Laboratory for Foundations of Computer Science
Department of Computer Science - The University of Edinburgh

The Lazy Lambda Calculus in a Concurrency Scenario

The Lazy Lambda Calculus in a Concurrency Scenario

by

Davide Sangiorgi

LFCS Report Series

ECS-LFCS-91-189

November 1991

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

The Lazy Lambda Calculus in a Concurrency Scenario

Davide Sangiorgi
LFCS - Dep. Computer Science
Edinburgh University, Edinburgh EH9 3JZ, U.K.

Abstract

The use of lambda calculus in richer settings, possibly involving parallelism, is examined in terms of its effect on the equivalence between lambda terms. We concentrate here on Abramsky's lazy lambda calculus and we follow two directions. First, the lambda calculus is studied within a process calculus by examining the equivalence $\stackrel{\sim}{\leftrightarrow}$ induced by Milner's encoding into the π-calculus. We give exact operational and denotational characterizations for $\stackrel{\sim}{\leftrightarrow}$. Secondly, we examine Abramsky's applicative bisimulation when the lambda calculus is augmented with (well-formed) operators, i.e. symbols equipped with reduction rules describing their behaviour. Then, maximal discrimination is obtained when all operators are considered; we prove that this discrimination coincides with the one given by $\stackrel{\sim}{\leftrightarrow}$ and that the adoption of certain non-deterministic operators is sufficient and necessary to induce it. We conclude that the introduction of non-determinism into the lambda calculus is exactly what makes applicative bisimulation appropriate to reason about the functional terms when also concurrent features are present in the language or when embedded into a concurrent language.

1 Introduction

The lambda calculus is the commonly accepted basis for functional programming. The syntax is simple, and based on the operators of abstraction and application. Then an evaluation mechanism defines the operational meaning of a term built out of this syntax. We will concentrate here on Abramsky's ideas [1]. His lazy lambda calculus is proposed as a basis for the (lazy) functional programming languages and the evaluation mechanism is guided by what the implementation of such languages suggest; in particular reductions are forbidden within a lambda abstraction. In such a setting, termination means reduction to an abstraction and is the only observable property. Then Abramsky decrees two closed lambda terms applicative bisimilar, written $\simeq$, if either both or neither of them terminates and recursively, this property is maintained for any input provided by the external observer.

In [2] Abramsky develops a theory for applicative bisimulation which runs in parallel with his treatment of concurrency. The definition itself of $\simeq$ inherits the bisimulation idea originally developed for concurrency [20], [14]. It has also an alternative characterization which is reminiscent of the testing equivalence idea [9]; it says that two terms are equivalent when they induce the same termination property in all the pure lambda contexts:

$$M \simeq N \text{ iff for all contexts } C,\ C(M)\ \textit{terminates} \Leftrightarrow C(N)\ \textit{terminates} \quad (*)$$

However, so defined, applicative bisimulation carries some problems. It is based on the notion of termination, but you cannot "speak" in the pure lambda calculus about termination. For instance, you cannot define the convergency test (see section 2). Such an operator is used to show that Abramsky's canonical domain for the lazy lambda calculus and Milner's encoding into the π-calculus [15, 13] do not induce fully abstract models. Now, what does this mean? Maybe Abramsky's canonical domain and Milner's encoding are not good enough for the lazy lambda calculus; but maybe the problem is just that the pure lambda calculus is too weak versus the predicate of termination. Moreover, since equivalence notions developed for frameworks of reactive and concurrent systems are used, one might consider restrictive, for instance, to prevent any "parallel" operator in the contexts of the definition $(*)$.

Various enrichments of the lazy lambda calculus with operators not lambda definable have already appeared in the literature. However either the operators themselves — as in the case of convergency test and parallel convergency in [3, 17, 18] — or the semantics chosen — as for the non-deterministic choice and parallel operator in [6] and [7] — are rather ad hoc, chosen to achieve full abstraction for some canonical domain. Or at least, from a programming language point of view, they do not seem to be justified by any common practice. Moreover it remains unclear whether the equivalences induced are insensitive to the adoption of more operators. This is indeed essential when considering the integration of functional and concurrent calculi: In order to reason effectively, we would like, for instance, to know when two functional terms can be exchanged without affecting the behaviour of the process in which they are used.

The above discussion intended to point out the interest for the study of the lazy lambda calculus in "richer" settings, the focus of the paper. We have pursued two approaches: in the first one the lambda calculus is studied *within* a process calculus, hence in some sense making complete the immersion of the lazy lambda calculus into concurrency which Abramsky had started. Because the lazy lambda calculus reflects the language implementation philosophy, a "powerful" process calculus should make possible a simple encoding. We have chosen the π-calculus for this because it responds to our requirements and because a nice encoding already exists, namely Milner's encoding. We have called *lambda observational equivalence* the equivalence induced by the embedding, that is two lambda terms are identified if their process-encodings are (weak) bisimilar. It has to be stressed that the study of Milner's encodings into the π-calculus was in fact a major concern in our work. If the lambda calculus is universally accepted as the calculus to reason about sequential programs and systems, the π-calculus aims at being its counterpart for the parallel ones. This makes the comparison between the two calculi something worth looking at. In this direction, Milner's encoding of the lazy lambda calculus, because of its simplicity and canonicity, represents an interesting starting point.

The other approach arises the issue of a systematic study of the lazy lambda calculus and applicative bisimulation in presence of a richer class of operators than those definable within the lambda calculus. Notice that here we do not go through the embedding into an auxiliary language but we simply enrich the pure lambda calculus. We will admit only *well-formed* operators. Intuitively, an operator is well-formed if its behaviour depends only on the semantics — not on the syntax — of its operands. Groote and Vaandrager (see [10]) have studied the meaning of well-formed operators and transition systems in a CCS-like setting. Their format for the behavioural rules of the operators has been adapted to the lambda calculus. Then the most discriminating congruence is the one obtained when all well-formed operators are admitted; we have called it *rich applicative congruence.*

We have proved that lambda observational equivalence and rich applicative congruence represent indeed the same equivalence on the pure lambda terms. Put in other words, the encoding into the π-calculus induces the maximal observational discrimination on the lambda terms. A very interesting problem is then to find a minimal set of operators giving the same discrimination. This means to understand *what it is necessary to add to the lambda calculus to make it as discriminating as the π-calculus.* Would it be the case, for instance, that the parallel convergency operator is the solution to this problem in the same way as it was the solution to the full abstraction problem for Abramsky's canonical model [2, 3]? The answer is no. The parallel convergency test can be expressed as a deterministic operator and one of our results is that it is not possible to obtain maximal discriminatory power using only deterministic operators. Instead, the right answer is non-determinism. In fact we have proved that one of the simplest form of non-determinism one could think of, an unary operator which when applied to some argument either behaves like the argument or diverges, is enough to do the job. All this gives a significant measure of the power which non-determinism can induce. Let us mention at this point some interesting domain-theoretical work on a non-deterministic (quasi-lazy) lambda calculus, currently carried out by Ong [19].

It has been crucial for the development of our work an alternative interpretation for the constants. The standard way to treat a constant is to introduce it together with some rules describing its operational behaviour. In the sequel we will call these *operators.* When only operators are used, lambda abstraction remains the only sensible normal form for closed terms. We keep the word *constant* to denote instead symbols which are added to the language without specifying any operational rule. Such a use of the constants can be found in the well known technique of the *top down specification and analysis*, where a system is developed through a series of refinement steps each representing a different level of abstraction; a lower level implements some details which at a higher level have left abstracted. A constant c is then an high level primitive standing for some lower level procedure K_c; at this stage we might want to explicitly abstract from the behaviour of K_c to facilitate the reasoning, or it might just be that we cannot make assumptions on the behaviour of K_c (for instance, we might be interested in refinments of c with different K_c's). Now, $c\tilde{M}$ becomes a sensible normal form too. Operationally we really can see it as the *output* of the tuple $\tilde{M}$ along the channel c and towards K_c. The definition of applicative bisimulation has been generalized according to such interpretation to the case where also constants are admitted.

Two interesting characterizations for lambda observational equivalence can be obtained using only constants and no operators. One of them looks promising for the mechanization of the equivalence checking. A further characterization is possible on the open terms, where no operator or constant is used; it represents about the simplest way one could think of to consider the notion of applicative bisimulation on open terms. We will prove full abstraction results for it in terms of Longo Trees and free lazy Plotkin-Scott-Engeler models. They represent the specialization to the lazy regime of, respectively, the Böhm Trees and the class of lambda models called Plotkin-Scott-Engeler models ([12, 17]).

Having introduced a number of equivalences, usually defined on enriched, distinct lambda languages, and being willing to compare the discriminatory power induced on the common core of the pure lambda calculus, we have explicitly introduced an ordering on equivalences based on it. Then an equivalence is e_1 is preceded by an equivalence

e_2 if any pair of pure closed lambda terms which are equated by e_1 are also equated by e_2. Particularly interesting is the ordering that one gets by restricting to applicative bisimulation equivalences defined on lambda calculus languages enriched with only operators. Not much is known about the resulting preorder, which looks like inducing a semilattice. Clearly there is a maximal element represented by Abramsky's original $\simeq$ and a minimal one represented by the rich applicative congruence. Further informations can be deduced from our work. However the right relationship among various interesting equivalences has still to be revealed.

We will not introduce here the π-calculus nor Milner's encoding of the lazy lambda calculus, since we use them only as our starting point and will never be performing manipulations of π-calculus processes. We refer to [15] and [13] for a detailed exposition.

STRUCTURE OF THE PAPER. Section 2 introduces some necessary notation for the lazy lambda calculus. In section 3 applicative bisimulation is generalized over lambda calculus languages enriched with a set of operators P and constants C; we have denoted it by $\simeq_C^P$ ($\simeq_C$ when the set P is empty). The preorder $<$ over lambda calculus equivalences is defined. The remainder of the paper is conceptually divided into two parts. In the first one, including sections 4 and 5, the equivalence induced on the lambda terms by the encoding into the π-calculus — written $\tilde{\leftrightarrow}$ — is studied operationally and denotationally. In the second one, including sections 6, 7 and 8, enrichments of the lazy lambda calculus with operators are examined. Various results concerning the discriminatory power offered by different sets of operators are presented but the real target here is to use operators to describe the discriminatory power given by $\tilde{\leftrightarrow}$.

