The Lazy Lambda Calculus in a Concurrency Scenario*

Davide Sangiorgi
LFCS, Department of Computer Science,
University of Edinburgh
JCMB, The Kings' Buildings
Edinburgh, EH9 3JZ, U.K.

email: `sad@dcs.ed.ac.uk`

(∗): To appear in Information and Computation. An extended abstract has appeared in the Proceedings of the seventh annual IEEE Symposium on Logic in Computer Science (LICS '92), IEEE Computer Society Press.

This research has been partially supported by the BRA ESPRIT project 6454 CONFER.

Abstract

The use of λ-calculus in richer settings, possibly involving parallelism, is examined in terms of the effect on the equivalence between λ-terms. We concentrate on Abramsky's *lazy λ-calculus* (Abramsky 1989) and we follow two directions. Firstly, the λ-calculus is studied within a process calculus by examining the equivalence $\overset{\sim}{\leftrightarrow}$ induced by Milner's encoding into the π-calculus. We start from a characterisation of $\overset{\sim}{\leftrightarrow}$ presented in (Sangiorgi 1992). We derive a few simpler operational characterisations, from which we prove full abstraction w.r.t. Levy-Longo Trees. Secondly, we examine Abramsky's *applicative bisimulation* (in *op. cit.*) when the λ-calculus is augmented with (well-formed) operators, that is symbols equipped with reduction rules describing their behaviour. In this way, the maximal discrimination between pure λ-terms (i.e., the finest behavioural equivalence) is obtained when all operators are used. We prove that the presence of certain non-deterministic operators is sufficient and necessary to induce it and that it coincides with the discrimination given by $\overset{\sim}{\leftrightarrow}$. We conclude that the introduction of non-determinism into the λ-calculus is exactly what makes applicative bisimulation appropriate for reasoning about the functional terms when concurrent features are also present in the language, or when they are embedded into a concurrent language.

1 Introduction

The λ-calculus is canonical for calculations with functions. We concentrate here on Abramsky's ideas (Abramsky 1989). His *lazy λ-calculus* is proposed as a basis for 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 λ-abstraction. In such a setting, termination means "reduction to an abstraction" and is the only observable property. Then Abramsky decrees two closed λ-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 (Abramsky 1987) he develops a theory for applicative bisimulation which runs in parallel with his treatment of concurrency. The definition of $\simeq$ itself inherits the bisimulation idea originally formulated in concurrency theory (Park 1981, Milner 1989). It also has an alternative characterisation reminiscent of *testing equivalence* (De Nicola and Hennessy 1984); it says that two terms are equivalent when they induce the same termination property in all pure λ-contexts:

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

However, such a definition of applicative bisimulation causes some problems. It is based on the notion of termination, but one cannot distinguish termination in the pure λ-calculus. For instance, one cannot define the *convergence test* (see Section 2). Such an operator is used to show that Abramsky's canonical domain for the lazy λ-calculus and Milner's encoding of it into the π-calculus (Milner 1990) are not fully abstract. In other words, the pure λ-calculus is too weak w.r.t. the predicate of termination. Moreover, since applicative bisimulation derives from ideas developed for frameworks of reactive and concurrent systems, one might find it appealing the introduction of "parallel" operators in the contexts of definition ($*$). Indeed, various enrichments of the lazy λ-calculus with operators not λ-definable have already appeared in the literature. However either the operators themselves — as in the case of convergence test and parallel convergence in (Abramsky and Ong 1989, Ong 1988 and 1988a) — or their semantics — as for non-deterministic choice and the parallel operator in (Boudol 1990 and 1991) — 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 the common practice. Furthermore it is unclear whether the induced equivalences are sensitive to the addition of more operators. This question is relevant when considering the integration of functional and concurrent calculi: For instance we might like 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 λ-calculus in "richer" settings, the focus of this paper. We have pursued two approaches: In the first, the λ-calculus is studied *within* a process calculus, in some sense completing Abramsky's immersion of the lazy λ-calculus into concurrency. Since the lazy λ-calculus was inspired by the language-implementation experience, a "powerful" process calculus should yield a simple encoding. We have chosen the π-calculus for this, where a nice encoding already exists, namely Milner's (1990, 1991). It has to be stressed that the study of Milner's encoding was a major concern in our work. If the λ-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 something worth looking at, and Milner's encoding of the lazy λ-calculus, because of its simplicity and canonicity, represents an interesting starting point. We have called the equivalence induced by the embedding *λ-observation equivalence*, written $\overset{\sim}{\leftrightarrow}$; that is, two λ-terms are λ-observation equivalent if their process-encodings are (weak) bisimilar[1]. A characterisation of $\overset{\sim}{\leftrightarrow}$ is presented in (Sangiorgi 1992, 1993). We derive a few simpler operational characterisations, from which we prove full abstraction of $\overset{\sim}{\leftrightarrow}$ w.r.t. Levy-Longo Trees (Longo 1983, Ong 1988a), the lazy variant of Böhm Trees. As a corollary, due to previous results by Longo and Ong in *op. cit.*, we also get full abstraction w.r.t. the class of models called (free lazy) Plotkin-Scott-Engeler models.

