bot_0 = [f]_1$, $\bot_0 = [g]_1$. Then,
    $f \times_{(s,t)} g = \bot_{(s+t)}$.
- $(s+t \neq 0)$. $f = \{[f]_1 \dots f_m\}$, $g = \{g_1 \dots g_n\}$. Then,
  $f \times_{(s,t)} g = \star[f_i \times_{(s,t)} g_j | 1 \leq i \leq m,\ 1 \leq j \leq n]$

We first show that $\times_{(s,t)}$ is well-defined and monotone in both its arguments.

Lemma 34. $(\forall s,\ t \in \omega)$, the following holds. Let $f \in D_s$, $g \in D_t$. Then,

- $[f \times_{(s,t)} g] \in D_{(s+t)}$
- Let $f' \in D_s$, $g' \in D_t$, $f \sqsubseteq f'$, $g \sqsubseteq g'$. Then, $f \times_{(s,t)} g \sqsubseteq f' \times_{(s,t)} g'$.

Proof: The proof of both parts is by induction on $s+t$. The base case $s+t=0$, is checked easily. For the induction step, assume result for $s+t \leq n$. Consider $s+t = n+1$. Let $f = \{f_1 \dots f_m\}$, $g = \{g_1 \dots g_n\}$, such that $f \in D_s$, $g \in D_t$.

1. (Proving that $f \times_{(s,t)} g \in D_{(s+t)}$).
   Thus we need to prove the monotonicity of $f \times_{(s,t)} g$, as a function from $D_{(s+t-1)}$ to $D_{(s+t-1)}$.
   - $(\{\lambda x.\bot_0\} \sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \sqsubseteq [g]_1)$. Then,
   $$\begin{aligned}(f \times_{(s,t)} g) \diamond x &= \star[f_i \times_{(s,t)} g_j | 1 \leq i \leq m,\ 1 \leq j \leq n] \diamond x \\ &= \star[(f_i \times_{(s,t)} g_j) \diamond x | 1 \leq i \leq m,\ 1 \leq j \leq n] \\ &= \star[((f_i \diamond x) \times_{(s-1,t)} g_j) \star (f_i \times_{(s,t-1)} (g_j \diamond x)) | 1 \leq i \leq m,\ 1 \leq j \leq n]\end{aligned}$$
   The result follows from induction hypothesis and the monotonicity of $\star$.
   - $(\{\lambda x.\bot_0\} \sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \not\sqsubseteq [g]_1)$. Let $g_1 = \bot$. Then,
   $$\begin{aligned}(f \times_{(s,t)} g) \diamond x &= \star[f_i \times_{(s,t)} g_j | 1 \leq i \leq m,\ 2 \leq j \leq n] \diamond x \\ &\quad \star \bot \star\ [(\star[f_i \times_{(s,t)} \bot]) | 1 \leq i \leq m] \diamond x] \\ &= [((f_i \diamond x) \times_{(s-1,t)} g_j) \star (f_i \times_{(s,t-1)} (g_j \diamond x)) | 1 \leq i \leq m,\ 2 \leq j \leq n] \\ &\quad \star \bot\ \star\ [((f_i \diamond x) \times_{(s-1,t)} \bot) |\ 1 \leq i \leq m]\end{aligned}$$
   The result follows from induction hypothesis and the monotonicity of $\star$.

- $(\{\lambda x.\bot_0\} \not\sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \sqsubseteq [g]_1)$. Proof is similar to the preceding case.
- $(\{\lambda x.\bot_0\} \not\sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \not\sqsubseteq [g]_1)$. Proof is similar to the previous cases.

2. (Proving monotonicity of $(\times_{(s,t)})$
We prove the monotonicity of $\times_{(s,t)}$ in its left argument. Proof of monotonicity of $\times_{(s,t)}$ in its right argument is similar, and is omitted. Let $h = \{h_1 \dots h_l\}$, $h \in D_s$, $f \sqsubseteq h$. Then,

- Let $\bot \in h \times_{(s,t)} g$. Then,

$$\begin{aligned} \bot \in h \times_{(s,t)} g &\Rightarrow \bot \in h \vee \bot \in g \\ &\Rightarrow \bot \in f \vee \bot \in g \\ &\Rightarrow \bot \in f \times_{(s,t)} g \end{aligned}$$

- $\bot \neq f \times_{(s,t)} g \Rightarrow \bot \neq h \times_{(s,t)} g$ is proved similarly.

So, proof is complete if we check
$(\forall x \in D_{s+t})\ [(f \times_{(s,t)} g) \diamond x \sqsubseteq (h \times_{(s,t)} g) \diamond x]$. This splits up into the following cases.

- $(\{\lambda x.\bot_0\} \sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \sqsubseteq [g]_1)$. So, $\{\lambda x.\bot_0\} \sqsubseteq [h]_1$, as $f \sqsubseteq h$. Then,

$$\begin{aligned} (f \times_{(s,t)} g) \diamond x &= (\star[f_i \times_{(s,t)} g_j | 1 \leq i \leq m,\ 1 \leq j \leq n]) \diamond x \\ &= \star[(f_i \times_{(s,t)} g_j) \diamond x | 1 \leq i \leq m,\ 1 \leq j \leq n] \\ &= \star[((f_i \diamond x) \times_{(s-1,t)} g_j) \star (f_i \times_{(s,t-1)} (g_j \diamond x)) | 1 \leq i \leq m,\ 1 \leq j \leq n] \\ &= [(f \diamond x) \times_{(s-1,t)} g_j] \star [f \times_{(s-1,t)} (g_j \diamond x|\ 1 \leq j \leq n] \end{aligned}$$

Similarly,

$$(h \times_{(s,t)} g) \diamond x = [(h \diamond x) \times_{(s-1,t)} g_j] \star [h \times_{(s-1,t)} (g_j \diamond x)|\ 1 \leq j \leq n]$$

The result now follows from induction hypothesis, monotonicity of $\star$ and noting that $f \diamond x \sqsubseteq h \diamond x$

- $(\{\lambda x.\bot_0\} \not\sqsubseteq [f]_1,\ \{\lambda x.\bot_0\} \sqsubseteq [g]_1,\ \{\lambda x.\bot_0\} \sqsubseteq [h]_1)$. Then, $\bot \in f$. Without loss of generality, assume that $f_1 = \bot$

$$\begin{aligned} (f \times_{(s,t)} g) \diamond x &= (\star[f_i \times_{(s,t)} g_j | 1 \leq i \leq m,\ 1 \leq j \leq n]) \diamond x \\ &= \star[(f_i \times_{(s,t)} g_j) \diamond x | 1 \leq i \leq m,\ 1 \leq j \leq n] \\ &= \star[((f_i \diamond x) \times_{(s-1,t)} g_j) \star (f_i \times_{(s,t-1)} (g_j \diamond x)) | 2 \leq i \leq m,\ 1 \leq j \leq n] \\ &\quad \star\bot\ \star [\bot \times_{(s,t-1)} (g_j \diamond x))|\ 1 \leq j \leq n] \end{aligned}$$

Furthermore,

$$\begin{aligned} (h \times_{(s,t)} g) \diamond x &= \star[h_k \times_{(s,t)} g_j | 1 \leq k \leq l,\ 1 \leq j \leq n] \diamond x \\ &= \star[(h_k \times_{(s,t)} g_j) \diamond x | 1 \leq k \leq l,\ 1 \leq j \leq n] \\ &= \star[(h_k \times_{(s,t)} g_j) \diamond x | 1 \leq k \leq l,\ 1 \leq j \leq n] \\ &= \star[((h_k \diamond x) \times_{(s-1,t)} g_j) \star (h_k \times_{(s,t-1)} (g_j \diamond x)) | 1 \leq k \leq l,\ 1 \leq j \leq n] \\ &= \star[((h \diamond x) \times_{(s-1,t)} g_j) \star (h \times_{(s,t-1)} (g_j \diamond x))|\ 1 \leq j \leq n] \end{aligned}$$

$[(f_i \times_{(s,t-1)} (g_j \diamond x))|2 \leq i \leq m] \star [\perp \times_{(s,t-1)} (g_j \diamond x)] = f \times_{(s,t-1)} (g_j \diamond x)$.
From induction hypothesis, $f \times_{(s,t-1)} (g_j \diamond x) \sqsubseteq h \times_{(s,t-1)} (g_j \diamond x)$. Also,

$$\begin{aligned}
\perp \star [(f_i \diamond x) \times_{(s-1,t)} g_j|2 \leq i \leq m] &\sqsubseteq (\perp \times_{(s-1,t)} g_j) \star [(f_i \diamond x) \times_{(s-1,t)} g_j|2 \leq i \leq m] \\
&= (f \diamond x) \times_{(s-1,t)} g_j \\
&\sqsubseteq (h \diamond x) \times_{(s-1,t)} g_j
\end{aligned}$$

Hence, the result.

- $(\{\lambda x.\perp_0\} \not\sqsubseteq [f]_1,\ \{\lambda x.\perp_0\} \sqsubseteq [g]_1,\ \{\lambda x.\perp_0\} \not\sqsubseteq [h]_1)$. As in previous case,

$$\begin{aligned}
(f \times_{(s,t)} g) \diamond x = &\star[((f_i \diamond x) \times_{(s-1,t)} g_j) \star (f_i \times_{(s,t-1)} (g_j \diamond x))|2 \leq i \leq m,\ 1 \leq j \leq n] \\
&\star\perp\ \star[\perp \times_{(s,t-1)} (g_j \diamond x))|\ 1 \leq j \leq n]
\end{aligned}$$

Also by a similar argument,

$$\begin{aligned}
(h \times_{(s,t)} g) \diamond x = &\star[((h_k \diamond x) \times_{(s-1,t)} g_j) \star (h_k \times_{(s,t-1)} (g_j \diamond x))|2 \leq k \leq l,\ 1 \leq j \leq n] \\
&\star\perp\ \star\ [\perp \times_{(s,t-1)} (g_j \diamond x))|\ 1 \leq j \leq n]
\end{aligned}$$

As in previous case, we have

$$\begin{aligned}
[(f_i \times_{(s,t-1)} (g_j \diamond x))|2 \leq i \leq m] &\star [\perp \times_{(s,t-1)} (g_j \diamond x)] \\
&= f \times_{(s,t-1)} (g_j \diamond x) \\
&\sqsubseteq h \times_{(s,t-1)} (g_j \diamond x) \\
&= [(h_k \times_{(s,t-1)} (g_j \diamond x))|2 \leq k \leq l] \star [\perp \times_{(s,t-1)} (g_j \diamond x)]
\end{aligned}$$

Also,

$$\begin{aligned}
\perp \star [(f_i \diamond x) \times_{(s-1,t)} g_j|2 \leq i \leq m] &\sqsubseteq (\perp \times_{(s-1,t)} g_j) \star [(f_i \diamond x) \times_{(s-1,t)} g_j|2 \leq i \leq m] \\
&= (f \diamond x) \times_{(s-1,t)} g_j \\
&\sqsubseteq [((h_k \diamond x) \times_{(s-1,t)} g_j)|2 \leq k \leq l]
\end{aligned}$$

The last step follows because

$$\begin{aligned}
f \diamond x &\sqsubseteq h \diamond x \\
&\sqsubseteq [h_k \diamond x|\ 2 \leq k \leq l]
\end{aligned}$$

So, from idempotence and monotonicity of $\star$, and induction hypothesis $\perp \star [(f_i \diamond x) \times_{(s-1,t)} g_j|2 \leq i \leq m] \sqsubseteq [((h_k \diamond x) \times_{(s-1,t)} g_j)|2 \leq k \leq l] \star \perp$.
Hence, the result.