In section 4 $\tilde{\leftrightarrow}$ is studied *operationally*; we begin with its characterization in term of $\simeq_\Psi$ given in [22, 24], where Ψ is a set containing an infinite number of constants; we prove that this result is actually independent of the set of constants used, as long as it is nonempty; a simple direct proof (which does not exploit the encoding into the π-calculus) of the congruence of $\simeq_C$ is presented; we finish with two further useful characterizations of $\tilde{\leftrightarrow}$. In section 5 $\tilde{\leftrightarrow}$ is studied *denotationally*; full abstraction results in terms of Longo Trees and free lazy Plotkin-Scott-Engeler models are presented. In section 6 the well-formed operators are formally introduced by describing and motivating the format which the rules defining them must obey; we show that $\tilde{\leftrightarrow}$ is at least as fine as rich applicative congruence. In section 7 the opposite containment is established by using a simple non-deterministic operator. In section 8 we prove that for such a result non-determinism is actually *necessary*, i.e. that the same discriminatory power cannot be recovered using only deterministic operators. In section 9 we report some conclusions and, as future work, some questions which remain to be examined.

2 Preliminary Notation and Definition

We will use $x, y, ...$ to range over variables; $c, d, ...$ and C over constants and sets of constants; $p, ...$ and P over operators and sets of operators. We assume that each operator p has an arity $r(p)$ representing the number of arguments which p needs.

The class of the $\Lambda_C^P(X)$ terms, i.e. lambda terms enriched with operators in P and constants in C is defined by the following grammar

$$M = c \;\|\; pM_1 \ldots M_{r(p)} \;\|\; x \;\|\; \lambda x.M \;\|\; M_1 M_2, \qquad \text{where } c \in C \text{ and } p \in P.$$

The definitions of free variables, closed terms, substitutions, α conversion etc. are taken to be the standard ones. Throughout the paper we will assume that *all α convertible terms are completely identified* and we will write $M = N$ if M and N are α-convertible. The subclass of $\Lambda_C^P(X)$ made of the closed terms is denoted by Λ_C^P. When P or C is empty, we will feel free to omit the corresponding index. So for instance, Λ and Λ_C denote respectively the class of the closed pure lambda terms and the class of the closed lambda terms enriched with constants from C.

As always, I and Ω represent, respectively, the terms $\lambda x.x$ and $(\lambda x.xx)(\lambda x.xx)$. We will use M, N, R, T to range over (enriched) lambda terms; also, we use $\tilde{M}$ in $N\tilde{M}$ stands for a sequence of arguments, i.e. n and M_1, ..., M_n exist such that $N\tilde{M} = NM_1..M_n$. We will denote the set of constants and the set of free variables appearing in the term M respectively by $ct(M)$ and $fv(M)$.

We now turn to the definition of the binary reduction relation $\Rightarrow \subseteq \Lambda_C^P(X) \times \Lambda_C^P(X)$. First of all we have the rules (β), (App), the core rules for the pure lazy lambda calculus and the rules $Refl$ and $Trans$, describing the reflexive and transitive nature of $\Rightarrow$.

$$(\beta)\ (\lambda x.M)N \Rightarrow M\{N/x\} \qquad\qquad (App)\ \frac{M \Rightarrow M'}{MN \Rightarrow M'N}$$

$$(Refl)\ M \Rightarrow M \qquad\qquad (Trans)\ \frac{M \Rightarrow M' \qquad M' \Rightarrow M''}{M \Rightarrow M''}$$

Besides these, we need also a set of rules for each operator to describe its operational meaning. We will call such rules *behavioural rules*. We will only admit rules in *tfyt* format ensuring — without really loosing expressiveness — that the behaviour of the operators they define, called *well-formed* operators, depends only on the semantics and not on the syntax of their operands. Such format will be described in detail in section 6. For the moment, let us just show as example, the rules for the operators ∇ (*convergency test*), $\Diamond$ (*parallel convergency test*) and $\oplus$ (*non-deterministic choice*). Here $M \Downarrow$ is just used as an abbreviation to mean that either $M \Rightarrow \lambda x.M'$ or $M \Rightarrow c\tilde{M}'$, for some c, M', $\tilde{M}'$.

$$(\nabla 1)\ \frac{M \Downarrow}{\nabla M \Rightarrow I} \qquad\qquad (\nabla 2)\ \frac{M \Rightarrow M'}{\nabla M \Rightarrow \nabla M'}$$

$$(\Diamond 1)\ \frac{M \Downarrow}{\Diamond MN \Rightarrow I} \qquad (\Diamond 2)\ \frac{M \Downarrow}{\Diamond NM \Rightarrow I} \qquad (\Diamond 3)\ \frac{M \Rightarrow M' \qquad N \Rightarrow N'}{\Diamond MN \Rightarrow M'N'}$$

$$(\oplus 1)\ \oplus MN \Rightarrow M \qquad\qquad (\oplus 2)\ \oplus MN \Rightarrow N$$

It is useful to have also the "one"-step reduction relation $\rightarrow$. It is obtained from $\Rightarrow$ by dropping the rules $Refl$ and $Trans$ and by replacing $\Rightarrow$ with $\rightarrow$ in the conclusion of the remaining rules. Notice that both $\Rightarrow$ and $\rightarrow$ have been defined on the open lambda terms. We will make use of this in definition 8 and in section 5.

Let us also give a name to the sets containing, respectively, all the operators and all the constants one might want; we will use Op to denote the set of all the well-formed operators, Ψ to denote the set $\{c_1, \ldots, c_n, \ldots\}$ containing an infinite, countable number of constants.

At this point it seems worth to summarize the notations for the equivalences and for the operators or sets of operators that will be introduced and used throughout the paper. In the entry for an equivalence, the central column represents its domain.

$\stackrel{\sim}{\leftrightarrow}$	$\Lambda(X) \times \Lambda(X)$	equivalence induced by the π-calculus encoding
$\simeq_C^P$	$\Lambda_C^P \times \Lambda_C^P$	applicative bisimulation over Λ_C^P
$\cong_C^P$	$\Lambda_C^P \times \Lambda_C^P$	congruence induced by $\simeq_C^P$
$\simeq'_\Psi$	$\Lambda_\Psi \times \Lambda_\Psi$	simpler characterization for $\simeq'_\Psi$
$\stackrel{o}{\simeq}$	$\Lambda(X) \times \Lambda(X)$	applicative bisimulation over open terms

∇	convergency test
$\Diamond$	parallel convergency
$\oplus$	non-deterministic choice
$\uplus$	$= \oplus\Omega$
Op	set containing all well-formed operators
DO	set containing all deterministic well-formed operators
Ψ	set containing an infinite countable number of constants

3 Applicative Bisimulation over Λ_C^P

We are now ready to define the applicative bisimulation for the terms in Λ_C^P. We will have to generalize appropriately the definition of applicative bisimulation given in [1] in order to be able to deal also with operators and constants. The main question is which condition should be imposed on the equality between the terms $c\tilde{M}$ and $c\tilde{N}$. According to our interpretation of the constants given in the introduction the most natural thing to do seems to require that the *ordered* sequence of arguments represented by $\tilde{M}$ and $\tilde{N}$ be equivalent; hence the clause (2) in the following definition.

Definition 1 *A symmetric relation $\Re \subseteq \Lambda_C^P \times \Lambda_C^P$ is a $\simeq_C^P$-bisimulation, if $(M, N) \in \Re$ implies:*

1. *whenever $M \Rightarrow \lambda x.M'$ then $\exists N' \,.\, N \Rightarrow \lambda x.N'$ and for all $R \in \Lambda_C^P$ it holds that $(M'\{R/x\}, N'\{R/x\}) \in \Re$,*
2. *whenever $M \Rightarrow cM_1 \ldots M_n$, for some $n \geq 0$ and $c \in C$, then $\exists N_1, \ldots N_n \,.\, N \Rightarrow cN_1 \ldots N_n$ and $(M_i, N_i) \in \Re$, $1 \leq i \leq n$,*
3. *whenever $M \Rightarrow M'$ then $\exists N'.N \Rightarrow N'$ and $(M', N') \in \Re$.*

M and N are applicative bisimilar over Λ_C^P, *written $M \simeq_C^P N$, if $(M, N) \in \Re$, for some $\simeq_C^P$-bisimulation $\Re$.* □

It is easy to see that $\simeq_C^P$ induces an equivalence relation over Λ_C^P. When P contains only one element, say p, we will simply write $\simeq_C^p$. We will drop the index P or C in $\simeq_C^P$ when the corresponding set is empty; then, the applicative bisimulation over Λ will be denoted by $\simeq$ and represents Abramsky's original definition. We will call it simply *applicative bisimulation.* Also, we will call $\simeq^{Op}$ *rich applicative bisimulation* (here the whole class of well-formed operators is considered).

Notice that clause (3) was not present in Abramsky's original definition. This is because the pure lambda calculus is deterministic, and if $M \Rightarrow M'$, then $M \simeq M'$. This in general is no longer true in Λ_C^P, hence clause (3) is required. For instance, it is this clause which discriminates between I and $I \oplus \Omega$. In fact these terms are considered equivalent by Boudol in [6] where clause (3) is ignored. However it seems natural to maintain the distinction between I and $I \oplus \Omega$ since the former would always accept an input whereas the latter can also refuse it.

As an aside let us just point out that if we added some labels to our reduction system and we wrote $M \overset{\lambda x}{\Rightarrow} M'$ instead of $M \Rightarrow \lambda x.M'$, and $M \overset{c\tilde{M}'}{\Rightarrow}$ instead of $M \Rightarrow c\tilde{M}'$, then our definition for $\simeq_C^P$ would become just a particular instance of the higher order bisimulation [21, 4].

<u>CONGRUENCE</u>. Although we conjecture it is true, we were not able to prove a general congruence result for $\simeq_C^P$, therefore forcing the introduction of a different notation. However we do prove congruence results for the cases which turn out to be interesting to us (see theorem 4, lemma 3, corollary 3).

Definition 2 *A* Λ_C^P-context *is a* $\Lambda_C^P(X)$ *term with a single variable free in it.* □

If C is a context, we will write $C(M)$ to denote $C\{M/x\}$, where x is the variable which is free in C.

Definition 3 Applicative congruence over Λ_C^P, *written* $\cong_C^P$, *is the largest relation over* $\Lambda_C^P \times \Lambda_C^P$ *such that* $M \cong_C^P N$ *implies* $C(M) \simeq_C^P C(N)$ *for every* Λ_C^P*-context* C. □

An interesting congruence is $\cong^{Op}$, which we call *rich applicative congruence.* It represents the most discriminating observation-based equivalence obtained by using only operators.

<u>THE PREORDER ON THE EQUIVALENCES</u>. In the sequel we will consider various equivalences defined on lambda calculus terms possibly enriched with constants and operators. We will be interested then in comparing their discriminatory power. Since the classes of terms on which they are defined in general do not coincide, we will compare them by examining the equivalence induced on the closed pure lambda terms.