Our other approach tackles a systematic study of the lazy λ-calculus and applicative bisimulation in presence of a richer class of operators than those λ-definable. That is, rather than going through the embedding into an auxiliary language, *we enrich the pure λ-calculus.* We admit only *well-formed* operators, intuitively operators whose behaviour only depends on the semantics — not on the syntax — of their operands. Groote and Vaandrager (1992) have studied the meaning of well-formed operators and transition systems in a process algebra setting. We adapt their format to the λ-calculus. Then the most discriminating congruence is obtained when all well-formed operators are admitted; we call it *rich applicative congruence.* We show that λ-observation equivalence and rich applicative congruence coincide on the pure λ-terms. In other words, the π-calculus encoding induces maximal observational discrimination on λ-terms. An interesting problem is then to find a minimal set of operators giving the same discrimination. The solution of this problem involves the understanding of *what is necessary to add to the λ-calculus to make it as discriminating as the π-calculus.* Could the parallel convergence operator be the solution to this problem just as it was the solution to the full abstraction problem for Abramsky's canonical model (Abramsky 1987)? The answer is no: Parallel convergence is a *Church-Rosser* operator (i.e., it yields confluent derivatives), and one of our results is that Church-Rosser operators do not give maximal discriminating power. The right answer is non-determinism. We prove that one of the simplest forms of non-determinism one could think of, a unary operator which when applied to some argument either behaves like the argument or diverges, is enough. This gives an indicative measure of the power of non-determinism.

An alternative use of constants has played a crucial rôle in the operational study of $\overset{\sim}{\leftrightarrow}$. The standard way to treat a constant is to introduce it together with some rules describing its operational behaviour. In the sequel we call these *operators.* When only operators are used, λ-abstraction remains the only sensible normal form for closed terms. Instead, we take the word *constant* to denote symbols which are added to the language without specifying any operational rule. Such a use of constants can be found in the well-known technique of *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 been left hidden. A constant c is then a 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 can imagine it as the *output* of the tuple $\tilde{M}$ along the channel c and towards K_c.

---

[1] In the concurrency terminology, an equivalence is *weak* if it ignores possible internal moves of processes; due to the particular structure of the processes encoding λ-terms, we believe that many of the weak process equivalences studied in the literature would induce the same relation on the λ-terms.

We shall compare equivalences defined on different enriched λ-languages by the discrimination induced on pure closed λ-terms. Not much is known about the preorder on the equivalences so obtained, which looks like a semilattice. There is a maximal element represented by Abramsky's original applicative bisimulation and a minimal one represented by the rich applicative congruence. Further informations can be deduced from our work. However the relationships between various interesting equivalences remain unknown.

We do not introduce the π-calculus or Milner's encoding of the lazy λ-calculus, since we shall only use them as starting point and we shall never be performing manipulations of π-calculus processes. We refer to (Milner *et al.* 1992), (Milner 1990, 1991) and (Sangiorgi 1993) for detailed expositions.