The proofs of the other cases are similar and are omitted. ∎

The following lemma shows that the subscripts can be dropped from $\times_{(s,t)}$.

Lemma 35. (Well-definedness of $\times_{(s,t)}$)
Let $f \in D_s$, $g \in D_t$. Then $i_s(f) \times_{(s+1,t)} g = i_{(s+t)}(f \times_{(s,t)} g)$.

Proof: We prove that $i_s(f) \times_{(s+1,t)} g = i_{(s+t)}[f \times_{(s,t)} g]$, from which result follows. Proof is by induction on $s+t$. The base case $s+t=0$ follows immediately. Assume result for $s+t \leq b$. Consider the case $s+t = n+1$. From the linearity of i_s, $i_{(s+t)}$ and the bilinearity of $\times_{(s,t)}$, $\times_{(s+1,t)}$, it suffices to prove the result for singletons f, g. Thus we have the following cases.

- $\{\lambda x.\bot\} = [f]_1$, $\{\lambda x.\bot\} = [g]_1$. Then,
  $i_{(s+t)}(f \times_{(s,t)} g) = i_{(s+t)}(up(x \mapsto [((f \diamond x) \times_{(s-1,t)} g) \star (f \times_{(s-1,t)} (g \diamond x))]$. The first two conditions of definition 12 are verified easily. So, we just need to check that
  $(\forall x \in D_{(s+t)}[(i_{(s+t)}(f \times_{(s,t)} g)) \diamond x = (i_s(f) \times_{(s+1,t)} g) \diamond x]$.
  Let $x \in D_{(s+t)}$. Let $y = j_{(s+t-1)}(x)$. Then,

$$\begin{aligned}(i_{(s+t)}(f \times_{(s,t)} g)) \diamond x &= i_{(s+t-1)}[((f \diamond y) \times_{(s-1,t)}) \star (f \times_{(s,t-1)} (g \diamond y))] \\ &= i_{(s+t-1)}[((f \diamond y) \times_{(s-1,t)})] \star\ i_{(s+t-1)}[(f \times_{(s,t-1)} (g \diamond y))] \\ &= (i_{s-1}(f \diamond y) \times_{(s-1,t)}) \star\ (i_s(f) \times_{(s,t-1)} (g \diamond y))\end{aligned}$$

  The last step follows from the induction hypothesis. Since $f \in D-s$,

$$\begin{aligned}i_{s-1}(f \diamond y) &= i_{s-1}(f \diamond y_{(s-1)}) \quad [s \leq s+t-1 \Rightarrow y_{(s-1)} = x_{(s-1)}] \\ &= i_{s-1}(f \diamond x_{(s-1)}) \\ &= i_{s-1}(f \diamond x) \\ &= f \diamond x \qquad [i_s(f) = f] \\ &= i_s(f) \diamond x\end{aligned}$$

  Hence the result.

- $\{\lambda x.\bot\} = [g]_1$, $\bot = [f]_1$. Then,
  $f \times_{(s,t)} g = \bot\ \star\ up(x \mapsto\ [\bot \times_{(s,t-1)}])(g \diamond x)$. So,
  $i_{(s+t)}(f \times_{(s,t)} g) = i_{(s+t)}(\bot) \star\ i_{(s+t)}(up(x \mapsto\ (\bot \times_{(s,t-1)} (g \diamond x))))$.
  The first two conditions in the definition 12 are checked easily.
  Let $x \in D_{(s+t)}$, $y = j_{(s+t-1)}(x)$. Then,

$$\begin{aligned}i_{(s+t)}(\bot) \star\ (i_{(s+t)(\bot \times_{(s,t)} g))}{}^{x} &= \bot\ \star\ i_{(s+t-1)}[\bot\ \star (\bot \times_{(s,t-1)} (g \diamond y))] \\ &= \bot\ \star i_{(s+t-1)}[\bot] \star\ i_{(s+t-1)}[(\bot \times_{(s+1,t-1)} (g \diamond y))] \text{ Indn. Hyp} \\ &= \bot \star\ (i_s(\bot) \times_{(s+1,t-1)} (g \diamond y)) \\ &= \bot \star\ (i_s(\bot) \times_{(s+1,t-1)} (g \diamond x)) \\ &= ((i_s(\bot)) \times_{(s+1,t)} g) \diamond x\end{aligned}$$

  $x \in D_{(s+t)} \Rightarrow [g \diamond x = g \diamond x_{(t-1)}]$, and $x_{(t-1)} = y_{(t-1)}$.

- $\{\lambda x.\bot\} = [f]_1$, $\bot = [g]_1$. Then,
  $f \times_{(s,t)} g = \bot \star up(x \in D_{(s+t-1)} \mapsto [f \diamond x \times_{(s-1,t)} \bot])$. So,
  $i_{(s+t)}(f \times_{(s,t)} g) = \bot \star i_{(s+t)}[up(x \in D_{(s+t-1)} \mapsto [f \diamond x \times_{(s-1,t)} \bot])]$. The first two conditions in the definition 12 are checked easily.
  Let $x \in D_{(s+t)}$, $y = j_{(s+t-1)}(x)$. Then,

$$\begin{aligned}
(i_{(s+t)}(f \times_{(s,t)} g)) \diamond x &= \bot \star i_{(s+t-1)}[\bot \star (f \diamond x) \times_{(s-1,t)} \bot] \\
&= \bot \star i_{(s+t-1)}[\bot] \star i_{(s+t-1)}[(f \diamond y) \times_{(s,t-1)} \bot] \quad \textbf{Indn. Hyp} \\
&= \bot \star [(i_{(s-1)}(f \diamond y)) \times_{(s,t-1)} \bot] \; (f \diamond y = (f_s \diamond y_{(s-1)}) \\
&= \bot \star [(f \diamond y)) \times_{(s,t-1)} \bot] \quad (f = i_s(f)) \\
&= \bot \star [(i_s(f) \diamond y) \times_{(s,t-1)} \bot] \quad (i_s(f) \diamond x = i_s(f) \diamond y) \\
&= \bot \star [(i_s(f) \diamond x) \times_{(s,t-1)} \bot] \\
&= ((i_s(f)) \times_{(s+1,t)} \bot) \diamond x \qquad \blacksquare
\end{aligned}$$

So, we can drop the subscripts on the $\times$. The above lemmas ensure that $\times : B(D) \times B(D) \to B(D)$ is well-defined and monotone. Extend $\times$ to a continuous function $\times : D \times D \to D$. The following lemma delineates some algebraic properties that describes the interaction between $\times$, $\star$ and $\times$, $\diamond$. The proofs are done for finite elements of D. The results for arbitrary elements of D follow from the continuity of all functions involved.

Lemma 36. Let d, ,e, f, $x \in B(D)$. Then

1. $d \times (e \star f) = (d \times f) \star (e \times f)$
2. $(d \times \bot) \star \bot = (d \times \bot)$
3. $\bot \times \bot = \bot$
4. $d \times e = e \times d$
5. $[(d \uplus \bot \neq d) \wedge (e \uplus \bot \neq e)] \Rightarrow$
   $(d \times e) \diamond f = ((d \diamond f) \times e) \uplus (d \times (e \diamond f))$
6. $(d \uplus \bot \neq d) \Rightarrow$
   $(d \times e) \diamond f = (d \times e) \diamond f \uplus d \diamond f$
7. $d \times (e \times f) = (d \times e) \times f)$

Proof: The proofs of 1, 2, 3, 4, 5, 6 are immediate from the definitions of the indexed version of $\times$. 7 is proved below. The proof uses the indexed version of the definition of $\times$. We prove
$(f \times_{(s,t)} g) \times_{(s+t,u)} h = f \times_{(s,t+u)} (g \times_{(t,u)} h$. The proof proceeds by induction on $s + t + u$. The base case is immediate. For the induction step, from the bilinearity of $\times_{(s,t)}$, $\times_{(s,t+u)}$, it suffices to show the result for singletons f, g, h. The proof proceeds by cases on $[f]_1$, $[g]_1$, $[h]_1$.

1. $[f]_1 = \bot$, $[g]_1 = \bot$, $[h]_1 = \bot$. Result is immediate.