Definition 4 *Let* $\sim$, $\sim'$, *be two equivalences defined on some enriched lambda language* Λ_C^P, $\Lambda_{C'}^{P'}$, *respectively. We write* $\sim < \sim'$ *if for each* $M, N \in \Lambda$, $M \sim N$ *implies* $M \sim' N$ *(i.e.* $\sim$ *is a finer relation); and* $\sim <> \sim'$ *if* $\sim < \sim'$ *and* $\sim' < \sim$. □

From the definition of $\simeq_C^P$, it follows immediately that

Theorem 1 *If* $P \subseteq P'$ *and* $C \subseteq C'$, *then* $\simeq_{C'}^{P'} < \simeq_C^P$.

4 The equivalence induced by Milner's encoding into the π-calculus

Our starting point is the equivalence $\stackrel{\sim}{\leftrightarrow}$ induced on the lambda terms by Milner's encoding into the π-calculus and its characterization in terms of $\simeq_\Psi$ proved in [22, 24]. Let $[M]$ denote the encoding of the term M and $\approx$ the observational equivalence for the π-calculus. To be precise, according to the terminology in [15] we should also say which version of the observational equivalence we mean, if the late or the early, ground or not ground. We do not because on the encoding of the closed lambda terms all such versions coincide.

Definition 5 *Let $M, N \in \Lambda(X)$; then $M \stackrel{\sim}{\leftrightarrow} N$ if $[M] \approx [N]$. In such case we say that M, N are* lambda observational equivalent. □

Milner started the study of $\stackrel{\sim}{\leftrightarrow}$ in [13]. He showed that $M \stackrel{\sim}{\leftrightarrow} N$ implies $M \simeq N$ and that the converse is not true (for the latter the terms M, N in example 2 were used). The characterization of $\stackrel{\sim}{\leftrightarrow}$ was left as an open problem. This target was achieved in [22, 24], where Milner's encoding was extended to an encoding for $\Lambda_\Psi(X)$ to obtain the following full abstraction result. It was proved by exploiting a more abstract encoding into *Higher Order π-calculus* ([23, 22]), an enrichment of the π-calculus with higher order communications.

Theorem 2 *For $M, N \in \Lambda_\Psi$, it holds that $[M] \approx [N]$ iff $M \simeq_\Psi N$.* □

4.1 Congruence for $\simeq_C$

By appealing to the encoding and to the properties of the π-calculus processes, we would get for free the congruence of $\simeq_\Psi$ over Λ_Ψ. However we thought that it was important as a test for $\simeq_\Psi$ to show that a nice direct proof is possible. Such proof becomes a simple application of the technique proposed by Stoughton to show the congruence of $\simeq$ over Λ (see [3]). With respect to Stoughton's original proof, here we need also the notion of "bisimulation up to".

For generality, we give the proof in terms of $\simeq_C$. If $\Re$ is a relation, let us write $M \Re N$ if $(M, N) \in \Re$ and $M \Re \simeq_C N$ if N' exists such that $M \Re N' \simeq_C N$.

Definition 6 *$\Re \subseteq \Lambda_C \times \Lambda_C$ is an* applicative bisimulation up to $\simeq_C$ *if $\Re$ is symmetric and $M \Re N$ implies:*

1. *if $M = \lambda x.M'$ then $\exists N' . N \Rightarrow \lambda x.N'$ and $\forall R \in \Lambda_C$, $M'\{R/x\} \Re \simeq_C N'\{R/x\}$),*
2. *if $M = cM_1 ... M_n$, for some $n \geq 0$ and $c \in C$, then $\exists N_1, \dots N_n . N \Rightarrow cN_1 \dots N_n$ and $M_i \Re \simeq_C N_i$, $1 \leq i \leq n$,*
3. *whenever $M \rightarrow M'$ then $\exists N' . N \Rightarrow N'$ and $M' \Re \simeq_C N'$.* □

Theorem 3 *If $\Re$ is an applicative bisimulation up to $\simeq_C$ then $\Re \subseteq \simeq_C$*

Proof. Standard technique for bisimulations up to. □

Lemma 1 *For each $M, N, R \in \Lambda_C$, $M \simeq_C N$ implies $MR \simeq_C NR$*

Proof. Immediate from the definition of $\simeq_C$. $\square$

Corollary 1 *For each sequence $R_1, \ldots R_n$ of Λ_C terms, $M \simeq_C N$ implies $MR_1 \ldots R_k \simeq_C NR_1 \ldots R_k$.* $\square$

Theorem 4 $\simeq_C = \cong_C$.

Proof. We will show that
$\Re = \{(C(M), C(N)) \mid C \text{ is a } \Lambda_C\text{-context and } M \simeq_C N\}$
is an applicative bisimulation up to $\simeq_C$, by induction on the structure of the context C.

1. Suppose C is of the form $(\lambda x.C_1)C_2 \ldots C_n$: then, take $C' = (C_1\{C_2/x\})C_3 \ldots C_n$. We have $C(M) \to C'(M)$, $C(N) \to C'(N)$ and $(C'(M), C'(N)) \in \Re$.
2. Suppose C is of the form $cC_1 \ldots C_n$: immediate
3. Suppose C is of the form $xC_1 \ldots C_n$: define $C' = MC_1 \ldots C_n$. Then C' is either of the form (1) or (2); in both cases we have that $C'(M)$ is matched by $C'(N)$. This is enough because by corollary 1, $C(N) \simeq_C C'(N)$ and because $\Re$ is a $\simeq_C$ bisimulation up to. $\square$

It remains an open problem whether the result can be generalized to $\simeq_C^P$. Unfortunately, Stoughton's technique does not seem to work in this case; nor we have been able to prove it using other techniques.

4.2 Alternative <>-characterizations

Although theorem 2 was expressed in terms of Λ_Ψ, what we are really interested here is the equivalence induced on the pure closed lambda terms. We start our study by looking at other interesting characterizations of such an equivalence.

<u>Independence from the set of constants</u>. Our first result is that the choice of the set of constants is not important, as long as it is nonempty.

Theorem 5 *For any nonempty C and C', it holds that $\simeq_C <> \simeq_{C'}$.*

Proof. It will suffice to show to compare $\simeq_d$, where only the constant d is used, with $\simeq_\Psi$, where a countable infinite number of constants is used, and show that $\simeq_d < \simeq_\Psi$. Let $A_1, \ldots, A_n, \ldots$ be in Λ and all $\not\simeq$ each other. For instance one could define $A_i = \lambda x_1 \ldots x_n.\Omega$. For $M \in \Lambda_\Psi$, let M^+ be the term of Λ_d obtained from M by substituting the term dA_i for the constant c_i. Then one can prove (easy transition induction) that
$\{(M, N) \mid M^+ \simeq_d N^+\} \subseteq \simeq_\Psi$. $\square$

<u>For equivalence checking</u>. The next characterization is quite useful when we actually have to *verify* lambda observational equivalence. It also opens interesting perspectives for its *mechanical* verification. The question seems closely related to the mechanical verification of equivalences in higher order process algebras. See [22, 23], where the bisimulation equivalence proposed for higher order process algebras is shown to have a characterization very close in intent to the following $\simeq'_\Psi$.

Definition 7 *Let $\simeq'_\Psi$ be the largest relation on $\Lambda_\Psi \times \Lambda_\Psi$ such that $M \simeq'_\Psi N$ implies:*

1. *if $M \Rightarrow \lambda x.M'$ then $\exists N'.N \Rightarrow \lambda x.N'$ and for $c \notin ct(MN)$ it holds that $M'\{c/x\} \simeq'_\Psi N'\{c/x\}$,*
2. *if $M \Rightarrow cM_1...M_n$, for some $n \geq 0$ and $c \in \Psi$, then $\exists N_1, \ldots N_n \,.\, N \Rightarrow cN_1 \ldots N_n$ and $M_i \simeq'_\Psi N_i$, $1 \leq i \leq n$.* □

What really makes the difference between the definitions of $\simeq_\Psi$ and $\simeq'_\Psi$ is simplicity of clause (1). While $\simeq_\Psi$ requires to test the equality between $\lambda x.M'$ and $\lambda x.N'$ over *all* the terms in Λ_Ψ thus making the verification problem very troublesome (this is a general inconvenience for $\simeq^P_C$), here one term — a constant — is enough. The moral is that *constants are very powerful*; lemma 2 later will provide further evidence to this. Notice also that since the reduction system for Λ_Ψ is deterministic, we do not have to use clause (3), nor to explicitly require that $\simeq'_\Psi$ is symmetric relation.

Theorem 6 $\simeq_\Psi <> \simeq'_\Psi$

Proof. The proof is immediate, exploiting the encoding into Higher Order π-calculus and its theory.

A proof inside the lambda calculus is also possible; we will only sketch it. We have to show $M\{c/x\} \simeq_\Psi N\{c/x\}$ implies $M\{R/x\} \simeq_\Psi N\{R/x\}$, for any $R \in \Lambda_\Psi$. Suppose, without loss of generality, that $c \notin ct(R)$. Since Ψ contains an infinite number of constants, it is enough to prove that $M\{R/x\} \simeq_{\Psi-\{c\}} N\{R/x\}$. This can be derived by proving a slightly stronger version of lemma 2 (section 6) saying that if P is a set of operators and C a set of constants, then $M \simeq_{P \cup C} N$ implies $M \simeq^P_C N$. The modifications to bring into the proof of lemma 2 are straightforward.

Then one has that $M\{c/x\} \simeq_\Psi N\{c/x\}$ implies $M\{c/x\} \simeq^c_{\Psi-\{c\}} N\{c/x\}$, for any choice for c as an operator; in particular this is true for $c \Rightarrow R$ as the only behavioural rule of c. Finally, it is immediate to verify that with such a definition for c as operator, it is $M'\{c/x\} \simeq^c_{\Psi-\{c\}} N'\{c/x\}$ iff $M'\{R/x\} \simeq^c_{\Psi-\{c\}} N'\{R/x\}$. □

ON THE OPEN TERMS. It is a short step to go from $\simeq'_\Psi$ to a characterization on open terms, where no constants are mentioned at all. The definition looks fairly attractive. It is about the simplest way one could think of to consider the notion of applicative bisimulation on open terms. In the next section we will show that Longo Trees represent a nice and simple fully abstract model for it.