Related work. We are not aware of other studies on equivalences between λ-terms induced via a mapping into a concurrent language. On the contrary, a number of studies of extensions of the λ-calculus have appeared in the literature; for the lazy λ-calculus, we have already mentioned those by Abramsky, Ong and Boudol. Coming, more specifically, to non-deterministic extensions of λ-calculus, these have been mainly analysed on the typed calculus (see for instance (Astesiano and Costa, 1980) and (Sieber 1993)) and only very recently on the untyped calculus, by Jagadeesan and Panangaden (1990), Boudol (1990), Ong (1993), de'Liguoro and Piperno (1992). The emphasis in these works is mainly domain-theoretic. Operationally, the equivalences defined are different from ours and the reason is clear if we relate them to standard behavioural equivalences of process algebras. Apart from (Ong 1993), all cited authors closely follow the testing theory (De Nicola and Hennessy 1984), in its modalities *may* or *must*, separately or together. By contrast, we follow the treatment of non-determinism and reductions in the classical theory of (weak) bisimulation (Milner 1989). Ong's approach (Ong 1993) lies between the two, since his definition of equivalence inherits both testing and bisimulation elements. See also Examples 3.2, 3.3 and 3.4 for comparisons among these equivalences.

Organisation of the paper. Section 2 introduces some necessary notation for the lazy λ-calculus. In Section 3 applicative bisimulation is generalised to λ-calculus languages enriched with a class P of operators and a class C of constants; we denote it by $\simeq_C^P$ ($\simeq_C$ when P is empty). The remainder of the paper is conceptually divided into two parts. The first part includes Sections 4 and 5. In Section 4 we study $\overset{\sim}{\leftrightarrow}$ operationally: We begin with its characterisation in term of $\simeq_\Psi$ given in (Sangiorgi 1992), where Ψ is an infinite set of constants; we prove that this result is actually independent of the class of constants used, as long as it is nonempty; we present a simple direct proof (which does not exploit the encoding into the π-calculus) of the congruence of $\simeq_C$; we finish with two further useful characterisations of $\overset{\sim}{\leftrightarrow}$. In Section 5 we show the full abstraction of $\overset{\sim}{\leftrightarrow}$ w.r.t. Levy-Longo Trees.

In the second part of the paper, including Sections 6, 7 and 8, we examine enrichments of the lazy λ-calculus with well-formed operators. Various comparison results between equivalences induced by different classes of operators are derived but the real aim is to use operators to understand and describe the discriminating power given by $\overset{\sim}{\leftrightarrow}$. We introduce well-formed operators in Section 6; we define and motivate the format of their operational rules and we show that $\overset{\sim}{\leftrightarrow}$ is at least as fine as rich applicative congruence. In Section 7 we establish the opposite containment. This is done only using a simple non-deterministic operator. In Section 8 we prove that for such a result non-determinism is actually *necessary*, i.e., the same discriminating power cannot be recovered using only Church-Rosser operators. Finally, in Section 9 we report some conclusions and, as future work, some questions which remain to be examined.

2 Preliminary notations and definitions

We use $x, y, \ldots$ to range over variables; $c, d, \ldots$ and C over constants and classes of constants; $p, \ldots$ and P over operators and classes of operators. We assume that each operator p has an arity $r(p)$ representing the number of arguments that p needs. The class of $\Lambda_C^P(\mathcal{X})$-terms, i.e., λ-terms enriched with operators in P and constants in C is defined by the following grammar

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

The definitions of free variables, closed terms, substitution, α-conversion etc. are the standard ones (see Barendregt 1984). The set of constants and of free variables in the term M are denoted by $ct(M)$ and $fv(M)$, respectively. Throughout the paper we assume that *all α-convertible terms are identified* and we write $M = N$ if M and N are α-convertible. The subclass of $\Lambda_C^P(\mathcal{X})$ only containing the closed terms is denoted by Λ_C^P. We omit P or C if they are empty. Thus, Λ and Λ_C are respectively the class of the closed pure λ-terms and the class of the closed λ-terms enriched with constants from C. We group brackets on the left; therefore $MNRL$ is $((MN)R)L$. We use M, N, R, T to range over (enriched) λ-terms; also, $\tilde{M}$ in $N\tilde{M}$ stands for a sequence of arguments, i.e., $N\tilde{M} = NM_1..M_n$, for some n and M_1, ..., M_n. We abbreviate $\lambda x_1. \cdots \lambda x_n.M$ as $\lambda x_1 \cdots x_n.M$, or $\lambda\tilde{x}.M$ if the length of $\tilde{x}$ is not important. As usual, the symbol I is the identity term $\lambda x.x$; the symbol Ω is the always-divergent term $(\lambda x.xx)(\lambda x.xx)$; and Ξ the always-convergent term $(\lambda x.\lambda y.(xx))(\lambda x.\lambda y.(xx))$.