2. $[f]_1 \neq \bot$, $[g]_1 \neq \bot$, $[h]_1 \neq \bot$. The first two clauses in definition 12 are verified easily. So, proof is complete if
$(\forall x \in D_{(s+t+u-1)})\ [((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x = (f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x]$. From definitions,
$((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x$
$= [((f \times_{(s,t)} g) \diamond x) \times_{(s+t-1,u)} h] \star\ [(f \times_{(s,t)} g)] \times_{(s+t,u-1)} (h \diamond x)]$
$= [(f \diamond x \times_{(s-1,t)} g) \star (f \times_{(s,t-1)} g \diamond x) \times_{(s+t-1,u)} h] \star\ [(f \times_{(s,t)} g)] \times_{(s+t,u-1)} (h \diamond x)]$
$= ((f \diamond x \times_{(s-1,t)} g) \times_{(s+t-1,u)} h) \star ((f \times_{(s,t)} g \diamond x) \times_{(s+t-1,u)} h) \star\ ((f \times_{(s,t)} g) \times_{(s+t,u-1)} (g \diamond x))$ Indn. Hyp
$= (f \diamond x \times_{(s-1,t+u)} (g \times_{(t,u)} h)) \star (f \times_{(s,t+u-1)} (g \diamond x) \times_{(t-1,u)} h) \star\ (f \times_{(s,t+u-1)} (g \times_{(t,u-1)} h \diamond x))$
$= (f \diamond x \times_{(s-1,t+u)} (g \times_{(t,u)} h)) \star (f \times_{(s,t+u-1)} ((g \times_{(t,u)} h) \diamond x))$

3. $[f]_1 = \bot$, $[g]_1 = \bot$, $[h]_1 \neq \bot$. From definitions,we have

$$\begin{aligned}(f \times_{(s,t)} g) \times_{(s+t,u)} h &= \bot \star\ up(x \in D_{(s+t+u-1)} \mapsto [\bot \times_{(s+t,u-1)} (h \diamond x)])\\ f \times_{(s,t+u)} (g \times_{(t,u)} h) &= \bot \star\ \bot \times_{(s,t+u)} (up(x \in D_{(t+u-1)} \mapsto [\bot \times_{(t,u-1)} (h \diamond x)]))\end{aligned}$$

The first two clauses in definition 12 are verified easily. So, proof is complete if $(\forall x \in D_{(s+t+u-1)})\ [((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x = (f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x]$.
From definitions,

$$((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x = \bot \star\ (\bot \times_{(s+t,u-1)} (h \diamond x))$$

Also,

$$\begin{aligned}(f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x &= \bot \star\ (\bot \times_{(s,t+u-1)} (\bot \times_{(t,u-1)} (h \diamond x))) \text{ Indn. Hyp}\\ &= \bot \star\ ((\bot \times_{(s,t)} \bot) \times_{(s+t,u-1)} (h \diamond x))\\ &= \bot \star\ ((\bot \times_{(s+t,u-1)} (h \diamond x))\end{aligned}$$

4. $[f]_1 \neq \bot$, $[g]_1 = \bot$, $[h]_1 = \bot$. As in the preceding case.

5. $[f]_1 = \bot$, $[g]_1 \neq \bot$, $[h]_1 = \bot$. Follows from Part 4 of the lemma.

6. $[f]_1 = \bot$, $[g]_1 \neq \bot$, $[h]_1 \neq \bot$. From definitions,

$$\begin{aligned}(f \times_{(s,t)} g) \times_{(s+t,u)} h &= [\bot \star\ up(x \in D_{(s+t-1)} \mapsto (\bot \times_{(s,t-1)} g \diamond x))] \times_{(s+t,u)} h\\ &= (\bot \times_{(s+t,u)} h) \star\ (up(x \in D_{(s+t-1)} \mapsto (\bot \times_{(s,t-1)} g \diamond x)) \times_{(s+t,u)} h)\\ &= \bot \star\ up(x \in D_{(s+t+u-1)} \bot \times_{(s+t,u-1)} (h \diamond x))\\ &\quad \star\ (up(x \in D_{(s+t)} \mapsto (\bot \times_{(s,t-1)} g \diamond x))) \times_{(s+t,u)} h)\end{aligned}$$

So, we have,

$$\begin{aligned}((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x &= \bot \star \bot \times_{(s+t,u-1)} (h \diamond x) \star ((\bot \times_{(s,t-1)} g \diamond x) \times_{(s+t-1,u)} h) \\ &\quad \star (up(x \in D_{(s+t-1)} \mapsto (\bot \times_{(s,t-1)} g \diamond x))) \times_{(s+t,u-1)} h \diamond x\end{aligned}$$

From definitions,
$f \times_{(s,t+u)} (g \times_{(t,u)} h)$
$= f \times_{(s,t+u)} (up(x \in D_{(t+u-1)} \mapsto [(g \times_{(t,u-1)} h \diamond x) \star (g \diamond x \times_{(t-1,u)} h)]))$
$= \bot \star (up(x \in D_{(s+t+u-1)} \mapsto [(\bot \times_{(s,t+u-1)} (g \times_{(t,u-1)} h \diamond x)) \star (\bot \times_{(s,t+u-1)} (g \diamond x \times_{(t-1,u)}$
$h))]))$
So, we have
$(f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x$
$= \bot \star (\bot \times_{(s,t+u-1)} (g \times_{(t,u-1)} h \diamond x))$
$\star (\bot \times_{(s,t+u-1)} (g \diamond x \times_{(t-1,u)} h))$.
The first two clauses in definition 12 are verified easily. So, proof is complete if
$(\forall x \in D_{(s+t+u-1)}) [((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x = (f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x]$.
From induction hypothesis, $\bot \times_{(s,t+u-1)} (g \diamond x \times_{(t-1,u)} h)$
$= \bot \times_{(s,t+u-1)} (g \diamond x \times_{(t-1,u)} h)$.
Also,

$$\begin{aligned}\bot \times_{(s+t,u-1)} (h \diamond x) \star \quad & (up(x \in D_{(s+t-1)} \mapsto (\bot \times_{(s,t-1)} g \diamond x))) \times_{(s+t,u-1)} h \diamond x \\ &= (\bot \star (up(x \in D_{(s+t-1)} \mapsto (\bot \times_{(s,t-1)} g \diamond x)))) \times_{(s+t,u-1)} h \diamond x \\ &= (\bot \times_{(s,t)} g) \times_{(s+t,u-1)} h \diamond x \text{ Indn. Hyp} \\ &= \bot \times_{(s,t+u-1)} (g \times_{(t,u-1)} h \diamond x)\end{aligned}$$

Hence, the result.

7. $[f]_1 \neq \bot$, $[g]_1 = \bot$, $[h]_1 = \bot$. As in preceding case.

8. $[f]_1 \neq \bot$, $[g]_1 = \bot$, $[h]_1 \neq \bot$. From definitions,

$$\begin{aligned}(f \times_{(s,t)} g) \times_{(s+t,u)} h &= (\bot \star up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot])) \times_{(s+t,u)} h \\ &= (\bot \times_{(s+t,u)} h) \star (up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot]) \times_{(s+t,u)} h) \\ &= (\bot \star up(x \in D_{(s+t+u-1)} \mapsto \bot \times_{(s+t,u-1)} h \diamond x)) \\ &\quad \star (up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot]) \times_{(s+t,u)} h)\end{aligned}$$

Let $x \in D_{(s+t+u-1)}$. We have $((f \times_{(s,t)} g) \times_{(s+t,u)} h) \diamond x$
$= \bot \star (\bot \times_{(s+t,u-1)} h \diamond x)$
$\star (((f \diamond x) \times_{(s-1,t)} \bot) \times_{(s+t-1,u)} h)$
$\star (up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot]) \times_{(s+t,u-1)} h \diamond x)$.
From definitions, $f \times_{(s,t+u)} (g \times_{(t,u)} h)$
$= f \times_{(s,t+u)} (\bot \star up(x \in D_{(t+u-1)} \mapsto \bot \times_{(t,u-1)} (h \diamond x)))$
$= (f \times_{(s,t+u)} \bot) \star (f \times_{(s,t+u)} up(x \in D_{(t+u-1)} \mapsto \bot \times_{(t,u-1)} (h \diamond x)))$

$= \bot \star\ up(x \in D_{(s+t+u-1)} \mapsto (f \diamond x \times_{(s-1,t+u)} \bot)) \star\ (f \times_{(s,t+u)} up(x \in D_{(t+u-1)} \mapsto \bot \times_{(t,u-1)} (h \diamond x)))$.
Let $x \in D_{(s+t+u-1)}$. We have
$(f \times_{(s,t+u)} (g \times_{(t,u)} h)) \diamond x$
$= \bot \star\ (f \diamond x \times_{(s-1,t+u)} \bot)) \star\ (f \diamond x \times_{(s-1,t+u)} up(x \in D_{(t+u-1)} \mapsto \bot \times_{(t,u-1)} (h \diamond x)))$
$\star\ (f \times_{(s,t+u-1)} (\bot \times_{(t,u-1)} (h \diamond x)))$
Also,

$$\begin{array}{rcl} (\bot \times_{(s+t,u-1)} h \diamond x) & & \star\ (up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot]) \times_{(s+t,u-1)} h \diamond x) \\ & = & (\bot \star\ (up(x \in D_{s+t-1} \mapsto [(f \diamond x) \times_{(s-1,t)} \bot]) \times_{(s+t,u-1)} h \diamond x) \\ & = & (f \times_{(s-1,t)} \bot) \times_{(s+t,u-1)} h \diamond x) \end{array}$$

Also, by induction hypothesis,

$$\begin{array}{rcl} (f \diamond x \times_{(s-1,t)} \bot) \times_{(s+t-1,u)} h & = & (f \diamond x) \times_{(s-1,t+u)} (\bot \times_{(t,u)} h) \\ & = & (f \diamond x) \times_{(s-1,t+u)} (\bot \star\ up(x \in D_{(t+u-1)} \mapsto \bot \times_{(t,u-1)} (h \diamond x)) \end{array}$$

Result follows from the linearity of $\times_{(s-1,t+u)}$ in its right argument. ∎

7 Adequacy

In this section, we establish a relationship between $\bot$ and non-termination in the calculus. This is, of course, crucial if our mathematical model is to say anything interesting about computation. The proof superficially resembles the proof of adequacy in the setting of the lazy lambda calculus [1]. The details, however, are rather more intricate than that situation as we have to deal with many possible reduction sequences.

The plan of the proof is as follows. We first introduce a labelled calculus and show that it is strongly normalizing. We then consider a reduction strategy $\rightarrow_\omega$. We show that any $\rightarrow_\omega$ reduction in the labelled calculus can be mimicked in the γ-calculus. We then define a semantics for the labelled calculus and show that the meaning of a completely labelled term is less than the "union" of the meanings of all terms derived from one step $\rightarrow_\omega$ reductions. Because the fully labelled calculus is strongly normalizing and reduction is finitely-branching, we can classify all the "normal forms" that might exist after a fully labelled term is reduced. We can also show that the meaning of a term in the γ-calculus is given by the least upper bound of the meanings of the completely labelled terms derived from it. If we have a term M that never terminates, i.e. $\neg(M\Downarrow^{may})$, we can inspect all the terms that arise from reducing all its completely labelled versions and show that they all denote $\bot$. Thus the original term itself must have meaning bottom.

A similar but slightly more subtle argument is used for the "must converge" case. Suppose that we have a term, M, satisfying $\neg(M\Downarrow^{must})$. Reductions in the γ-calculus cannot be mimicked completely in the labelled version. However, if we examine a divergent reduction sequence of M and attempt to mimic it in the labelled calculus, we will reach a point where the head redex has label 0. At this point we know that the meaning of the original term must "contain" $\bot$.