Definition 8 *Let $\overset{\circ}{\simeq}$ be the largest relation on $\Lambda(X) \times \Lambda(X)$ such that $M \overset{\circ}{\simeq} N$ implies*

1. *if $M \Rightarrow \lambda x.M'$, $x \notin fv(M, N)$ then $\exists N' \,.\, N \Rightarrow \lambda x.N'$ and $M' \overset{\circ}{\simeq} N'$*
2. *if $M \Rightarrow xM_1...M_n$, for some $n \geq 0$, then $\exists N_1, \ldots N_n \,.\, N \Rightarrow xN_1 \ldots N_n$ and $M_i \overset{\circ}{\simeq} N_i$, $1 \leq i \leq n$.* □

Theorem 7 $\simeq'_\Psi <> \overset{\circ}{\simeq}$

Proof. Straightforward. $\square$

Milner's encoding was originally defined on the open pure lambda terms. One can show by using theorems 2, 6, 7 that the equivalence it gives rise to — open terms included — is exactly $\stackrel{o}{\simeq}$; that is, $\stackrel{o}{\simeq}$ is not just a characterization of $\stackrel{\sim}{\leftrightarrow}$ on the closed pure lambda terms but is another way to describe the *same* relation. Hence in the following theorem we can put the symbol $=$ instead of the weaker $<>$.

Theorem 8 $\stackrel{o}{\simeq} = \stackrel{\sim}{\leftrightarrow}$.

5 Full Abstraction

We examine in this section the problem of a denotational characterization of $\stackrel{\sim}{\leftrightarrow}$. We will express the full abstraction theorems in terms of $\stackrel{o}{\simeq}$ (see definition 8) because they become particularly effective. Theorem 8 makes them into full abstraction results also for $\stackrel{\sim}{\leftrightarrow}$. We will take the so-called Longo Trees and free lazy Plotkin-Scott-Engeler models. Through the study of $\stackrel{\sim}{\leftrightarrow}$ conducted in the paper, we hope that these results could also help in giving insights and reinforcing the importance of such models.

We are particularly grateful to L. Ong for having directed us to Longo Trees. The terminology, notation, etc., that we will use is mainly from [17].

NOTATION: we will abbreviate $\lambda x_1 \ldots . \lambda x_n . M$, $n \geq 0$ as $\lambda x_1 \ldots x_n . M$ or $\lambda \tilde{x} . M$. We will denote by $\lambda\beta$ the classical (i.e. not lazy) formal theory of the lambda calculus given by the usual axioms α, β and the rules μ, ν, ξ. We use ω to denote the set of nonnegative integers $\{0, 1, \ldots\}$ and $\{\omega\}$ to represent the first ordinal limit. Unless otherwise specified, all the lambda terms in this section are supposed to belong to $\Lambda(X)$.

5.1 Full Abstraction using Longo Trees

Following an initial suggestion by Milner, we tried to figure out whether any relationship existed between $\stackrel{\sim}{\leftrightarrow}$ and some tree representation of the terms.

The most popular tree structure in the lambda calculus are Böhm Trees (briefly BT). However, BT's only correctly expresses the computational content of lambda terms in a "strict" regime, while they fail to do so in the lazy one. Without entering the details of the definition of a BT, (see [5]), let us just make this clearer through an example. Consider the term $\lambda x . \Omega$ and Ω. In a lazy regime we would always distinguish between them. However, since both of them are *insolvable* ([5]), their BT's are identical.

The right notion of tree in the lazy regime is a variant of the BT's called Longo Trees (briefly LT). LT's were introduced by Giuseppe Longo in [12] — where they were simply called trees — developing an original idea by Levy [11]. They were called Longo Trees in [17]. Let us briefly introduce the model.

The order of M expresses the maximum length of the outer sequence of lambda abstractions in a term to which M is β-convertible (it says how high order is). A term of order n has proper order n if after the initial n lambda abstractions it behaves like Ω.

Definition 9 *The* order *of a term is defined inductively as follows:*
i) a term M has order 0, *denoted $M \in O_0$, iff $\neg[\exists N . \lambda\beta \vdash M = \lambda x . N]$;*
ii) a term M has order n+1, *denoted $M \in O_{n+1}$, iff $\exists N \in O_n . \lambda\beta \vdash M = \lambda x . N$;*
iii) a term M has order ω, *denoted $M \in O_\omega$, iff $\forall n, M \notin O_n$.* $\square$

Definition 10 *Similarly we define the* proper order *of a term:*
i) a term M has proper order 0, *denoted $M \in PO_0$, iff*
$M \in O_0$ *and* $\neg[\exists \tilde{N} \,.\, \lambda\beta \vdash M = x\tilde{N}]$;
ii) a term M has proper order n+1, *denoted $M \in PO_{n+1}$, iff*
$\exists N \in PO_n \,.\, \lambda\beta \vdash M = \lambda x.N$;
iii) we decree also that a term M has proper order ω , *denoted $M \in PO_\omega$, iff*
M has order ω. □

Definition 11 *The* Longo Tree *of M, $LT(M)$, is a labelled tree defined inductively as follows:*
1) $LT(M) = \top$ *if* $M \in PO_\omega$
2) $LT(M) = \lambda x_1 \ldots x_n.\bot$ *if* $M \in PO_n$, $n \in \omega$
3) $LT(M) =$

$$\begin{array}{c} \lambda\tilde{x}.y \\ \diagup \quad \diagdown \\ LT(M_1) \ldots LT(M_n) \end{array}$$

if M is solvable and has principal head normal form $\lambda\tilde{x}.yM_1 \ldots M_n$, $n \geq 0$. □

Example 1 *Let* $M = x(\lambda y.y)\Omega z\Xi(\lambda x_1x_2.\Omega)$, *where Ξ is a PO_ω term, for instance* $\Xi = (\lambda x.\lambda y.(xx))(\lambda x.\lambda y.(xx))$. *Then*

$$LT(M) = \begin{array}{c} x \\ \diagup \; \diagup \; | \; \diagdown \; \diagdown \\ \lambda y.y \quad \bot \quad z \quad \top \quad \lambda x_1x_2.\bot \end{array}$$

□

We will not distinguish between α convertible LT's (where α conversion on LT's is defined in the obvious way). Therefore the equality of LT's is considered modulo α-convertibility.

Theorem 9 $M \overset{o}{\simeq} N$ *iff* $LT(M) = LT(N)$

Proof. The following is an alternative characterization for $\overset{o}{\simeq}$:
$M \overset{o}{\simeq} N$ iff

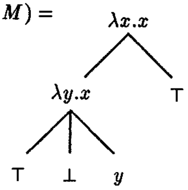
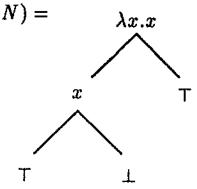
1. $M \in PO_n \Leftrightarrow N \in PO_n$, $n \in \omega \cup \{\omega\}$
2. if $M \Rightarrow \lambda x_1 \ldots x_n.yM_1 \ldots M_m$, for some $m, n \geq 0$ and $x_1, \ldots, x_n \notin n(M, N)$, then $N_1, \ldots, N_m$ exist such that $N \Rightarrow \lambda x_1 \ldots x_n.yN_1 \ldots N_m$ and $M_i \overset{o}{\simeq} N_i$, for $1 \leq i \leq m$.

The above characterization is really a characterization in terms of the LT's. Then the assertion of the theorem follows directly. □

Example 2 *For Ξ as in example 1, consider the terms*

$$M = \lambda x.(x(\lambda y.(x\Xi\Omega y))\Xi) \qquad\qquad N = \lambda x.(x(x\Xi\Omega)\Xi).$$

These terms have been used to prove non full abstraction results by Abramsky-Ong ([3]) and by Milner ([13]). This is due to the fact that both in Abramsky's canonical model and in the π-calculus the convergency test is definable. Such operator can distinguish between them, as $M\nabla$ reduces to an abstraction, whereas $N\nabla$ diverges. However, no pure lambda term can make the same distinction as can be shown by a case analysis on the order of the term. Let us show how we can prove $M \not\approx N$ by simply looking at their Longo Trees. We have that

$LT(M) =$  $LT(N) =$ 

therefore it is $M \not\approx N$, as the two trees are different. □

5.2 Full Abstraction in the Free Lazy PSE models

The characterization of $\overset{\circ}{\simeq}$ given in the proof of theorem 9 reminds us very closely of the *lazy Plotkin-Scott-Engeler preorder* (written $\preceq$) defined by Ong in [17]. It is pretty straightforward to prove in fact that $\overset{\circ}{\simeq} = \preceq \cap \succeq$. Ong uses $\preceq$ to give an exact operational characterization of the class of models called *free lazy Plotkin-Scott-Engeler (PSE) models.* This leads to the following full abstraction result for lambda observational equivalence. Let us write $\Im \vdash M = N$ if $\Im$ is a lambda model identifying the terms M and N.

Theorem 10 *Let $\Im$ be a free lazy PSE model; then $M \overset{\circ}{\simeq} N$ iff $\Im \vdash M = N$.* □

In the following, we review the basic concepts of PSE models. We refer to the nice presentation in [17] for the details.

The family of combinatory algebras called *Plotkin-Scott-Engeler (PSE) Algebra* arises from the study conducted by Engeler in [8] (which followed earlier ideas by Plotkin and Scott). Its theoretical properties were later investigated by Longo in [12]. PSE algebras are defined a very natural set theoretic way and the notion of application generalizes the classical Myhill-Shepherdson-Roger definition of application in $P\omega$.

Definition 12 (Plotkin, Scott, Engeler and Longo)
Let B be a nonempty set such that if β ranges over finite sets,

$$\beta \subseteq B \;\&\; b \in B \text{ iff } <\beta, b> \in B$$

Then $(\wp B, \cdot)$ is called Plotkin-Scott-Engeler (PSE) Algebra, where the application $\_\cdot\_ : \wp B \times \wp B \to \wp B$ is defined as

$$d \cdot e \stackrel{\text{def}}{=} \{b :< \beta, b >\in d \ \& \ \beta \subseteq e\}$$

□

Particularly interesting is the PSE Algebra $(D_A, \cdot)$ generated by a set A, called *free PSE Algebra*; its constructive nature is amenable to investigations of its "local" structure (following the terminology in [5]).