Now the reduction relation $\Rightarrow \subseteq \Lambda_C^P(\mathcal{X}) \times \Lambda_C^P(\mathcal{X})$ (for our purposes it is convenient to have it defined on open terms). We express the reduction rules using metavariables $X, \cdots, Y, \cdots$, which are instantiated with enriched λ-terms when a rule is applied (the use of metavariables will be particularly handy in Section 6, to describe the rules of the Groote-Vaandrager format). First of all we have the rules (β) and (App), the core of the pure lazy λ-calculus; then the rules *(Refl)* and *(Trans)*, describing the reflexive and transitive nature of $\Rightarrow$:

$$(\beta) \quad (\lambda x.X)Y \Rightarrow X\{Y/x\} \qquad (App) \quad \frac{X_1 \Rightarrow Y}{X_1 X_2 \Rightarrow Y X_2}$$

$$(Refl) \quad X \Rightarrow X \qquad (Trans) \quad \frac{X \Rightarrow Y_1 \qquad Y_1 \Rightarrow Y_2}{X \Rightarrow Y_2}$$

Finally, there is a set of rules for each operator. We call them *behavioural rules.* Following Groote and Vaandrager (1992), we only admit rules in *Groote-Vaandrager* format, which ensure that the behaviour of the operators defined, called *well-formed*, only depends on the semantics and not on the syntax of their operands. We shall describe such a format in detail in Section 6. For the moment, as an example, we give the rules for ∇ (*convergence test*), $\Diamond$ (*parallel convergence test*) and $\oplus$ (*unconditional choice*, sometimes called *internal choice* (De Nicola and Hennessy, 1987)). We use $\Downarrow_\lambda$ as a convergence predicate; $M \Downarrow_\lambda N$ holds if M reduces to N and N is an abstraction.

$$(\nabla 1) \ \frac{X \Downarrow_\lambda Y}{\nabla X \Rightarrow I} \qquad (\nabla 2) \ \frac{X \Rightarrow Y}{\nabla X \Rightarrow \nabla Y}$$

$$(\Diamond 1) \ \frac{X_1 \Downarrow_\lambda Y}{\Diamond X_1 X_2 \Rightarrow I} \qquad (\Diamond 2) \ \frac{X_2 \Downarrow_\lambda Y}{\Diamond X_1 X_2 \Rightarrow I} \qquad (\Diamond 3) \ \frac{X_1 \Rightarrow Y_1 \qquad X_2 \Rightarrow Y_2}{\Diamond X_1 X_2 \Rightarrow \Diamond Y_1 Y_2}$$

$$(\oplus 1) \ \oplus X_1 X_2 \Rightarrow X_1 \qquad (\oplus 2) \ \oplus X_1 X_2 \Rightarrow X_2$$

$\overset{\sim}{\leftrightarrow}$	$\Lambda(\mathcal{X}) \times \Lambda(\mathcal{X})$	λ-observation equivalence
$\simeq^P_C$	$\Lambda^P_C \times \Lambda^P_C$	applicative bisimulation over Λ^P_C
$\cong^P_C$	$\Lambda^P_C \times \Lambda^P_C$	congruence induced by $\simeq^P_C$
$\overset{o}{\simeq}$	$\Lambda(\mathcal{X}) \times \Lambda(\mathcal{X})$	applicative bisimulation over open terms

∇	convergence test
$\Diamond$	parallel convergence
$\oplus$, $\talloblong$, $\uplus$	choice operators
Op	class containing all well-formed operators
Ψ	countable infinite set of constants

Table 1: Main symbology

To aid readability, we shall often write binary operators in infix position, as e.g., $M \oplus N$. Table 1 summarises the main notations for the equivalences and for the operators or classes of operators that will be used throughout the paper. In the entries for equivalences, the central column represents their domain.

Remark 2.1 Since constants have no associated reduction rules, they behave, essentially, as free variables. Constants are separated from free variables for two reasons: Firstly, they play logically distinct rôles in the proof of Theorem 4.2 below in (Sangiorgi 1992). Secondly, we think that constants have their own natural interpretation, as described in Section 1.