7.1 Labelled calculus

The terms of the labelled calculus with bottom, denoted $BC^{\omega}\perp$ is defined by the folowing grammar
Terms ::= $x \mid\mid \lambda\langle x_1 \ldots x_k\rangle.M \mid\mid MN \mid\mid M|N \mid\mid M^n$
where $n \in \omega$. Following the notation used in the study of the lazy lambda calculus [16], we have

- $|M|$ is notation for the term got by erasing the labels of M.
- *subterm*(M) denotes the set of all subterms of M
- $Seq^{\star}$ denotes the set of all non-empty finite sequences of ω.
- A labelled term M is formalised as a pair (M, I_M) where
  I_M : *subterm*$(M) \rightarrow Seq^{\star} \bigcup\{\infty\}$, maps a subterm N of M to the non-empty sequence of (nested)labels of N. If N has no labels then $I_M(N) = \infty$.
- The set of completely labelled labelled terms CL is defined by structural induction as follows.
  - $x^n \in CL$
  - $M \in CL,\ N \in CL \Rightarrow$
    $M^n,\ (MN)^n, (\lambda\langle x_1, \ldots x_k\rangle.M)^n, M|N \in CL$

  Define
  $CL(M) = \{I_M | (M, I_M)$ is completely labelled $\}$

Definition 13. The syntactic equality $\equiv$ is the congruence (with respect to substitution) that is generated by the following equation:

$$p|(q|r) \equiv\ (p|q)|r$$

Let $[x \mapsto N]M$ be notation for the usual notion of substitution. The reduction relation is presented as a transition system. (The presentation here is semi-formal, but is formalised clearly in the appendix where strong normalisation is proved).

- $(M^m)^n \rightarrow_l M^{min(m,n)}$
- $(\lambda\langle x_1 \ldots x_k\rangle.M)^{n+1}N \rightarrow_l \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N^n]M$
  if $1 \leq i \leq k$
- $(M_1| \ldots |(\lambda\langle x_1 \ldots x_k\rangle.M)^{n+1}| \ldots |M_n)N \rightarrow_l$
  $M_1| \ldots |\lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N^n]M| \ldots |M_n,$
  if $1 \leq i \leq k$
- $M^0 \rightarrow_l \perp$

- $\bot M \rightarrow_l \bot$
- $\bot^n \rightarrow_l \bot$
- $(M_1 | \ldots M_m)^n \rightarrow_l M_1^n | \ldots | M_m^n$
- $M \rightarrow_l M' \Rightarrow M|N \rightarrow_l M'|N$
- $N \rightarrow_l N' \Rightarrow M|N \rightarrow_l M|N'$
- $M \rightarrow_l M' \Rightarrow MN \rightarrow_l M'N$
- $N \rightarrow_l N' \Rightarrow MN \rightarrow_l MN'$

$\overset{*}{\rightarrow}_l$ is the reflexive and transitive closure of $\rightarrow_l$.

Theorem 2. (Strong normalisation)
Every reduction starting from a completely labelled term (M, I) terminates.

Proof: The proof is complicated but unsurprising. A detailed sketch of the proof is in an appendix. ∎

The following definitions are needed to access the label of aparticular term. Define a map $min : Seq^\star \bigcup \{\infty\} \rightarrow \omega \bigcup \{\infty\}$ as

- $min(\vec{l}) =$ minimum of $\vec{l}$, if $\vec{l} \in Seq^\star$
- $min(\vec{l}) = \infty$, if $\vec{l} \notin Seq^\star$

Next, we define a reduction strategy $\rightarrow_\omega$ on the labelled terms as follows. This is done in a manner similar to the treatment of the unlabelled calculus. The following definitions, though simple are set out so that the proofs later will become intuitive. Define, by mutual recursion:
$Terms_1 ::= x \;||\; \lambda\langle x_1 \ldots x_k\rangle.M \;||\; MN || \; P^n$
$Terms_2 ::= M \;||\; M|N \;||\; Q^n$
where $M,\ N \in Terms_1 \bigcup Terms_2,\ P \in Terms_1,\ Q \in Terms_2$. The following definition is intended to capture the "number" of $t_i's$.

Define $len : Terms_2 \rightarrow Int$ as follows:

- $len(p) = 1$, if $p \in Terms_1$
- $len(p|q) = len(p) + len(q)$

It can be checked that this function is well-defined on the terms quotiented by the syntactic equality $\equiv$. The following definition is intended to capture the "position" of t_i in $t_1| \ldots t_n$. Define a partial function
$index : \omega \times Terms_2 \rightarrow Terms_1$ as follows:

- $index(n, p) =$ undefined if $len(p) \leq n \wedge \ len(p) \neq n$

- $index(1,p) = p$, if $p \in Terms_1$
- $index(n, p|q) = index(n,p)$, if $n \leq len(p)$
- $index(n, p|q) = index(n - len(p), q)$, if $len(p) \leq n \wedge len(p) \neq n$

Now, we have the machinery required to define the reduction strategy $\rightarrow_\omega$.

- $(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}} N \rightarrow_{\omega\langle 1,\, i\rangle} \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N^n]M$
  if $1 \leq i \leq k$, $\min(\vec{l}) = n+1$
- $(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}} N \rightarrow_{\omega\langle 1,\, i\rangle} \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M$
  if $1 \leq i \leq k$, $\min(\vec{l}) = \infty$
- $index(s, (M_1| \ldots |(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}}| \ldots |M_n)) = \lambda\langle x_1 \ldots x_k\rangle.M \Rightarrow$
  $(M_1| \ldots |(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}}| \ldots |M_n) N \rightarrow_{\omega\langle s,\, i\rangle}$
  $M_1| \ldots |\lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N^n]M| \ldots |M_n$
  if $1 \leq i \leq k$, $\min(\vec{l}) = n+1$
- $index(s, (M_1| \ldots |(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}}| \ldots |M_n)) = \lambda\langle x_1 \ldots x_k\rangle.M \Rightarrow$
  $(M_1| \ldots |(\lambda\langle x_1 \ldots x_k\rangle.M)^{\vec{l}}| \ldots |M_n) N \rightarrow_{\omega\langle s,\, i\rangle}$
  $M_1| \ldots |\lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M| \ldots |M_n$
  if $1 \leq i \leq k$, $\min(\vec{l}) = \infty$
- $M \rightarrow_{\omega\sigma} M' \Rightarrow M|N \rightarrow_{\omega\sigma} M'|N$
- $N \rightarrow_{\omega\sigma} N' \Rightarrow M|N \rightarrow_{\omega\sigma'} M|N'$
  where $\sigma' = \langle first(\sigma) + len(N), second(\sigma)\rangle$
- $M \rightarrow_{\omega\sigma} M' \Rightarrow MN \rightarrow_{\omega\langle 1,\, \sigma\rangle} M'N$

Define:

- $\overset{*}{\rightarrow}_\omega$is the reflexive, transitive closure of $\rightarrow_\omega$
- $nf(\rightarrow_\omega) = \{M |\ M \notin dom(\rightarrow_\omega)\}$

We have the following corollary to the strong normalisation theorem for completely labelled terms.

Lemma 37. The reduction $\rightarrow_\omega$ restricted to completely labelled terms is strongly normalising.

The closed, completely labelled terms that are in $nf(\rightarrow_\omega)$ can be described completely. Define sets S_1, S_2, S_3, S_4 inductively as follows:

- S_1
  - $(\lambda\langle x_1 \dots x_k\rangle.M)^{\vec{l}} \in S_1$ if $FV(M) \subseteq \{x_1 \dots x_k\}$, $min(\vec{l}) = 0$
  - $M \in S_1 \Rightarrow MN \in S_1$, if N is closed.
  - $M, N \in S_1 \Rightarrow M|N \in S_1$
- S_2
  - $(\lambda\langle x_1 \dots x_k\rangle.M)^{\vec{l}} \in S_2$ if $FV(M) \subseteq \{x_1 \dots x_k\}$, $min(\vec{l}) = n+1$
  - $M, N \in S_1 \Rightarrow M|N \in S_2$
- S_3
  - $M \in S_1,\ N \in S_2 \Rightarrow M|N \in S_3,\ N|M \in S_3$
  - $M \in S_3,\ N \in S_2 \Rightarrow M|N \in S_3,\ N|M \in S_3$
  - $M \in S_3,\ N \in S_1 \Rightarrow M|N \in S_3,\ N|M \in S_3$
- S_4
  - $M \in S_3 \bigcup S_1 \Rightarrow MN \in S_4$
  - $M \in S_4, \Rightarrow MN \in S_4$

Note that the sets S_1, S_2, S_3 are pairwise disjoint.

Lemma 38. (Classification of normal forms of closed terms)
Let (M, I_M) be a completely labelled term. Then
$M \in nf(\rightarrow_\omega)$, M closed $\Rightarrow M \in S_1 \bigcup S_2 \bigcup S_3$

Proof: Structural induction. ∎

The reduction relations $\rightarrow_\omega$ and $\longrightarrow$ are closely related. Since we need to talk about specific redices, we define the notion of a redex occurring at a position. This is done by structural induction.

- $\lambda\langle x_1 \dots x_k\rangle.M$ occurs at position $\langle 1, i\rangle$ in $\lambda\langle x_1 \dots x_k\rangle.M$, if $1 \leq i \leq k$.
- P occurs at position σ in $M \Rightarrow P$ occurs at position $\langle 1, \sigma\rangle$ in MN
- P occurs at position $\langle a, \sigma\rangle$ in $M \Rightarrow$
  - P occurs at position $\langle a, \sigma\rangle$ in $M|N$
  - P occurs at position $\langle a + len(N), \sigma\rangle$ in $N|M$

Many labels are omitted in the following discussion, for the sake of clarity.

Lemma 39. (Relating $\rightarrow_\omega$ and $\longrightarrow$)

1. Let (M, I_M) $(N, I_N) \in BC^\omega$. Then, $(M, I_M) \rightarrow_\omega (N, I_N) \Rightarrow [M \longrightarrow N \ \vee \ M = N]$.
2. Let $M \longrightarrow_\sigma N$. Let I_M be a labelling of M. Then, we have one of the following:
   - $(\exists I_N)$ $[(M, I_M) \rightarrow_{\omega\sigma} (N, I_N)]$, or
   - The redex P occurring at σ has minimum label 0. Also, in this case, $(M, I_M) \in S_4$.

Proof: Structural Induction. ∎

The relationship between the sets $S_i, i = 1 \ldots 4$ and the convergence predicates $\Downarrow^{may}$ and $\Downarrow^{must}$ is described in the following lemma.

Lemma 40. Let M be a term. Let (M, I_M) be a complete labelling of M.