Definition 13 (Longo)
Let $A \neq \emptyset$. Define successively a family of sets $\{B_n : n \in \omega\}$ as follows:
$B_0 \stackrel{\text{def}}{=} A \qquad B_{n+1} \stackrel{\text{def}}{=} B_n \cup \{< \beta, b >: \beta \subseteq B_n \ \& \ b \in B_n\}$
$B \stackrel{\text{def}}{=} \cup B_n, \quad D_A \stackrel{\text{def}}{=} \wp B$
We call $(D_A, \cdot)$ the PSE Algebra freely generated by A. □

There are two canonical ways to expand a PSE algebra into a lambda model, called PSE model, depending on the choice of the graph function Gr which selects a unique representative $Gr(f)$ for each representable function f. The first one, called *strict*, selects the "least" representative $Gr^{\perp}(f)$ from the extensionality class of f, namely

$$Gr^{\perp}(f) \stackrel{\text{def}}{=} \{< \beta, b >: \beta \subseteq B \ \& \ b \in f(\beta)\}$$

giving rise to the *free strict PSE models.* The second one selects the "largest" representative, namely

$$Gr^{\top}(f) \stackrel{\text{def}}{=} Gr^{\perp}(f) \cup A$$

giving rise to the *free lazy PSE model.*

The strict free PSE models have a neat syntactic characterization of their local structure in terms of Böhm Trees:

Theorem 11 (Longo)
Let $\Im$ be a free strict PSE model; then $BT(M) = BT(N)$ iff $\Im \vdash M = N$. □

The free lazy PSE models enjoy the same property versus the Longo Trees. The following result, which confirmed and strengthened a conjecture in [12], was proved by Ong in terms of his lazy PSE preordering ([17]):

Theorem 12 (Ong)
Let $\Im$ be a free lazy PSE model; then $LT(M) = LT(N)$ iff $\Im \vdash M = N$. □

It is this result which determines full abstraction for $\stackrel{\sim}{\leftrightarrow}$, as stated in theorem 10.

6 The use of Well-formed Operators

We prove in this section that lambda observational equivalence is at least as fine as rich applicative congruence. Put another way, it is not possible using well formed operators, to discriminate more than the lambda observational equivalence does. Hence, if we can prove that two pure closed lambda terms are lambda observational equivalent, we can replace them in any context built from well-formed operators without affecting the overall behaviour.

First we have to define the class of well-formed operators.

6.1 Well-formed operators and the *tfyt* format

We will try to follow [10] in the format of the rules that we will allow. We are working here in an higher order calculus which owns notions like variable instantiation and substitution that were not present in the setting considered by Groote and Vaandrager. Therefore we will have to adapt their ideas to give to — what they called — the *tfyt* rules the necessary expressive power. Let us just point out that the rules describing the operators $\nabla, \Diamond, \oplus$ in section 2 fit this formalism. Moreover the argument used in [10] could be reformulated in our setting to show that such a format cannot be generalized in any obvious way without loosing the "good behaviour" of the corresponding operators.

In an attempt to use a more convenient notation to make such rules as powerful as possible, we introduce the convergency predicates $\Downarrow_\lambda$, $\Downarrow_c$; they are abbreviations to express the possibility for a term to reduce to an abstraction or to a term with the constant c in the head position. Formally,

$M \Downarrow_\lambda N$ if $M \Rightarrow N$ and N is an abstraction;

$M \Downarrow_c N$ if $M \Rightarrow N$ and $N = c\tilde{N}$, for some $\tilde{N}$.

Definition 14 *A rule is in* tfyt format *if it can be written as*

$$\frac{\{t_i \varrho_i \sigma_i \mid i \in I\}}{p x_i \ldots x_{r(p)} \Rightarrow t}$$

where

- *I is a finite index set,*
- *$p \in Op$,*
- *$t_i, t, \sigma_i \in \Lambda_C^P(X)$, and*
  $fv(t, t_i) \subseteq \{fv(\sigma_i) \mid i \in I\}) \bigcup \{x_j \mid j \in 1, \ldots, r(p)\}$,
- *$\forall i, \varrho_i \in \{\Rightarrow, \Downarrow_\lambda, \Downarrow_c\}$,*
- *if $\varrho_i \in \{\Rightarrow, \Downarrow_\lambda\}$ then $\sigma_i = y_i$, if $\varrho_i = \Downarrow_c$, then $\sigma_i = c y_{i_1} \ldots y_{i_n}$, for some i_n.*

Moreover the rule has to be non circular. □

Therefore, as in [10], we reject the rules which contain circularity. We will explain in the next subsection what circular rules are and why they can be dangerous.

Remark In [10] also rules in fxyt format are considered, i.e., rules with the same premises as the *tfyt* rules but with conclusion of the form $x \Rightarrow t$. We cannot use such rules in our setting because in general they would take us beyond the lambda calculus. In order to be sound each new redex has to be motivated by an operator p in the head position (we do have however two rules in fxyt format, namely the rules $Rifl$ and $Trans$).

Definition 15 *An operator whose behaviour can be described by rules in tfyt format is called* well-formed. □

6.2 The problem with the circular rules

Definition 16 *A rule is* non circular *if it is possible to order the index set I s.t. if some variable in $fv(\sigma_j)$ appears in the term t_i, then $j < i$ must hold. A rule is* circular *if the above is not true.* □

The following is a circular rule:

$$\frac{p_1 y_2 \Rightarrow y_1 \qquad\qquad p_2 y_1 \Rightarrow y_2}{p_3 \Rightarrow p_4}$$

Circular rules are dangerous: they can give rise to "funny" transition systems, where the behaviour of an operand could really become dependent on the *syntax* of its operands.

The following example shows that in general, if circular rules are allowed, then it is possible to discriminate (unnaturally) more than $\simeq^{Op}$ does. It should also make clear the way how circular rules can implicitly introduce syntactical conditions.

Example 3 *Let p be an operator defined by the following rules (where the first one is circular):*

$$(R1)\,\frac{xy \Rightarrow y}{px \Rightarrow y} \qquad\qquad (R2)\, px \Rightarrow I$$

Let $M_1 = \lambda x.\Omega$, $M_2 = \lambda x.\Omega x$; clearly, $M_1 \overset{\sim}{\leftrightarrow} M_2$. However $pM_1 \not\simeq^p pM_2$. In fact, using R1, we have that $pM_1 \Rightarrow \Omega$; but pM_2 cannot evolve using R1. In fact, for any y we have that $M_2 y \Rightarrow \Omega y \neq y$. The only possible reduction for pM_2 is $pM_2 \Rightarrow I$, but $I \not\simeq^p \Omega$ (notice that in the example, the presence of (R2) is necessary, otherwise $\Omega \simeq^p pM_2 \simeq^p pM_1$). □

In [10] it is left as an open problem whether the hypothesis on non circularity is necessary to keep the validity of their theorems. Unfortunately our example does not seem to help there.

6.3 The result

A set P of symbols can be viewed as a set of operators (when some arity and behavioural rules are specified for each of them) but also as a set of constants. Then the corresponding languages are, respectively, Λ^P and Λ_P, and the applicative bisimulations $\simeq^P$ and $\simeq_P$. In general, Λ^P and Λ_P are not the same; because of the arity conditions, it might be that some Λ_P term is not a Λ^P term. Notice that the simplicity of the treatment of constants (see also subsection 4.2) makes the verification of $\simeq_P$ much easier than the verification of $\simeq^P$. The above language containment and the following result prove that we can indeed exploit such simplicity. Notice that in the assertion of lemma 2, when writing $M \simeq_P N$ the symbols in P are interpreted as constants and M, N as symbols in Λ_P.

Lemma 2 *For every set P of operators and any $M,\ N \in \Lambda^P$, it holds that $M \simeq_P N$ implies $M \simeq^P N$.*

Proof. Remember that we are working here with two languages, Λ_P (where the P symbols are constants) and Λ^P (where they are operators). Therefore we have also two reduction systems, $\Rightarrow_1$ for Λ_P and $\Rightarrow_2$ for Λ^P (with the correspondent convergency predicate $\Downarrow^i_\lambda$, $i \in 1,2$ (since in Λ^P there is no constant, we will never have to use the other convergency predicate, $\Downarrow_c$). An easy fact to prove is that everything derivable using $\Rightarrow_1$ is also derivable with $\Rightarrow_2$, i.e. $M \Rightarrow_1 M'$ implies $M \Rightarrow_2 M'$. In the following, in order to simplify the notation, we will avoid the indices whenever the reduction system used is evident. We will make use in the proof of the congruence properties for $\simeq_P$, proved in the theorem 4.

We will prove that $\Re = \{(M,N) \mid M \simeq_P N\} \subseteq \simeq^P$, by showing that $(M,N) \in \Re$ implies:

1. if $M \Downarrow^2_\lambda M'$ then $\exists N' \,.\, N \Downarrow^2_\lambda N'$ and $(M',N') \in \Re$
2. if $M \Rightarrow_2 M'$ then $\exists N' \,.\, N \Rightarrow_2 N'$ and $(M'N') \in \Re$

Notice that the above clause (1) is different from the corresponding one in definition 1, which requires that for every $R \in \Lambda_P$, $(M'R, N'R) \in \Re$. Our simpler clause suffices because $\simeq_P$ is a congruence for application on the left (this is true in general for Λ^P_C).

We use transition induction.

Basic case: depth of the transition equal to one.

Let us consider clause (2): the case when the rule used was β is immediate from the definition of $\simeq_P$. We have only to prove it when a behavioural rule was used.

Let $px_1 \ldots x_r \Rightarrow_2 t$ be the rule used. Then M is of the form $pM_1 \ldots M_r$ and M' is $t\{M_1/x_1, \ldots, M_r/x_r\}$. By definition of $\simeq_P$, $N \Rightarrow_1 pN_1 \ldots N_r$ and $M_i \simeq_P N_i$, for each $1 \leq i \leq r$. Then also $N \Rightarrow_2 pN_1 \ldots N_r$ and we can infer $N \Rightarrow_2 t\{N_1/x_1, \ldots, N_r/x_r\} = N'$. Since $\simeq_P$ is a congruence, $t\{M_1/x_1, \ldots, M_r/x_r\} \simeq_P t\{N_1/x_1, \ldots, N_r/x_r\}$, yielding $(M',N') \in \Re$.

Let us consider now clause (1), that is $M \Downarrow^2_\lambda \lambda x.M'$ with a reduction of length at most 1. By the above discussion for clause (2) it follows that for some N', $N \Rightarrow_2 N' \simeq_P \lambda x.M'$ and by the definition of $\simeq_P$, N'' exists such that $N' \Rightarrow_1 \lambda x.N'' \simeq_P \lambda x.M'$. Therefore also $N \Downarrow^2_\lambda \lambda x.N''$ and $(\lambda x.M', \lambda x.N'') \in \Re$.

Inductive case: depth of the inference greater than 1

We will only look at clause (2). The problem with clause (1) can be solved by exploiting clause (2) in the same way as we did in the basic case.

Suppose that the last rule to be used was *comp*: then it becomes a simple application of induction hypothesis.

Suppose that the last rule used was the behavioural rule (Q). Remember that in $\Rightarrow_2$ no constant is present and the convergency predicate $\Downarrow_c$ cannot be used; therefore (Q) can be written in the form

$$(Q)\ \frac{\{t_i \varrho_i y_i \mid i \in I\}}{px_1 \ldots x_r \Rightarrow t}$$

for $\varrho_i \in \{\Rightarrow, \Downarrow_\lambda\}$. Moreover (Q) must be not-circular so that an ordering on the index set I can be imposed such that for each i, $fv(t_i) \subseteq \{x_i \mid i \in I\} \cup \{y_j \mid j < i\}$.