3 Applicative Bisimulation over Λ^P_C

In (Abramsky 1987, Abramsky and Ong 1989, Boudol 1990 and 1991) the study of λ-terms is conducted in terms of simulations and preorders[1]; indeed it is always the case that a bisimulation coincides with the equivalence induced by the corresponding simulation. However this is not true in general with non-determinism, hence we prefer to work with *bi*simulations. When generalising the definition of applicative bisimulation given by Abramsky (1987) to terms in Λ^P_C, the main question is how to define it between the terms $c\tilde{M}$ and $c\tilde{N}$. Following our interpretation of constants given in Section 1, it is natural to require that the *ordered* sequence of the arguments $\tilde{M}$ and $\tilde{N}$ be equivalent.

Definition 3.1 *A symmetric relation $\mathcal{S} \subseteq \Lambda^P_C \times \Lambda^P_C$ is a $\simeq^P_C$-bisimulation, if $(M, N) \in \mathcal{S}$ implies:*

1. *if $M \Rightarrow \lambda x.M'$ then N' exists s.t. $N \Rightarrow \lambda x.N'$ and $(M'\{R/x\}, N'\{R/x\}) \in \mathcal{S}$, for all $R \in \Lambda^P_C$;*
2. *if $M \Rightarrow cM_1 ... M_n$, for some $n \geq 0$ and $c \in C$, then $N_1, \ldots, N_n$ exist s.t. $N \Rightarrow cN_1 \cdots N_n$ and $(M_i, N_i) \in \mathcal{S}$, $1 \leq i \leq n$;*

---

[1] Abramsky uses the word bisimulation for the preorder; we shall follow the concurrency tradition and call the symmetric relation bisimulation.

3. if $M \Rightarrow M'$ then N' exists s.t. $N \Rightarrow N'$ and $(M', N') \in \mathcal{S}$.

The terms M and N are applicative bisimilar over Λ_C^P, *written $M \simeq_C^P N$, if $(M, N) \in \mathcal{S}$, for some $\simeq_C^P$-bisimulation $\mathcal{S}$.* □

It is easy to see that $\simeq_C^P$ induces an equivalence relation. When P contains only one element, say p, we just write $\simeq_C^p$. We drop the index P or C when the corresponding set is empty.

In Definition 3.1, clause (3) requires us to compare two λ-terms also after some internal activity has happened. This is important when the language can express non-determinism, since it allows us to detect the branching structure of the terms, as the examples below show. Clause (3) can be omitted when the reduction relation $\Rightarrow$ is confluent — for instance, if P is empty — since then, $M \Rightarrow M'$ implies that M and M' are bisimilar. In consequence, $\simeq$, i.e., applicative bisimulation over Λ, represents Abramsky's original applicative bisimulation (Abramsky 1987).

Example 3.2 *It holds that $I \not\simeq^{\oplus} I \oplus \Omega$, since the latter has the reduction $I \oplus \Omega \Rightarrow \Omega$ which the former cannot match. The terms I and $I \oplus \Omega$ are equated in (Boudol, 1990), where clause (3) is absent. However, the distinction between them seems senseful, since I always accepts an input whereas $I \oplus \Omega$ can also refuse it.* □

Example 3.3 *We have $\lambda x.(M \oplus N) \not\simeq^{\oplus} (\lambda x.M) \oplus (\lambda x.N)$, i.e., abstraction does not distribute over unconditional choice as, on the contrary, holds in (de'Liguoro and Piperno 1992) and (Boudol 1990). For instance, we can show $\lambda x.(I \oplus \Omega) \not\simeq^{\oplus} (\lambda x.I) \oplus (\lambda x.\Omega)$ as follows: The term $(\lambda x.I) \oplus (\lambda x.\Omega)$ can reduce to the abstraction $\lambda x.I$. On the other hand, the only abstraction to which $\lambda x.(I \oplus \Omega)$ can reduce is itself, for in lazy λ-calculus reductions underneath an abstraction are forbidden. But now, any input M can distinguish between $\lambda x.I$ and $\lambda x.(I \oplus \Omega)$, since*

$$\big(\lambda x.(I \oplus \Omega)\big)M \Rightarrow I \oplus \Omega \quad \textit{whereas} \quad (\lambda x.I)M \Rightarrow I$$

and $I \not\simeq^{\oplus} I \oplus \Omega$, as seen in Example 3.2.