1. $\neg(M \Downarrow^{may}) \Rightarrow$
   $(\forall (N, I_N))$ $[((M, I_M) \overset{*}{\rightarrow}_\omega (N, I_N) \wedge (N, I_N) \in nf(\rightarrow_\omega)) \Rightarrow (N, I_N) \in S_1]$
2. $\neg(M \Downarrow^{must}) \Rightarrow$
   $(\exists (N, I_N))$ $[((M, I_M) \overset{*}{\rightarrow}_\omega (N, I_N) \wedge (N, I_N) \in S_4]$

Proof:

1. $(N, I_N) \in S_2 \bigcup S_3 \Rightarrow N \Downarrow^{may}$. The result follows from lemma 39.1.
2. $\neg(M \Downarrow^{must}) \Rightarrow$ there is an infinite reduction sequence $M = M_0 \longrightarrow M_1 \longrightarrow M_2 \ldots M_k \ldots$. Let (M, I_M) be a complete labelling. Since the reduction relation $\rightarrow_\omega$ is strongly normalising, and using lemma 39.2 there exists a k such that, $(M, I_M) \overset{*}{\rightarrow}_\omega (M_k, I_{M_k})$, and $(M_k, I_{M_k}) \in S_4$. ∎

7.2 Approximable models

First, we abstract out the properties that are required for the proof of adequacy. The following definition is an extension of the definition of models with approximable application that is used for an analogous result for the lazy lambda calculus [16].

Definition 14. $\langle D, Fun, Gr, \uplus, \star, \times, \diamond \rangle$ is approximable if

1. $\langle D, \uplus \rangle$ is a continuous algebra.
2. $\langle P(D), \star \rangle$ is a continuous algebra, with a singleton embedding $\{\!|\ |\!\} : (D \rightarrow D)_\perp \rightarrow P(D)$
3. $(Gr : P(D) \rightarrow D, Fun : D \rightarrow P(D))$ is a linear embedding-projection pair.
4. D has a least element $\perp$.
5. $\diamond$ is continuous.

6. There are continuous maps $[]_n : D \to D$, for each $n \in \omega$ satisfying:

   (a) $d = \bigsqcup_{n\in\omega} d_n$

   (b) $d_0 = \perp$

   (c) $\perp \diamond d = \perp$

   (d) $d_{n+1} \diamond e \sqsubseteq [d \diamond e_n]_n$

   (e) $[[d]_n]_m \sqsubseteq [d]_{min(n,m)}$

7. $\uplus : D \times D \to D$, is a continuous function that satisfies:

   (a) $d \uplus e = e \uplus d$

   (b) $(d \uplus e) \uplus f = d \uplus (e \uplus f)$

   (c) $d \uplus d = d$

   (d) $[d \uplus e]_n \sqsubseteq [d]_n \uplus [e]_n$

   (e) $(d \uplus e) \diamond f \sqsubseteq (d \diamond e) \uplus (e \diamond f)$

8. $\times : D \times D \to D$, is a continuous function that satisfies:

   (a) $\perp \times \perp = \perp$

   (b) $(d \times \perp) \uplus \perp = d \times \perp$

   (c) $d \times e = e \times d$

   (d) $(d \times e) \times f = d \times (e \times f)$

   (e) $[d \times e]_n \sqsubseteq [d]_n \times [e]_n$

   (f) $d \times (e \uplus f) = (d \times e) \uplus (d \times f)$

   (g)
   - $[(d \uplus \perp \neq d) \wedge (e \uplus \perp \neq e)] \Rightarrow$
     $(d \times e) \diamond f \sqsubseteq ((d \diamond f) \times e) \uplus (d \times (e \diamond f))$
   - $(d \uplus \perp \neq d) \Rightarrow$
     $(d \times e) \diamond f \sqsubseteq (d \times e) \diamond f \uplus d \diamond f$

9. $d \sqsubseteq e \uplus \perp \Rightarrow d \uplus \perp = d$

The initial solution to the recursive domain equation $D \simeq \overline{P}(D)$ satisfies the above conditions.

The semantics of the labelled calculus is defined as follows:

- $[\![(\perp, I)]\!]\,\rho = \perp$
- $[\![(x, I)]\!]\,\rho = [\rho(x)]_{min(I(x))}$
- $[\![(\lambda\langle x\rangle M, I)]\!]\,\rho = [Gr(d \mapsto [\![(M, I_M)]\!]\,\rho[x \mapsto d]\,)]_{min(I(\lambda\langle x\rangle M))}$

- $[\![(\lambda\langle x_1, x_2\rangle M, I)]\!]\,\rho =$
  $[Gr(\star[(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I_M)]\!]\,\rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\,)),$
  $(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I_M)]\!]\,\rho[x_2 \mapsto d_1, x_1 \mapsto d_2]\,))])]_{min(I(\lambda\langle x_1, x_2\rangle M))}$
- $[\![(MN, I)]\!]\,\rho = [[\![(M, I_M)]\!]\,\rho\,)\diamond\,[\![(N, I_N)]\!]\,\rho\,]_{min(I(MN))}$
- $[\![(M|N, I)]\!]\,\rho = [[\![(M, I_M)]\!]\,\rho\,\times\,[\![(N, I_N)]\!]\,\rho\,]_{min(I(M|N))}$

The following lemma establishes the relationship between $\perp$ and the syntactic classes S_1 and S_4.

Lemma 41. Let (M, I) be a completely labelled term. Then,

1. $(M, I) \in S_1 \Rightarrow [\![(M, I)]\!]\,\rho = \perp$
2. $(M, I) \in S_4 \Rightarrow [\![(M, I)]\!]\,\rho \uplus \perp = [\![(M, I)]\!]\,\rho$

Proof: Both proofs proceed by structural induction.

1. Recall that S_1 was defined inductively as:
   - $(\lambda\langle x_1 \dots x_k\rangle.M)^{\vec{l}} \in S_1$ if $FV(M) \subseteq \{x_1 \dots x_k\}$, $min(\vec{l}) = 0$
   - $M \in S_1 \Rightarrow MN \in S_1$, if N is closed.
   - $M, N \in S_1 \Rightarrow M|N \in S_1$

   Laws 6(a), 6(c), 8(a) of the definition of approximable models are used respectively for the cases.

2. Recall that S_3 was defined inductively as:
   - $M \in S_1,\ N \in S_2 \Rightarrow M|N \in S_3,\ N|M \in S_3$
   - $M \in S_3,\ N \in S_2 \Rightarrow M|N \in S_3,\ N|M \in S_3$
   - $M \in S_3,\ N \in S_1 \Rightarrow M|N \in S_3,\ N|M \in S_3$

   Laws 8(f), 8(c), 8(b), 8(d) of the definition of approximable models are used to prove the result. Recall that S_4 was defined inductively as:
   - $M \in S_3 \bigcup S_1 \Rightarrow MN \in S_4$
   - $M \in S_4, \Rightarrow MN \in S_4$

   Laws 6(c) of the definition of approximable models is used to prove the result. ∎

Lemma 42. Let (M, I) be a completely labelled term. Let (M_j, I_j), $j = 1 \dots n$, be all the terms such that $(M, I) \rightarrow_\omega (M_j, I_j)$. Then,
$[\![(M, I)]\!]\,\rho \sqsubseteq [\![(M_1, I_1)]\!]\,\rho \uplus \dots \uplus [\![(M_n, I_n)]\!]\,\rho$.

Proof: Proof is by structural induction. Let (M, I) be a completely labelled term.
Case 0: $M = NP_1P_2$. Result follows from structural induction on NP_1, and the monotonicty of $\diamond$.

Case 1: $M = N|P$.
Let N_i, $i \leq i \leq n$ be all the possible terms such that $(N, I_N) \rightarrow_\omega (N_i, I_{N_i})$. Similarly, let P_j, $i \leq j \leq m$ be all the possible terms such that
$(P, I_P) \rightarrow_\omega (P_j, I_{P_j})$. From structural induction hypothesis,

- $[\![(N, I_N)]\!]\ \rho \sqsubseteq [\![(N_1, I_{N_1})]\!]\ \rho \uplus \ldots [\![(N_n, I_{N_n})]\!]\ \rho$
- $[\![(P, I_P)]\!]\ \rho \sqsubseteq [\![(P_1, I_{P_1})]\!]\ \rho \uplus \ldots [\![(P_m, I_{P_m})]\!]\ \rho$

Result follows from the bilinearity of $\times$ as given by laws $8(f)$ and $8(c)$ of approximable models.

Case 2: $M = (\lambda\langle x_1, x_2\rangle.N)P$. Let $M' = \lambda\langle x_1, x_2\rangle.N$
Let $min(I((\lambda\langle x_1, x_2\rangle.N)) = n + 1$. The two possible one step reductions($\rightarrow_\omega$) yield $\lambda x_1.[x_2 \mapsto P^n]M$ and $\lambda x_2.[x_1 \mapsto P^n]M$. By definitions, we have
$[\![(\lambda\langle x_1, x_2\rangle N, I)]\!]\ \rho\ =$
$[Gr(\star[(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\ )),$
$(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_2 \mapsto d_1, x_1 \mapsto d_2]\ ))])]_{min(I(M')}$
Using the linearity of Gr and law $7(d)$ of approximable models
$[\![(\lambda\langle x_1, x_2\rangle N, I)]\!]\ \rho \sqsubseteq$
$[Gr([(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\ ))]_{min(I(M'))} \uplus$
$[Gr(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_2 \mapsto d_1, x_1 \mapsto d_2]\ ))])]_{min(I(M'))}$
From definitions
$[\![((\lambda\langle x_1, x_2\rangle.N)P, I)]\!]\ \rho\ = [[\![(M, I)]\!]\ \rho \diamond [\![(P, I)]\!]\ \rho\ ]_{min(I(MN))}$.
Using law $7(e)$ of approximable models
$[\![((\lambda\langle x_1, x_2\rangle.N)P, I)]\!]\ \rho \sqsubseteq$
$[Gr([(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\ ))])]_{min(I(M'))} \diamond [\![(P, I)]\!]\ \rho$
$\uplus\ [Gr([(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_2 \mapsto d_1, x_1 \mapsto d_2]\ ))])]_{min(I(M'))} \diamond [\![(P, I)]\!]\ \rho$.
From law $6(d)$ of approximable models, we have
$[Gr([(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_1 \mapsto d_1, x_2 \mapsto d_2]\ ))])]_{min(I(M')} \diamond [\![(P, I)]\!]\ \rho$
$\sqsubseteq [\![(\lambda x_2.[x_1 \mapsto P^n]M, I)]\!]\ \rho$
From law $6(d)$ of approximable models, we have,
$[Gr([(d_1 \mapsto Gr(d_2 \mapsto [\![(M, I)]\!]\ \rho[x_1 \mapsto d_2, x_2 \mapsto d_1]\ ))])]_{min(I(M'))} \diamond [\![(P, I)]\!]\ \rho$
$\sqsubseteq [\![(\lambda x_1.[x_2 \mapsto P^n]M, I)]\!]\ \rho$. The result follows.