Then, M is of the form $pM_1 \ldots M_r$ and for some term t_i^M, y_i^M, t^M, the last step of the derivation for $M \Rightarrow_2 M'$ has the form

$$\frac{\{t_i^M \varrho_i y_i^M \mid i \in I\}}{pM_1 \ldots M_r \Rightarrow t^M}.$$

Also, $M \simeq_P N$ implies $N \Rightarrow_1 pN_1 \ldots N_r$ with $M_i \simeq_P N_i$ for $1 \leq i \leq r$.

We will show that

(*) for $1 \leq i \leq r$, t_i^N, y_i^N exist such that $t_i^N \varrho_i y_i^N$ and $y_i^M \simeq_P y_i^N$.

This would allow us to use the rule (Q) also on $pN_1 \ldots N_r$ and infer

$$\frac{\{t_i^N \varrho_i y_i^N \mid i \in I\}}{pN_1 \ldots N_r \Rightarrow t^N}$$

for some t^N. Then, since t^M and t^N are obtained from t by instantiating the variables with $\simeq_P$ equivalent terms, we get $t^M \simeq_P t^N$ and hence $(t^M, t^N) \in \Re$.

We will prove (*) by induction on i:

- case i=1. Then $fv(t_1) \subseteq \{x_1, \ldots, x_r\}$ and $t_1^M = t_1\{M_1/x_1, \ldots, M_r/x_r\}$. Take $t_1^N = t_1\{N_1/x_1, \ldots, N_r/x_r\}$; by congruence for $\simeq_P$, we have that $t_1^M \simeq_P t_1^N$. Moreover since $t_1^M \varrho_1 y_1^M$ with a shorter inference, using the induction hypothesis, it exists y_1^N such that $t_1^N \varrho_1 y_1^N$ and $y_1^M \simeq_P y_1^N$.
- case i+1 . Analogous. The only difference is that now in t_{i+1} can also appear a variable y_j, for $j < i$. However by induction hypothesis, we know that $y_j^M \simeq_P y_j^N$; therefore the same reasoning yields $t_i^N \varrho_i y_i^N$ and $y_i^M \simeq_P y_i^N$. □

We can now use this lemma to show that lambda observational equivalence is at least as discriminating as rich applicative congruence (and hence also as rich applicative bisimulation).

Corollary 2 $\tilde{\leftrightarrow} < \cong^{Op}$

Proof. Let $M, N \in \Lambda$ with $M \tilde{\leftrightarrow} N$ and C be any Λ^{Op}-context. From theorems 2 and 5, we know that $M \simeq_{Op} N$ and from theorem 4 we can infer $C(M) \simeq_{Op} C(N)$. Finally, from lemma 2, we get $C(M) \simeq^{Op} C(N)$. □

7 The power of Non-Determinism

Corollary 2 left open the question whether $\tilde{\leftrightarrow}$ is strictly finer than $\cong^{Op}$, which, if true, would mean that after all $\tilde{\leftrightarrow}$ is introducing some superfluous discrimination over the lambda terms. We prove in this section that this is not the case and $\tilde{\leftrightarrow}$ and $\cong^{Op}$ represent the same relation (on the pure Λ terms); it therefore inherits from lambda observational equivalence simple and useful characterizations and from rich applicative congruence a measure of its fineness in term of the class of the well-formed operators. This represents also an interesting result from the point of view of the π-calculus and of the modeling it can provide.

Moreover the proof will show that it is sufficient to add non-determinism to the lambda calculus to get full discriminatory power; we will prove in the next section that non-determinism is also necessary. This is perhaps the main result in this paper. Notice that the characterizations of $\tilde{\leftrightarrow}$ in terms of constants and open terms seen in subsection 4.2, although with their own interest, did not give any indication on what the lambda calculus is lacking in terms of *control mechanism* (and actually, none of them seemed to suggest any form of non-determinism).

We will show that $\tilde{\leftrightarrow}<>\simeq^{\oplus}$, i.e. the simple non-deterministic choice operator is enough to give the fully discriminatory power. Actually we will prove that an even less powerful form of non-determinism than $\oplus$ is enough. In fact, we will prove that $\tilde{\leftrightarrow}<>\simeq^{\uplus}$, where $\uplus$ is the operator described by the following rules:

$$(\uplus 1)\ \ \uplus M \to M \qquad\qquad (\uplus 2)\ \ \uplus M \to \Omega$$

First let us point out that both $\simeq^{\oplus}$ and $\simeq^{\uplus}$ are indeed congruences. This is a consequence of the following result:

Lemma 3 *Suppose that P is a set of operators whose behavioural rules do not contain premises (i.e. they are of the simple form $px_1 \ldots x_{r(p)} \Rightarrow t$), then $\cong^P = \simeq^P$.*

Proof. By following exactly the same proof schema used in section 4.1 to show the analogous congruence property for $\simeq_C$. □

The key fact on the way to prove that non-determinism gives the maximal discriminatory power will be lemma 7. For that, we will have also to use the sequence $\{\simeq_{d_n}\}_n$ of approximations of $\simeq_d$ (remember that in $\simeq_d$ only one constant is used and that $\simeq_d$ is a characterization of $\tilde{\leftrightarrow}$ in virtue of theorem 5). Moreover, in order to make the proof of lemma 7 more readable, we found convenient to use also the equivalence $\approxeq^{\uplus}$, a weakening of $\simeq^{\uplus}$ where the first step of the checking only considers convergency. We will introduce $\approxeq^{\uplus}$ in the definition 18.

Definition 17 *We define the sequence of relations $\{\simeq_{d_n}\}_n$ inductively as follows:*

- $\simeq_{d_0} = \Lambda_d \times \Lambda_d$
- $M \simeq_{d_{n+1}} N$ *if*
  1. *whenever $M \Rightarrow \lambda x.M'$, then N' exists such that $N \Rightarrow \lambda x.N'$ and for all $R \in \Lambda_d$, $M'\{R/x\} \simeq_{d_n} N'\{R/x\}$;*
  2. *whenever $M \Rightarrow dM_1 ... M_m$, then $N_1, ..., N_m$ exist such that $N \Rightarrow dN_1 ... N_m$ and for $1 \le i \le m$, $M_i \simeq_{d_n} N_i$.* □

Lemma 4 $\simeq_d <> (\bigcap_n \simeq_{d_n})$

Proof. We have only to prove that $\bigcap_n \simeq_{d_n}$ implies $\simeq_d$. This can be done by showing that $S = \{(M,N) \mid (M,N) \in \bigcap_n \simeq_{d_n}\} \subseteq \simeq_d$. The proof is straightforward because S involves only deterministic terms. □

Definition 18 *Given $M, N \in \Lambda^{\uplus}$, we say that $M \approxeq^{\uplus} N$ when $M \Rightarrow \lambda x.M'$ iff $N \Rightarrow \lambda x.N'$ and $\lambda x.M' \simeq^{\uplus} \lambda x.N'$.* □

We will use $\approxeq^{\uplus}$ to prove negative results about $\simeq^{\uplus}$, as clearly $\not\approxeq^{\uplus}$ implies $\not\simeq^{\uplus}$. Notice that the other way round is not true: for instance, $\uplus I \not\simeq^{\uplus} I$ but $\uplus I \approxeq^{\uplus} I$. The coming lemma describes a property which will make $\approxeq^{\uplus}$ useful to our purposes.

NOTATION Let us write $M \Rightarrow_{\star} N$ if there exists a reduction $M \Rightarrow N$ which does not use the rule $\uplus 2$.

Lemma 5 *Suppose that $M \Rightarrow_{\star} M'$ and $N \Rightarrow_{\star} N'$ with $M' \not\approxeq^{\uplus} N'$. Then also $M \not\approxeq^{\uplus} N$.*

Proof. The assertion $M' \not\approxeq^{\uplus} N'$ means that at least one of them, say M', can reduce to an abstraction, say $\lambda x.M''$, and that if it exists N'' s.t. $N' \Rightarrow \lambda x.N''$, then it is $\lambda x.M'' \not\simeq^{\uplus} \lambda x.N''$. If this is the case, then since $M \Rightarrow M'$, we have also $M \Rightarrow \lambda x.M''$. Now, suppose that N''' exists such that $N \Rightarrow \lambda x.N'''$. This means that $N \Rightarrow_{\star} \lambda x.N'''$, for if the rule $\uplus 2$ were used we would derive a divergent term. Moreover, when rule $\uplus 2$ is not used, the reduction system becomes deterministic. Therefore it is also $N \Rightarrow N' \Rightarrow \lambda x.N'''$. But then from the hypothesis it follows that $\lambda x.N''' \not\simeq^{\uplus} \lambda x.M''$. □

Notice that the above lemma is not true for $\simeq^{\uplus}$. For instance we have $\uplus I \Rightarrow_{\star} I$ and $I \uplus I \Rightarrow_{\star} \uplus I$ with $I \not\simeq^{\uplus} \uplus I$. However it is $\uplus I \simeq^{\uplus} I \uplus I$.

A small technical piece before the crucial lemma 7:

Lemma 6 *Let $M, N \in \Lambda^{\uplus}(X)$ with $fv(MN) \subseteq \{x\}$; if it holds that $N \Rightarrow \Omega$ and $M \Rightarrow \lambda y.M'$, where for $M \Rightarrow \lambda y.M'$ neither the rule $\uplus 1$ nor the rule $\uplus 2$ were used, then $\lambda x.M \not\simeq^{\uplus} \lambda x.N$.*

Proof. The hypothesis on M means that it can only reduce to an abstraction. Hence, whatever be the input R, $(\lambda x.M)R$ cannot diverge, whereas $(\lambda x.N)R$ can. □

NOTATIONS: let $(\lambda x)^n.M$ abbreviate $\lambda x_1.\ldots.\lambda x_n.M$ and $N(M)^n$ abbreviate $NM\ldots M$, where the term M is taken n times. Let us also denote by q_n, $n \in \omega$ the term $(\lambda x)^n.x_n x_1 x_2 \ldots x_{n-1}$. It is usually called the *Böhm permutator* of degree n. (Böhm permutators play a pivotal role in the so-called Böhm-out technique; they will also play a pivotal role in the proof of lemma 7).

Lemma 7 *Suppose that $M \not\simeq_{d_n} N$, for some n; then*
$\exists m_o, k \in \omega\, \exists f : \omega \to \Lambda^{\uplus} .\, \forall m > k\ \exists\ \tilde{R}_m$ and
$M\{f(m+m_o)/d\} \not\approxeq^{\uplus} N\{f(m+m_o)/d\}$.
Moreover, for all n, it is either $f(n) = q_n$ or there exists r s.t. for $r > n$,
$f(n) = (\lambda x)^r. \uplus \lambda x_{r+1}\ldots\lambda x_n.x_n x_1\ldots x_{n-1}$.