The failure of the equality $\lambda x.(M \oplus N) \simeq^{\oplus} (\lambda x.M) \oplus (\lambda x.N)$ closely resembles the well-known failure of the bisimilarity equality of process algebras between the processes $a.(P \oplus Q)$ and $(a.P) \oplus (a.Q)$ (the construct $a.P$ denotes a process which can perform the action a and then becomes the process P). □

Example 3.4 *Unconditional choice $\oplus$ is associative in (Ong 1993), but it is not so for us. For instance, we have*

$$(I \oplus \Omega) \oplus \lambda x.\Omega \not\simeq^{\oplus} I \oplus (\Omega \oplus \lambda x.\Omega)$$

since the former can reduce to $I \oplus \Omega$, which the latter cannot match, as can be shown by a case analysis on its possible reductions. Again, this result is in accordance with the theory of bisimulation in process algebras, where unconditional choice is not associative. □

We now introduce a choice operator, called *conditional choice* and written $[]$, which is indeed associative. It is inspired by the *external choice* operator of (De Nicola and Hennessy 1987), and is described by the rules

$$([]1)\ \frac{X_1 \Rightarrow Y}{[]X_1X_2 \Rightarrow []YX_2} \qquad\qquad ([]2)\ \frac{X_1 \Downarrow_{\lambda} Y}{[]X_1X_2 \Rightarrow Y}$$

plus the symmetric rules ($[\!]3$) and ($[\!]4$). We shall see later (discussion after Corollary 7.10) that $\oplus$ and $[\!]$ induce the same (maximal) discrimination on pure λ-terms.

Congruence. A Λ_C^P-context is a 'Λ_C^P-term' with a hole $[\,]$ in it. If C is a Λ_C^P-context, then $C(M)$ is the Λ_C^P term obtained from C by filling its hole with M.

Definition 3.5 Applicative congruence over Λ_C^P, *written* $\cong_C^P$, *is the largest relation over* $\Lambda_C^P \times \Lambda_C^P$ *s.t.* $M \cong_C^P N$ *implies* $C(M) \simeq_C^P C(N)$, *for every* Λ_C^P*-context* C. $\square$

Although we conjecture it is true, we were not able to prove that $\simeq_C^P$ always coincides with its congruence $\cong_C^P$; but, at least, we shall prove it for the specific cases which interest us.

The preorder on the equivalences. In the sequel we consider various equivalences defined on λ-calculus terms possibly enriched with constants and operators. Since the classes of terms on which they are defined may differ, we compare them on the common core of closed pure λ-terms.

Definition 3.6 *Let* $\sim$, $\sim'$, *be two equivalences defined on some enriched* λ*-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.,* $\sim$ *identifies less, is a finer relation); and* $\sim\, <> \,\sim'$ *if both* $\sim\, < \,\sim'$ *and* $\sim'\, < \,\sim$ *hold.* $\square$

Lemma 3.7 *If* $P \subseteq P'$ *and* $C \subseteq C'$, *then* $\simeq_{C'}^{P'} < \simeq_C^P$. $\square$

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

Our starting point is the equivalence induced on the λ-terms by Milner's encoding into the π-calculus, and its characterisation in terms of $\simeq_\Psi$ proved in (Sangiorgi 1992), where Ψ is a countable infinite set of constants. Formally, the encoding of Milner's we refer to is the one presented in (Milner 1991), slightly simpler than the previous one in (Milner 1990), due to the use of tuple communications and process abstractions. We write $[\![M]\!]$ for the encoding of the term M and $\approx$ for π-calculus's weak bisimulation, also called *observation equivalence*. According to the terminology in (Milner *et al.* 1992) we should also say which version of observation equivalence we mean, if the *late* or the *early*, *ground* or *not ground*; this is unnecessary because they all coincide on the encoding of λ-terms.

Definition 4.1 *We say that the terms* $M, N \in \Lambda(\mathcal{X})$ *are* λ-observation equivalent, *written* $M \overset{\sim}{\leftrightarrow} N$, *if* $[\![M]\!] \approx [\![N]\!]$. $\square$

The characterisation of $\overset{\sim}{\leftrightarrow}$, left as open problem in (Milner 1990), was achieved in (Sangiorgi 1992) by extending Milner's encoding to $\Lambda_\Psi(\mathcal{X})$ and by exploiting a more abstract encoding into the *Higher-Order* π*-calculus*, a development of π-calculus with higher-order communications.