Case 3: $M = NP \wedge N = N_1| \ldots |N_k$.
For notational convenience, assume $k = 2$. We have the following cases.

- All the N_is are of form $\lambda\langle x_1, \ldots x_k\rangle.N_i'$. Then, all the one step reductions are β-reductions involving P. The result follows from law $8(g)$ and the case above.

- N_i is not of form $\lambda\langle x_1, \ldots x_s\rangle.N_i'$, for $i = 1, 2$. Then, all the one-step reductions are of form

  - $N_1 \rightarrow_\omega N_1' \wedge\ (N_1|N_2)P \rightarrow_\omega (N_1'|N_2)P$
  - $N_2 \rightarrow_\omega N_2' \wedge\ (N_1|N_2)P \rightarrow_\omega (N_1|N_2')P$

  Then, result follows from structural induction hypothesis and observations 8(c), 8(f), 7(e).

- $N_1 = \lambda\langle x_1, \ldots x_s\rangle.N_1'$ and N_2 is not of this form. The one-step reductions are of form,

  - $N_2 \longrightarrow N_2' \wedge\ (N_1|N_2)P \longrightarrow (N_1|N_2')P$
  - $N_1P \longrightarrow N_1' \wedge\ (N_1|N_2)P \longrightarrow (N_1'|N_2)$

  Let Q_j, $j = 1 \ldots t$ be such that $N_2 \longrightarrow Q_j$. From structural induction hypothesis, $[\![N_2]\!]\ \rho \sqsubseteq [\![Q_1]\!]\ \rho\ \uplus \ldots [\![Q_t]\!]\ \rho$. Using law 8(c), 8(f), and structural induction hypothesis,
  $[\![N_1|N_2]\!]\ \rho \sqsubseteq [\![N_1|Q_1]\!]\ \rho\ \uplus \ldots [\![N_1|Q_t]\!]\ \rho$. Result now follows using law 8(g). ∎

The next lemma says that one can recover the meaning of a term in the original calculus by taking the least upper bound of the meanings of all the fully labelled terms.

Lemma 43. $[\![M]\!]\ \rho\ = \bigsqcup_{CL(M)}[[\![(M, I)]\!]\ \rho \mid I \in CL(M)]$

Proof: Structural induction. Note that the set $[[\![(M, I)]\!]\ \rho \mid I \in CL(M)]$ is directed.

- $M = x$. Result follows from law 6(a) of approximable models.

- $M = NP$.

$$\begin{aligned}
[\![M]\!]\ \rho &= \textstyle\bigsqcup_{n\in\omega}[[\![NP]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{n\in\omega}[[\![N]\!]\ \rho\ \diamond [\![P]\!]\ \rho\ ]_n \text{ Indn. Hyp} \\
&= \textstyle\bigsqcup_{n\in\omega}[(\bigsqcup_{I_N\in CL(N)}[\![(N, I_N)]\!]\ \rho\ ) \diamond (\bigsqcup_{I_P\in CL(P)}[\![(P, I_P)]\!]\ \rho\ )]_n \\
&= \textstyle\bigsqcup_{n\in\omega}[\bigsqcup_{I\in CL(M)}[\![(N, I_N)]\!]\ \rho\ \diamond [\![P, I_P]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{I\in CL(M)}\bigsqcup_{n\in\omega}[[\![(N, I_N)]\!]\ \rho\ \diamond [\![P, I_P]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{I\in CL(M)}[\![(N, I_N)]\!]\ \rho\ \diamond [\![P, I_P]\!]\ \rho
\end{aligned}$$

- $M = \lambda\langle x_1, x_2\rangle N$. This involves the continuity of Gr and $\uplus$ and is omitted.

- $M = N|P$

$$\begin{aligned}
[\![M]\!]\ \rho &= \textstyle\bigsqcup_{n\in\omega}[[\![N|P]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{n\in\omega}[[\![N]\!]\ \rho\ \times\ [\![P]\!]\ \rho\ ]_n \text{ Indn. Hyp} \\
&= \textstyle\bigsqcup_{n\in\omega}[(\bigsqcup_{I_N\in CL(N)}[\![(N, I_N)]\!]\ \rho\ ) \times\ (\bigsqcup_{I_P\in CL(P)}[\![(P, I_P)]\!]\ \rho\ )]_n \\
&= \textstyle\bigsqcup_{n\in\omega}[\bigsqcup_{I\in CL(M)}[\![(N, I_N)]\!]\ \rho\ \times\ [\![P, I_P]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{I\in CL(M)}\bigsqcup_{n\in\omega}[[\![(N, I_N)]\!]\ \rho\ \times\ [\![P, I_P]\!]\ \rho\ ]_n \\
&= \textstyle\bigsqcup_{I\in CL(M)}[\![(N, I_N)]\!]\ \rho\ \times [\![P, I_P]\!]\ \rho \qquad ∎
\end{aligned}$$

Lemma 44. Let M be a closed term. Then, $\neg(M\Downarrow^{may}) \Rightarrow [\![M]\!]\,\rho = \bot$

Proof: From lemma 43, $[\![M]\!]\,\rho = \bigsqcup_{CL(M)}[[\![(M,I)]\!]\,\rho \mid I \in CL(M)]$. Let (M,I) be any complete labelling of M. Let (M_j,I_j), $j = 1\dots n$, be all the terms such that $(M,I)\xrightarrow{*}_\omega (M_j,I_j) \wedge (M_j,I_j) \in nf(\rightarrow_\omega)$. Then, from lemma 42
$[\![(M,I)]\!]\,\rho \sqsubseteq [\![(M_1,I_1)]\!]\,\rho \uplus \dots \uplus [\![(M_n,I_n)]\!]\,\rho$.
From lemma 40, all the $(M_i,I_i) \in S_1$. From lemma 41, $(\forall 1 \le i \le n)\ [[\![(M_i,I_i)]\!]\,\rho = \bot$. Hence, the result. ∎

Lemma 45. Let M be a closed term. Then, $\neg(M\Downarrow^{must}) \Rightarrow [\![M]\!]\,\rho \uplus \bot = [\![M]\!]\,\rho$

Proof: From lemma 43, $[\![M]\!]\,\rho \doteq \bigsqcup_{CL(M)}[[\![(M,I)]\!]\,\rho \mid I \in CL(M)]$. Let (M,I) be any complete labelling of M. Let (M_j,I_j), $j = 1\dots n$, be all the terms such that $(M,I)\rightarrow_\omega (M_j,I_j)$. Then, from lemma 42
$[\![(M,I)]\!]\,\rho \sqsubseteq [\![(M_1,I_1)]\!]\,\rho \uplus \dots \uplus [\![(M_n,I_n)]\!]\,\rho$.
From lemma 40, $(\exists I)\ [(M_i,I_i)\xrightarrow{*}_\omega (N,J) \wedge (N,J) \in S_4]$. From lemma 41, $[[\![(N,J)]\!]\,\rho \uplus \bot = [\![(N,J)]\!]\,\rho$. Hence, the result follows. ∎

For the converse of the above two lemmas, we need the following lemma. In the proof, we use the following two facts about the model D.

1. $[(d \uplus \bot \neq d) \wedge (e \uplus \bot \neq e)] \Rightarrow$
   $(d \times e) \diamond f = ((d \diamond f) \times e) \uplus (d \times (e \diamond f))$
2. $(d \uplus \bot \neq d) \Rightarrow$
   $(d \times e) \diamond f \uplus d \diamond f = (d \times e) \diamond f$
3. $(d \uplus e) \diamond f = d \diamond f \uplus e \diamond f$
4. $(d \uplus e) \times f = (d \times f) \uplus (e \times f)$
5. $f \times (d \uplus e) = (f \times d) \uplus (f \times e)$
6. $(d \times e) \times f = d \times (e \times f)$

Lemma 46. Let M be a closed, unlabelled term. Let M_i, $i = 1\dots n$ be such that $M \longrightarrow M_i$. Then, $[\![M]\!]\,\rho = [\![M_1]\!]\,\rho \uplus \dots [\![M_n]\!]\,\rho$.

Proof: (Sketch) Proof is by structural induction.

- $M = NP_1P_2$. Follows from structural induction hypothesis on NP_1, and observation 3 above.
- $M = P_1|P_2$. Follows from structural induction hypothesis on P_1, P_2 and observation 4, 5, 6 above.
- $M = NP$, and $N = \lambda\langle x_1,\dots x_k\rangle.N'$. Proof is almost identical to the corresponding case in lemma 42.

- $M = NP$, and $N = N_1|\dots N_k$. Proof is almost identical to the corresponding case in lemma 42.

The following lemma is the converse of lemma 44 and lemma 45.

Lemma 47. let M be a closed term. Then,

- $M\Downarrow^{may} \Rightarrow [\![M]\!]\,\rho \neq \bot$
- $M\Downarrow^{must} \Rightarrow [\![M]\!]\,\rho \uplus \bot \neq [\![M]\!]\,\rho$

The following theorem is an immediate consequence of lemma 47, lemma 44, lemma 45.

Theorem 3. (Adequacy)
Let M, N be closed terms. Then, $[\![M]\!]\,\rho \sqsubseteq [\![N]\!]\,\rho \Rightarrow$M $\preceq$N

8 Conclusions and Future Work

The work in this paper represent part of a growing interest in higher-order process calculi. We feel that it is a significant achievement of Boudol's to describe a calculus that can be given a pleasant mathematical model and yet express concurrency and abstraction.

We plan to extend our model to the full calculus. We would also like to understand what it takes to make the calculus fully abstract. Given that the language has concurrency "naturally" built into it one might expect that one would get full abstraction by adding a simple convergence tester; unlike the case of the lazy lambda calculus where one needed a parallel convergence tester. This, however, seems unlikely though we do not yet have any definitive answers as yet. We also plan to understand the structure of the powerdomain more clearly as it seems to have pleasing algebraic properties. In particular, we would like a representation theorem for the elements of the powerdomain. Finally we would like to relate these semantical investigations to other formalisms and also to the implemnted systems that are starting to appear.

A Strong Normalisation

The terms of the labelled calculus with bottom, denoted $BC^{\omega}\bot$ is defined by the folowing grammar
Terms ::= $x \mid\mid \lambda\langle x_1 \dots x_k\rangle.M \mid\mid MN \mid\mid M|N \mid\mid M^n$
where $n \in \omega$.