Proof. By induction on n. Let f_n stands for $f(n)$ and for any $M \in \Lambda^{\uplus}$, let us use M^f as an abbreviation for the term $M\{(f_{m+m_o})/d\}$.

Basic case Suppose $M \not\simeq_{d_1} N$. The only interesting situation is when none of the terms diverges. Then two cases have to be considered (their symmetric ones can be handled in the analogous way)

- $M \Rightarrow dM_1 \ldots M_r$ and $N \Rightarrow dN_1 \ldots N_{r'}$ with $r < r'$.
  Take $m_o = r+1$, $k = r'$, $\tilde{R}_m = (\Omega)^m(\lambda x)^{m+r}.I$ and $f_n = q_n$. We have
  $M^f \tilde{R}_m\ = q_{(m+r+1)} M_1^f \ldots M_r^f (\Omega)^m (\lambda x)^{m+r}.I \Rightarrow$

[after $m+r+1$ applications of the β rule]

$((\lambda x)^{m+r}.I)M_1^f \dots M_r^f(\Omega)^m \Rightarrow I$

On the other hand,

$N^f\tilde{R}_m = q_{(m+r+1)}N_1^f \dots N_{r'}^f\tilde{R}_m \Rightarrow$

$q_{(m+r+1)}N_1^f \dots N_{r'}^f(\Omega)^m(\lambda x)^{m+r}.I \Rightarrow \simeq^{\uplus} \Omega$

because being $m+r' \geq m+r+1 > r'$, in $q_{(m+r+1)}$ we have that x_{m+r+1} is instantiated with Ω.

- $M \Rightarrow \lambda x.M'$ and $N \Rightarrow dN_1 \dots N_r$, for some r. Fix $m_o = r+1$. Any choice for k, and $\tilde{R}_m$ keeps the truth of what follows.

  Suppose M' is divergent; for $f_n = q_n$ then N^f always accepts two inputs, whereas M^f can only accept one. If M' is not divergent, it must reduce to a term which is either of the form $d\tilde{M}$ or of the form $\lambda y.M''$ or of the form $x\tilde{M}$. We will show that in each of these cases, we can pick up f such that N^f can reduce to an abstraction which is $\not\simeq^{\uplus}$ with $\lambda x.(M')^f$.

  - When $M' \Rightarrow d\tilde{M}$, take $f_n = \uplus q_n$. We have that:
    $N^f = \uplus q_{(m+r+1)}N_1^f \dots N_r^f \rightarrow q_{(m+r+1)}N_1^f \dots N_r^f \Rightarrow (\lambda x)^2.N'$,
    for some N' (remember that $m > 0$), and that
    $(M')^f = \uplus q_{(m+r+1)}\tilde{M}^f \rightarrow \Omega$.
    By lemma 6 it is $(\lambda x)^2.N' \not\simeq^{\uplus} \lambda x.M'$.
  - When $M_1 \Rightarrow \lambda y.M'$, take for $n > r$, (the definition of f_n for $n \leq r$ does not matter)
    $f_n = (\lambda x)^{r+1}.\uplus \lambda x_{(r+2)}\dots\lambda x_n.x_{(n}x_1\dots x_{(n-1)}$.
    We have $N^f \Rightarrow \lambda x.\uplus\tilde{N}'$, for some $\tilde{N}'$.
    Using lemma 6 we can infer $\lambda x.(M')^f \not\simeq^{\uplus} \lambda x.\uplus\tilde{N}'$, as $(M')^f \Rightarrow \lambda y.(M'')^f$ without requiring the rule $\uplus 1$ or $\uplus 2$.
  - Finally, when $M' \Rightarrow x\tilde{M}$, take $f_n = q_n$. We have $N^f \Rightarrow (\lambda x)^2.N'$, for some N'. Then $\lambda x.x\tilde{M}^f$ and $(\lambda x)^2.N'$ can be distinguished by feeding Ω as input, as the former would diverge, whereas the latter would converge.

Inductive case: $\simeq_{d_{n+1}}$ There are two cases to look at.

- $M \Rightarrow dM_1 \dots M_r$, $N \Rightarrow dN_1 \dots N_r$ and for some i, it is $M_i \not\simeq_{d_n} N_i$.

  By induction there are m'_o, k', f' and for $m > k$, $\tilde{R}'_m$ exists such that:

  $M_i\{f'_{(m+m'_0)}/d\}\tilde{R}'_m \not\simeq^{\uplus} N_i\{f'_{(m+m'_0)}/d\}\tilde{R}'_m$.

  Let $m_o = max\{m'_o, r+1\}$, $k = k'$, $f = f'$ and

  $\tilde{R}_m = (\Omega)^{(m+m_o-r-1)}((\lambda x)^{(m+m_o-1)}.x_i\tilde{R}'_{(m+m_o-m'_o)})$.

  Chosen $m > k$, we have to show that $M^f\tilde{R}_m \not\simeq^{\uplus} N^f\tilde{R}_m$.

  First notice that if we denote

  $$A = M_i^f\tilde{R}'_{(m+m_o-m'_o)}, \qquad B = N_i^f\tilde{R}'_{(m+m_o-m'_o)}$$

by reasoning on the indices one can check that the inductive hypothesis can be applied and derive $A \not\gtrsim^{\uplus} B$. Therefore it is enough to show that $M \Rightarrow_{\star} A$ and that $N \Rightarrow_{\star} B$, and then apply lemma 5 to infer $M \not\gtrsim^{\uplus} N$. We have:

$M^f \tilde{R}_m \ = f_{(m+m_o)} M_1^f \dots M_r^f (\Omega)^{(m+m_o-r-1)} (\lambda x)^{(m+m_o-1)} x_i \tilde{R}'_{(m+m_o-m'_o)} \Rightarrow_{\star}$

[after $m + m_o + 1$ applications of the β rule]

$((\lambda x)^{(m+m_o-1)}.x_i \tilde{R}'_{(m+m_o-m'_o-1)}) M_1^f \dots M_r^f (\Omega)^{(m+m_o-r)} \ \Rightarrow_{\star}$

[after $m + m_o - 1$ applications of the β rule]

$M_i^f \tilde{R}'_{m+m_o-m'_o} = A$,

and similarly:

$N^f \tilde{R}_m \ = q_{(m+m_o)} N_1^f \dots N_r^f \tilde{R}_m \ \Rightarrow_{\star}$

$((\lambda x)^{(m+m_o-1)}.x_i \tilde{R}'_{m+m_o-m'_o}) N_1^f \dots N_r^f (\Omega)^{(m+m_o-r-1)} \ \Rightarrow_{\star}$

$N_i^f \tilde{R}'_{m+m_o-m'_o} = B$

- $M \Rightarrow \lambda x.M'$, $N \Rightarrow \lambda x.N'$ and it exists T such that $M'\{T/x\} \not\simeq_{d_n} N'\{T/x\}$.
  Then by hypothesis, $m'_o, k', f', \tilde{R}'_m$ exist such that when $m \geq k'$,
  $(M'\{T/x\})\{f'_{(m+m_o)}/d\}\tilde{R}'_m \not\simeq^{\uplus} (N'\{T/x\})\{f'_{(m+m_o)}/d\}\tilde{R}'_m$. Take $m_o = m'_o$, $k = k'$, $f = f'$ and $\tilde{R}_m \ = \ T^f \tilde{R}'_m$. Then we have
  $M^f \tilde{R}_m \Rightarrow_{\star} \ (M'\{T/x\})^f \tilde{R}_m \ \not\gtrsim^{\uplus} \ (N'\{T/x\})^f \tilde{R}_m \ {}_{\star}\Leftarrow N^f \tilde{R}_m$
  which by lemma 5 implies $M^f \tilde{R}_m \not\gtrsim^{\uplus} N^f \tilde{R}_m$. □

Theorem 13 $\tilde{\leftrightarrow} <> \simeq^{\uplus}$. □

Proof. We already know (corollary 2) that $\tilde{\leftrightarrow} < \simeq^{\uplus}$. We have to prove the opposite and show that for $M, N \in \Lambda$, $M \simeq^{\uplus} N$ implies that $M \tilde{\leftrightarrow} N$. Suppose that $M \not\tilde{\leftrightarrow} N$. Since $\tilde{\leftrightarrow} <> \simeq_d$, then also $M \not\simeq_d N$. By lemma 4 there exists n such that $M \not\simeq_{d_n} N$, and by lemma 7 then we can find $\tilde{R}$ such that $M\tilde{R} \not\gtrsim^{\uplus} N\tilde{R}$ (remember that M, N do not contain the constant d) and therefore also $M\tilde{R} \not\simeq^{\uplus} N\tilde{R}$. This is in contradiction with $M \simeq^{\uplus} N$ because by lemma 3 we know that $\simeq^{\uplus}$ is a congruence. □

Example 4 *Let us show how non-determinism can provide the discriminatory power given by the convergency test. Let M, N be the terms in example 2. We know that $M \simeq N$, but $M \not\simeq^{\nabla} N$. We want to show that also $M \not\simeq^{\uplus} N$. Since both are abstractions, we have to prove that there exists an input R which can discriminate between them. Take $R = \uplus K$, for $K = \lambda xy.x$. Then we have*

$MR \to \uplus K(\lambda y.(\uplus K \Xi \Omega y))\Xi \to$
$K(\lambda y.(\uplus K \Xi \Omega y))\Xi \to$
$\lambda y.(\uplus K \Xi \Omega y)$.

In order to match this move — leading to an abstraction — NR cannot use $\uplus 2$. Then the only possible reduction becomes:

$NR \to \uplus K(\uplus K \Xi \Omega)\Xi \to$
$K(\uplus K \Xi \Omega)\Xi \to$
$\uplus K \Xi \Omega \to K \Xi \Omega \to \Xi$.