Definition 15. The syntactic equality $\equiv$ is the congruence (with respect to substitution) that is generated by the following equation:

$$p|(q|r) \equiv (p|q)|r$$

We need basic notation to be able to refer to redices. This is done in a style similar to the techniques used in the study of the operational semantics of the unlabelled calculus. Most of the definitions that appear below have been used before in the paper, but are repeated here to make this section self contained.

Define, by mutual recursion:
$Terms_1 ::= x \mid\mid \lambda\langle x_1 \ldots x_k\rangle.M \mid\mid MN \mid\mid M^n$
$Terms_2 ::= M \mid\mid M|N$
where $M,\ N \in Terms_1 \bigcup Terms_2$. The intuitive meanings of these classes are the same as the corresponding classes in the unlabelled calculus.

Intuitively, $Terms_2$ are the terms of the form $t_1| \ldots |t_n$, where the t_i are either abstractions or applications. The following definition is intended to capture the "number" of $t_i's$.

Define $len : Terms_2 \rightarrow Int$ as follows:

- $len(p) = 1$, if $p \in Terms_1$
- $len(p|q) = len(p) + len(q)$

It can be checked that this function is well-defined on the terms quotiented by the syntactic equality $\equiv$. The following definition is intended to capture the "position" of t_i in $t_1| \ldots t_n$. Define a partial function $index : \omega \times Terms_2 \rightarrow Terms_1$ as follows:

- $index(n,p) =$ undefined if $len(p) \leq n \wedge\ len(p) \neq n$
- $index(1,p) = p$, if $p \in Terms_1$
- $index(n,p|q) = index(n,p)$, if $n \leq len(p)$
- $index(n,p|q) = index(n - len(p), q)$, if $len(p) \leq n \wedge len(p) \neq n$

Let $[x \mapsto N]M$ be notation for the usual notion of substitution. Let $M \in Terms_2 \wedge 1 \leq s \leq len(M)$. Then, $M[s \mapsto N]$ is notation for the term $M' \in Terms_2$, such that

- $len(M') = len(M)$
- If, $1 \leq t \leq len(M') \wedge s \neq t$, $index(t', M) = index(M, t)$
- $index(s, M') = N$

The reduction relation is presented as a transition system. We also formalise the notion of "different" reductions. This is done by associating an ordered pair with a reduction, that captures the position and argument of the abstraction that is involved in the reduction.

- $(M^m)^n \rightarrow_{l\langle 1,0\rangle} M^{min(m,n)}$
- $(M^0) \rightarrow_{l\langle 1,0\rangle} \perp$

- $\perp^n \rightarrow_{l\langle 1, 0\rangle} \perp$
- $\perp^n N \rightarrow_{l\langle 1, 0\rangle} \perp$
- $(\lambda\langle x_1 \ldots x_k\rangle.M)N \rightarrow_{l\langle 1, i\rangle} \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M$
  if $1 \leq i \leq k$
- If $P \in Terms_2 \wedge\ 1 \leq s \leq len(P)$, then
  $PN \rightarrow_{l\langle s, 0\rangle} M[s \mapsto index(s, P)N]$
- $M \rightarrow_{l_\sigma} M' \Rightarrow M|N \rightarrow_{l_\sigma} M'|N$
- $N \rightarrow_{l_\sigma} N' \Rightarrow M|N \rightarrow_{l_{\sigma'}} M|N'$
  where $\sigma' = \langle first(\sigma) + len(M), second(\sigma)\rangle$
- $M \rightarrow_{l_\sigma} M' \Rightarrow MN \rightarrow_{l\langle 1, \sigma\rangle} M'N$
- $N \rightarrow_{l_\sigma} N' \Rightarrow MN \rightarrow_{l\langle 2, \sigma\rangle} MN'$
- $M \rightarrow_{l_\sigma} M' \Rightarrow M^n \rightarrow_{l\langle 1, \sigma\rangle} M'^n$

Two reductions $M \longrightarrow_{\sigma'} M'$, and $M \longrightarrow_{\sigma''} M''$ are different if $\sigma' \neq \sigma''$. Note that there are only finitely many different reductions. $\overset{*}{\rightarrow}_l$ is the reflexive and transitive closure of $\rightarrow_l$.

Let $M \in \Lambda_0$. Construct a tree with labelled edges corresponding to M denoted by $T(M)$ as follows. Let $M \longrightarrow_{\sigma_i} M_i$, be all the possible different one step reductions from M. Then, the root has an edge for each label σ_i. The subtree at the node at the other end of the edge with label σ_i is the one obtained by doing the construction for M_i. Define the set of strongly normalising terms in $BC^\omega\perp$ as follows.
$SN = \{M |$there is no infinite reduction sequence $M = M_0 \longrightarrow M_1 \longrightarrow M_2 \ldots\}$. By a Konig's lemma argument, we deduce that M is SN $\Rightarrow T(M)$ is finite.

Definition 16. Let $\langle D, \leq\rangle$ be the domain of labelled, finitely-branching trees of finite depth, where the ordering relation $\leq$ is the subtree ordering.

Note that $\langle D, \leq\rangle$ is well-founded. Furthermore, if $M \in SN$ and $M \longrightarrow M'$, then $T(M') \leq T(M)$, $T(M') \neq T(M)$. $T(M)$ plays a role analogous to the role of $d(M)$ which is the length of the reduction sequence from M in the proof of strong normalisation of the labelled lambda calculus, as in Barendregt's book [3].

We need to keep track of redices explicitly. First,we define positions of subterms in a term. We define a partial function $F_M : Seq^\star \rightarrow Terms$, as follows. The intuition is that $F_M(\sigma)$, is the subterm of M at position σ. This is done by structural induction.

- $M \in Terms_1$

- $M \in Var$. Say, $M = z$. Then, $F_z(\langle 0, 0\rangle) = z$
- $M = PN$. The,
  - $F_M(\langle 0, 0\rangle) = M$
  - $F_M(\langle 1, \sigma\rangle) = F_P(\sigma)$
  - $F_M(\langle 2, \sigma\rangle) = F_N(\sigma)$
- $M = \lambda\langle x_1 \ldots x_k\rangle.P$. Let σ be of form $\langle , \rangle$.
  - $F_M(\langle 0, 0\rangle) = M$
  - $F_M(\langle 1, \sigma\rangle) = F_P(\sigma)$
- $M = P^n$. Let σ be of form $\langle , \rangle$.
  - $F_M(\langle 0, 0\rangle) = M$
  - $F_M(\langle 1, \sigma\rangle) = F_P(\sigma)$

- $M \in Terms_2 \wedge\ len(M) = n + 1 \wedge\ 1 \leq n$. Let $1 \leq i \leq len(M)$. Then
  - $F_M(\langle 0, 0\rangle) = M$
  - $F_M(\langle i, \sigma\rangle) = F_{index(i,M)}(\langle 1, \sigma\rangle), 1 \leq i$

Let $M \in BC^\omega\bot$. Let $mark : BC^\omega\bot \times Seq^\star$ be a predicate. The intuition is that certain subterms of M are marked out. The following definition passes marking information through reductions. This is done by analysing the structure of the reductions.

Definition 17. (Extending marking through reductions)

- $(M^m)^n \rightarrow_{l\langle 1,0\rangle} M^{min(m,n)}$. Then
  - $mark((M^m)^n, \langle 1, \langle 1, \sigma\rangle\rangle) \Rightarrow mark(M^{min(m,n)}, \langle 1, \sigma\rangle)$
  - $mark((M^m)^n, \langle 1, \langle 0, 0\rangle\rangle) \vee mark((M^m)^n, \langle 1, \langle 1, \langle 0, 0\rangle\rangle\rangle) \Rightarrow mark(M^{min(m,n)}, \langle 0, 0\rangle)$

- $(M^0) \rightarrow_{l\langle 1,0\rangle} \bot$. Then
  $mark(M^0, \langle 0, 0\rangle) \vee\ mark(M^0, \langle 1, \langle 0, 0\rangle\rangle) \Rightarrow mark(\bot, \langle 0, 0\rangle)$

- $\bot^n \rightarrow_{l\langle 1,0\rangle} \bot$. Then
  $mark(\bot^n, \langle 0, 0\rangle) \vee mark(\bot^n, \langle 1, \langle 0. 0\rangle\rangle) \Rightarrow mark(\bot, \langle 0, 0\rangle)$

- $\bot N \rightarrow_{l\langle 1,0\rangle} \bot$. Then
  $mark(\bot N, \langle 1, \langle 0, 0\rangle\rangle) \Rightarrow mark(\bot, \langle 0, 0\rangle)$

- $(\lambda\langle x_1 \ldots x_k\rangle.M)N \rightarrow_{l\langle 1,i\rangle} \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M \wedge\ 1 \leq i \leq k$. Then,
  $mark((\lambda\langle x_1 \ldots x_k\rangle.M)N, \langle 1, \langle 0, 0\rangle\rangle) \Rightarrow mark(P, \sigma)$, if $F_P(\sigma)$ is defined
  where $P = \lambda\langle x_1 \ldots x_{i-1}, x_{i+1} \ldots x_k\rangle.[x_i \mapsto N]M$

- $(\lambda\langle x_1 \dots x_k\rangle.M)^{(n+1)}N \rightarrow_{l\langle 1,i\rangle} \lambda\langle x_1 \dots x_{i-1}, x_{i+1} \dots x_k\rangle.[x_i \mapsto N^n]M \wedge \; 1 \leq i \leq k$. Then,
  $mark((\lambda\langle x_1 \dots x_k\rangle.M)^{(n+1)}N, \langle 1, \langle 0,0\rangle\rangle) \vee mark((\lambda\langle x_1 \dots x_k\rangle.M)^{(n+1)}N, \langle 1, \langle 1, \langle 0,0\rangle\rangle\rangle) \Rightarrow$ $mark(P,\sigma)$, if $F_P(\sigma)$ is defined
  where $P = \lambda\langle x_1 \dots x_{i-1}, x_{i+1} \dots x_k\rangle.[x_i \mapsto N^n]M$

Marking information is extended to the whole term by structural induction.