But now, $\lambda y.(\uplus K \Xi \Omega y)$ is not equivalent to Ξ as the former after accepting an input can diverge, whereas the latter must converge to an abstraction. □

Theorem 13 allows us to conclude that

Corollary 3 *Suppose that P is a set of operators which contains the operator $\uplus$. Then it holds that $\stackrel{\sim}{\leftrightarrow}\ <>\ \simeq^P\ <>\ \cong^P$.*

Proof. For any set of operators as in the hypothesis it holds that $\cong^{Op}\ <\ \cong^P\ <\ \simeq^P\ <\ \simeq^{\uplus}$. Then it is enough to show that $\stackrel{\sim}{\leftrightarrow}\ <>\ \cong^{Op}\ <>\ \simeq^{\uplus}$. This is true because from corollary 2 and theorems 1, 13 we can infer $\stackrel{\sim}{\leftrightarrow}\ <\ \cong^{Op}\ <\ \simeq^{\uplus}\ <\ \stackrel{\sim}{\leftrightarrow}$. □

From corollary 3, it follows that each pair of elements from

$$S = \{\stackrel{\sim}{\leftrightarrow},\ \cong^{Op},\ \simeq^{Op},\ \cong^{\oplus},\ \simeq^{\oplus},\ \cong^{\uplus},\ \simeq^{\uplus}\}$$

are in the relation $<>$ (for $\oplus$ this is true because $\uplus$ can be defined as as $\oplus\Omega$).

8 Determinism is not enough

Definition 19 *An operator p is* deterministic *if for each P, C and $M_1, \ldots, M_{r(p)} \in \Lambda^P_C$, it holds that $pM_1 \ldots M_{r(p)}$ has at most one reduction.* □

Let DO be the class of all the deterministic operators. We will show in this section that deterministic operators do not give full discriminatory power, that is $\simeq^{DO} \not< \simeq Op$.

First we need a lemma.

Lemma 8 *Suppose $R \in \Lambda^{DO}$ and $R \Rightarrow R'$. Then, $R \cong^{DO} R'$.*

Proof. Take $S = \{(C(R), C(R')) \mid C$ is a Λ^{DO}–context and $R \Rightarrow R'\}$. The proof that S is a $\simeq^{DO}$ bisimulation is by a simple induction on the length of the inference. □

Theorem 14 $\simeq^{DO} \not< \simeq^{Op}$

Proof. Consider $M = \lambda x.(xx)$ and $N = \lambda x.(x\lambda y.(xy))$; M and N have different LT's and therefore $M \not\stackrel{\sim}{\leftrightarrow} N$. By corollary 3, also $M \not\simeq^{Op} N$. However we can show that it is $M \simeq^{DO} N$. This means to prove that for each $R \in \Lambda^{DO}$, $RR \simeq^{DO} R\lambda y.Ry$. There are two cases to consider according to whether R is convergent or not. Suppose that R is convergent to, say $\lambda x.R'$. Then by repeated use of lemma 8, we have that $R\lambda y.Ry \cong^{DO} R\lambda y.(\lambda x.R')y \cong^{DO} R\lambda y.R'\{y/x\} = R\lambda x.R' \cong^{DO} RR$. The case when R is not convergent is easy: in this case $RR \simeq^{DO} \Omega \simeq^{DO} R\lambda y.Ry$. □

Since $\Diamond$ can be expressed as a deterministic operator, theorem 14 proves also that $\stackrel{\sim}{\leftrightarrow}$ is finer than the $\simeq^{\Diamond}$ considered in [1]. This means that Abramsky's canonical domain, which is fully abstract for $\simeq^{\Diamond}$, cannot be a satisfactory model for $\stackrel{\sim}{\leftrightarrow}$. The same thing is true for the equivalences and their fully abstract canonical domains defined by Boudol in [6] and [7] (as on the pure lambda terms they coincide with $\simeq^{\Diamond}$).

9 Conclusions and future work

The study of Milner's encoding of the lazy lambda calculus into the π-calculus has been an extremely rich font of inspirations for the development of the theory of the latter. We hope to have convinced here that the lambda calculus itself can benefit from it. The purpose of our work was to show that the equivalence induced by such encoding is a *robust* one. It has nice fully abstract models in terms of Longo Trees and of free lazy PSE models (besides the one provided by the encoding itself) and it has a number of useful operational characterizations. Its discriminatory power coincide with the one provided by the whole set of well-formed operators. We have also seen that to this purpose one of them, a simple non-deterministic operator, is enough, whereas the use of only deterministic operators is not. We conclude that:

> *The introduction of non-determinism into the lazy lambda calculus is exactly what makes applicative bisimulation appropriate to reason about the functional terms when they are considered in richer settings, possibly involving parallelism.*

Thus, $\Lambda^{\oplus}$ is particularly appealing because of the naturalness of $\oplus$. All this becomes relevant when considering the integration of concurrent and functional programming, the embedding of the lambda calculus in a concurrent language or simply the addition of parallel operators to the lambda calculus. For instance, it tells you that it is sound to replace $\simeq^{\oplus}$-terms in contexts containing concurrency features and that $\simeq^{\oplus}$ is "simplest" applicative bisimulation with this property. Moreover, when applied to the problem of the encoding of the lazy lambda calculus as tackled in [13], this leads to our claim that:

> $\Lambda^{\oplus}$ *is the right form of lambda calculus to be translated into a process calculus.*

It is immediate to extend Milner's encoding to an encoding for $\Lambda^{\oplus}$, simply by taking

$$[M \oplus N](a) = ([M]a) \oplus ([N]a)$$

We do believe that following our study conducted here and in [22, 24], it is possible to prove that there exists an exact correspondence between rich applicative bisimulation on this calculus and lambda observational equivalence for π-calculus , i.e. we conjecture that for M, N in $\Lambda^{\oplus}$,

$$M \simeq^{\oplus} N \Leftrightarrow [M] \stackrel{\sim}{\leftrightarrow} [N]$$

We think also that the "canonicity" of Milner's encoding and the interest of the π-calculus in which they are expressed come strengthened out of our work.

Notice that in [1, 3, 6, 7] the study on the lambda terms is conducted in terms of simulations (and preorders); indeed it is always the case that the bisimulations coincide with the equivalence induced by the correspondent simulation. However this is not true in general when "real" non-determinism is present, and therefore we have preferred to use bisimulations here.

We have already mentioned as an open problem the question of the congruence for $\simeq_C^P$. Other interesting points concerning $\stackrel{\sim}{\leftrightarrow}$ or $\oplus$ would be the following:

- Can the work on canonical models and domain logics carried out in [2, 6, 7] be reformulated in terms of $\overset{\sim}{\leftrightarrow}$ or $\simeq^{\oplus}$? To start with, we would like to understand if the addition of a negation operator is what would turn the logics considered in [2, 6, 7] into suitable logics for $\simeq^{\oplus}$.

- Do they have a characterization in terms of *barbed bisimulation*, proposed in [16] to describe bisimulation-based equivalences in a uniform way over different calculi?

- What is their axiomatisation on finite state terms?

- We have found in this paper what should be added to the (lazy) lambda calculus so that its applicative bisimulation is preserved by the encoding into the π-calculus. The natural complement of such study then would be to examine what *it is necessary to eliminate from the π-calculus* to bring it down to the same level as the (pure) lambda calculus. That is, the question is: what exactly makes the π-calculus more discriminating?

- Can we carry the study that we have done here to other evaluation strategies? What about the call by value, the other strategy encoded by Milner into the π-calculus in [13]?

10 Acknowledgement

I would have not done this work without the suggestions and the encouragement that I received from Robin Milner. I would like also to thank Gerard Boudol, Eugenio Moggi and Luke Ong for their useful comments; and Peter Sewell and David N. Turner for the help in writing the paper.

References

[1] S., Abramsky, *The Lazy Lambda Calculus*, in Research Topics in Functional Programming (D. Turner ed.) Addison-Wesley 65-116, 1989.

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

[3] S., Abramsky and L. Ong, *Fully Abstraction in the Lazy Lambda Calculus*, Research Report, Dep. of Computing, Imperial College, 1989, to appear in Information and Computation.

[4] E., Astesiano and G., Reggio *SMoLCS-driven concurrent calculi*, Lecture Notes in Computer Science, Vol. 249, pp. 169-201, Springer Verlag, 1987.

[5] H., Barendregt, **The Lambda Calculus**, Studies in Logic 103, North-Holland, Revised Edition, 1984.

[6] G., Boudol, *A Lambda Calculus for (strict) Parallel Functions*, Rapport de Recherche 1387, INRIA-Sophia Antipolis, 1991, to appear in Information and Computation.

[7] G., Boudol, *A Lambda Calculus for Parallel Functions*, Rapport de Recherche 1231, INRIA-Sophia Antipolis, 1990.

[8] E., Engeler, *Algebras and Combinators*, Algebra Universalis, Vol. 13, 389-392, 1981.

[9] M., Hennessy, and R., de Nicola, *Testing Equivalences for Processes*, Theor. Comput. Sci. 34, 89-133, 1984.

[10] J., F., Groote and F., W., Vaandrager *Structured Operational Semantics and Bisimulation as a Congruence*, in Proc. ICALP '89, LNCS 372, 423-438, Springer Verlag, 1989.

[11] J., Levy, *An Algebraic Interpretation of Equality in Some Model of the Lambda Calculus*, in C. Böhm editor, Lambda Calculus and Computer Science Theory, 147-165, LNCS No. 37, 1975.

[12] G., Longo, *Set Theoretical Models of Lambda Calculus: Theory, Expansions and Isomorphisms*, Annales of Pure and Applied Logic, 24:153-188, 1983.

[13] R., Milner, *Function as Processes*, Report N.1154, INRIA-Roquencourt, 1990, to appear in Journal of Mathematical Structures in Computer Science.

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

[15] R., Milner, J.G., Parrow, and D.J., Walker, *A Calculus of Mobile Processes, Parts I and II*, Report ECS-LFCS-89-85 and -86, Laboratory for Foundations of Computer Science, Computer Science Department, Edinburgh University, 1989, to appear in Journal of Information and Computation.

[16] R., Milner and D., Sangiorgi, *Barbed Bisimulations*, submitted to ICALP 1992.

[17] L., Ong, *The Lazy Lambda Calculus: an Investigation into the Foundations of Functional Programming*, PhD thesis, University of London, 1988. Also Prize Fellowship Dissertation, Trinity College, Cambridge 256 pp.

[18] L., Ong, *Fully Abstract Models of the Lazy Lambda Calculus*, in Proc. 29th Conf. Foundations of Computer Science, pp. 368-376, 1988.

[19] L., Ong, *The Non-deterministic Lambda Calculus*, working draft, October 1991.

[20] , D., Park, *Concurrency and Automata on Infinite Sequences*, Vol 104, Springer Verlag, 1980.

[21] B., Thomsen, *Calculi for Higher Order Communicating systems*, PhD thesis, Imperial College, London University, 1990.

[22] D., Sangiorgi, forthcoming PhD thesis, University of Edinburgh, 1992.

[23] D., Sangiorgi, *From Higher Order π-calculus to the π-calculus*, notes, Edinburgh, 1991.

[24] D., Sangiorgi, *Lambda Calculus, Higher Order π-calculus and π-calculus*, notes, Edinburgh, 1991.