- $M = NP$. Then,
  - $mark(N,\sigma) \Rightarrow mark(MN, \langle 1, \sigma\rangle)$
  - $mark(P,\sigma) \Rightarrow mark(MN, \langle 2, \sigma\rangle)$
- $M = M_1|M_2$. Let $len(M) = s$, $1 \leq i \leq s$. Then,
  $mark(index(i,M),\sigma) \Rightarrow mark(M, \langle i, \sigma\rangle)$

Lemma 48. Let $M \in Terms_2 \wedge \; len(M) = k \wedge \; M \overset{*}{\rightarrow}_l P$. Then

- $P \in Terms_2$
- Let $len(P) = s$. Then,
  $(\exists i_1 \dots i_{k+1})\; [(1 \leq i_1 \leq i_2 \dots \leq i_k \leq i_{k+1} = s+1) \wedge (i_1 \neq i_2 \wedge \; \dots i_{k-1} \neq i_k \wedge \; i_k \neq i_{k+1}$
  such that
  $(\exists P_1 \dots P_k)\; (\forall j = 1 \dots k)\; [index(j,M) \overset{*}{\rightarrow}_l P_j \wedge \; P_j \in Terms_2]$, and
  - $(\forall j = 1 \dots k)\; [len(P_j) = i_{j+1} - i_j]$
  - $(\forall j = 1 \dots k)\; (\forall t = 1 \dots len(P_j))\; [index(t, P_j) = index(i_j + t - 1, P)]$

Proof: Induction on the length of reduction $M \overset{*}{\rightarrow}_l P$. ∎
Let $N \in BC^\omega\bot$. Then, $M^\star$ is notation for $[x \mapsto N]M$.

Lemma 49. Let $M \in Terms_2 \wedge \; len(M) = k$. Then

- $M^\star \in Terms_2$
- Let $len(M) = k$, $len(M^\star) = s$. Then,
  $(\exists i_1 \dots i_{k+1})\; [(1 \leq i_1 \leq i_2 \dots \leq i_k \leq i_{k+1} = s+1) \wedge (i_1 \neq i_2 \wedge \; \dots i_{k-1} \neq i_k \wedge i_k \neq i_{k+1})$
  such that
  $(\exists P_1 \dots P_k)\; (\forall j = 1 \dots k)\; [(index(j,M))^\star = P_j \wedge \; P_j \in Terms_2]$, and
  - $(\forall j = 1 \dots k)\; [len(P_j) = i_{j+1} - i_j]$
  - $(\forall j = 1 \dots k)\; (\forall t = 1 \dots len(P_j))\; [index(t, P_j) = index(i_j + t - 1, M^\star)]$

Proof: Structural induction. ∎

Lemma 50. Let $M \in Terms_2 \wedge \; len(M) = k \wedge M^\star \overset{*}{\rightarrow}_l P$. Then

- $P \in Terms_2$
- Let $len(P) = s$. Then,
  $(\exists i_1 \ldots i_{k+1})\ [(1 \leq i_1 \leq i_2 \ldots \leq i_k \leq i_{k+1} = s+1) \wedge (i_1 \neq i_2 \wedge\ \ldots i_{k-1} \neq i_k \wedge i_k \neq i_{k+1})$
  such that
  $(\exists P_1 \ldots P_k)\ (\forall j = 1 \ldots k)\ [(index(j, M))^\star \overset{*}{\rightarrow}_l P_j \wedge\ P_j \in Terms_2]$, and
  - $(\forall j = 1 \ldots k)\ [len(P_j) = i_{j+1} - i_j]$
  - $(\forall j = 1 \ldots k)\ (\forall t = 1 \ldots len(P_j))\ [index(t, P_j) = index(i_j + t - 1, M^\star)]$

Proof: From lemma 48 and lemma 49. ∎

Let $M \in Terms_2 \wedge\ len(M) = k \wedge M^\star \overset{*}{\rightarrow}_l P$. From above lemma, under the hypothesis of the above lemma, there is a well defined function
$f : \{1 \ldots len(P)\} \rightarrow len(M)$ such that

- $(index(f(s), M)) \overset{*}{\rightarrow}_l P_{f(s)} \wedge [(\exists 1 \leq t \leq len(P_{f(s)}))\ [index(t, P_{f(s)}) = index(s, P)]]$
- $i_{f(s)} \leq s \leq (i_{f(s)} + len(P_{f(s)}) - 1)$

The following special class of terms plays an important role in the proof of the strong normalisation theorem. Define, by mutual recursion:
$T_1 ::= x \mid\mid T_2 N \mid\mid M^n$
$T_2 ::= T_2|P \mid\mid P|T_2 \mid\mid T_1$
where $P \in BC^\omega\bot$

Consider $M^\star = [x \mapsto N]M$. Let $M^\star \rightarrow_l P$. For the proofs, we need to trace the terms of P that arise from the N's that have been substituted for x. This is done using the marking machinery that has been set up earlier. The initial marking on $M^\star$ is defined by structural induction. Without loss of generality, assume that x does not occur as the variable bound in an abstraction.

- $R = [x \mapsto N]x$. Then,
  $F_R(\sigma)$ defined $\Rightarrow mark(R, \sigma)$
- $R = [x \mapsto N](R_1 R_2)$. Then,
  - $mark([x \mapsto N]R_1, \sigma) \Rightarrow mark([x \mapsto N](R_1 R_2), \langle 1, \sigma\rangle)$
  - $mark([x \mapsto N]R_2, \sigma) \Rightarrow mark([x \mapsto N](R_1 R_2), \langle 2, \sigma\rangle)$
- $M = [x \mapsto N](M_1|M_2)$. Let $len(M) = s$, $1 \leq t \leq s$. Let $i_j \leq t \leq i_{j+1} \wedge\ t \neq i_{j+1}$, where the i_j's are the indices given by lemma 49. Then,
  - $i_j \leq len(M_1) \wedge mark([x \mapsto N]index(t, M_1), \sigma) \Rightarrow mark(M, \langle t, \sigma\rangle)$
  - $len(M_1) \leq i_j \wedge i_j \neq len(M_1) \wedge mark([x \mapsto N]index(t - len(M_1), M_2), \sigma)$
    $\Rightarrow mark(M, \langle t, \sigma\rangle)$

The following lemmas are analogous to lemma 14.1.8 to 14.1.10 of Barendregt's book on the lambda calculus [3]. Restricted to pure lambda-terms, the statements are identical to the corresponding lemmas for the labelled lambda calculus. The proofs are quite similar to the corresponding proofs for the labelled lambda calculus and are omitted, with $T(M)$ playing the role of $d(M)$ in the proofs in the case of the labelled lambda calculus.

Lemma 51. Let $M \in SN$. Let $M^\star \overset{*}{\rightarrow}_l R$, where $R \in Terms_2 \wedge index(s, R) = \lambda\langle y_1 \dots y_k\rangle.P$, for some $s, k \in \omega$. Then,

- $(\exists R_1 \in Terms_2)\ [M \overset{*}{\rightarrow}_l R_1 \wedge R_1^\star \overset{*}{\rightarrow}_l R]$
- Let $f(s) = k$. Then, one of the following hold.
  - $index(k, R) = \lambda\langle y_1 \dots y_k\rangle.P_1 \ \wedge P_1^\star \overset{*}{\rightarrow}_l P$
  - $index(k, R) = T$, where $T \in T_2$. Then, $mark(R, \langle s, \langle 0, 0\rangle\rangle)$, where $mark(T^\star,)$ is defined as above.

Lemma 52. Let $M \in BC^\omega\bot$. Let $mark(M,)$ be defined such that, $mark(M, \sigma)$ is true for exactly one σ, where $F_M(\sigma)$ is defined. Let $M \overset{*}{\rightarrow}_l R$, where $R \in Terms_2$. Let $index(t, R) = (\lambda\langle y_1 \dots y_k\rangle.P)^n$. Furthermore, let $mark(R, \langle t, \langle 0, 0\rangle\rangle)$ be true, where $mark(R,)$ is defined by using induction on the definition 17. Then, $n \leq label(F_M(\sigma))$.

Lemma 53. $M \in SN \Rightarrow [x \mapsto \bot]M \in SN$

Lemma 54. $M, N \in SN \Rightarrow [x \mapsto N]M \in SN$

Theorem 4. (Strong normalisation)
Every reduction starting from a completely labelled term (M, I) terminates.

References

[1] S. Abramsky. Unpublished lecture at MFPS, 1989.

[2] S. Abramsky and C. H. L. Ong. Full abstraction in the lazy lambda calculus. *Submitted to Information and Computation*, 1988.

[3] H. P. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*. Studies in Logic. North-Holland, Amsterdam, revised edition edition, 1984.

[4] G. Boudol. Towards a lambda-calculus for concurrent and communicating systems. In J. Diaz, editor, *TAPSOFT 89, Lecture Notes in Computer Science 351*, pages 149–161. Springer-Verlag, 1989.

[5] R. deNicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1983.

[6] U. Engberg and M. Neilsen. A calculus of communicating systems with label passing. DAIMI PB-208, Aarhus University, 1986.

[7] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[8] R. Jagadeesan and P. Panangaden. A domain-theoretic model for a higher-order process calculus. Technical report, Cornell, 1989.

[9] B. Jonsson. A fully abstract trace model for dataflow networks. In *Proceedings of the Sixteenth Annual ACM Symposium On Principles Of Programming Languages*, 1989.

[10] G. Kahn. The semantics of a simple language for parallel programming. In *Information Processing 74*, pages 993–998. North-Holland, 1977.

[11] J. Kok. A fully abstract semantics for dataflow nets. In *Proceedings of Parallel Architectures And Languages Europe 1987*, pages 351–368, Berlin, 1987. Springer-Verlag.

[12] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

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

[14] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[15] F. Neilson. The typed lambda-calculus with first-class processes. In *PARLE89, Lecture Notes in Computer Science 366*, pages 357–373, 1989.

[16] C. H. L. Ong. *The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming.* PhD thesis. Imperial College of Science and Technology, 1988.

[17] G. D. Plotkin. Lecture notes on domain theory. The Pisa Notes.

[18] G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976.

[19] J. H. Reppy. Synchronous operations as first-class values. In *Procedings of the SIGPLAN conference on Programming languaage design and implementation*, 1988.

[20] J. R. Russell. Full abstraction for nondeterministic dataflow networks. In *Proceedings of the 30th Annual Symposium of Foundations of Computer Science*, pages 170–177, 1989.

[21] M. B. Smyth and G. D. Plotkin. The category theoretic solution of recursive domain equations. *Siam Journal of Computing*, 11(4), 1982.

[22] B. Thomsen. A calculus of higher-order communicating systems. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Programming Languages*, 1988.

[23] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞ models of the λ-calculus. *SIAM J. Computing*, 5:488–521, 76.

[24] D. J. Walker. Bisimulation and divergence. In *Third IEEE Symposium on Logic in Computer Science*, pages 186–192, 1988.