Theorem 4.2 (from (Sangiorgi 1992)) *If* $M, N \in \Lambda$, *then* $M \overset{\sim}{\leftrightarrow} N$ *iff* $M \simeq_\Psi N$. $\square$

4.1 Congruence of $\simeq_C$

By appealing to the encoding and to the properties of π-calculus processes, we get for free the congruence of $\simeq_\Psi$. But it is a valuable test for $\simeq_\Psi$ to show that a simple direct proof is possible. Our proof follows the one proposed by Stoughton to show the congruence of $\simeq$ (see Abramsky and Ong 1989); in addition, here we need the notion of "bisimulation up-to". Unfortunately, Stoughton's technique does not work in general for $\simeq_C^P$. We give the proof in terms of a generic class C of constants. It is convenient to use the one-step reduction relation $\rightarrow$ for Λ_C; it is obtained from $\Rightarrow$ by dropping the rules *(Refl)* and *(Trans)* and replacing $\Rightarrow$ with $\rightarrow$ in the rules (β) and *(App)*. If $\mathcal{S}$ is a relation, we write $M \mathcal{S} N$ if $(M,N) \in \mathcal{S}$ and $M \mathcal{S} \simeq_C N$ if L exists s.t. $M \mathcal{S} L \simeq_C N$.

Definition 4.3 *$\mathcal{S} \subseteq \Lambda_C \times \Lambda_C$ is an* applicative bisimulation up-to $\simeq_C$ *if $\mathcal{S}$ is symmetric and $M \mathcal{S} N$ implies:*

1. *if $M = \lambda x.M'$ then N' exists s.t. $N \Rightarrow \lambda x.N'$ and $M'\{R/x\} \mathcal{S} \simeq_C N'\{R/x\}$, for all $R \in \Lambda_C$,*
2. *if $M = cM_1...M_n$, for some $n \geq 0$ and $c \in C$, then $N_1, \ldots, N_n$ exist s.t. $N \Rightarrow cN_1 \cdots N_n$ and $M_i \mathcal{S} \simeq_C N_i$, $1 \leq i \leq n$,*
3. *if $M \rightarrow M'$ then N' exists s.t. $N \Rightarrow N'$ and $M' \mathcal{S} \simeq_C N'$.* ☐

Lemma 4.4 *If $\mathcal{S}$ is an applicative bisimulation up-to $\simeq_C$ then $\mathcal{S} \subseteq \simeq_C$*

PROOF: Standard technique for bisimulations up-to (Milner 1989). ☐

Lemma 4.5 *$M \simeq_C N$ implies $MR_1 \cdots R_n \simeq_C NR_1 \cdots R_n$, for any Λ_C terms $R_1, \ldots, R_n$.*

PROOF: Use induction on n and the definition of $\simeq_C$. ☐

Proposition 4.6 (congruence of $\simeq_C$) *The relations $\simeq_C$ and $\cong_C$ coincide.* ☐

PROOF: We show that

$$\mathcal{S} = \{(C(M), C(N)) \mid C \text{ is a } \Lambda_C\text{-context and } M \simeq_C N\}$$

is an applicative bisimulation up-to $\simeq_C$, by induction on the structure of the context C.

1. Suppose C of the form $(\lambda x.C_1)C_2 \cdots C_n$: The case $n = 1$ is easy. Otherwise, take $C' = (C_1\{C_2/x\})C_3 \cdots C_n$; we have $C(M) \rightarrow C'(M)$, $C(N) \rightarrow C'(N)$ and $(C'(M), C'(N)) \in \mathcal{S}$.
2. Suppose C of the form $cC_1 \cdots C_n$: immediate.
3. Suppose C of the form $[\,]C_1 \cdots C_n$: Define $C' = MC_1 \cdots C_n$. Then C' is either of the form (1) or (2). In both cases, the behaviour of $C'(M)$ is matched by $C'(N)$; this is enough because, by Lemma 4.5, $C(N) \simeq_C C'(N)$ and because $\mathcal{S}$ is a bisimulation up-to $\simeq_C$. ☐

4.2 Simpler characterisations

Independence from the class of constants. Theorem 4.2 gives us a characterisation of λ-observation equivalence using an infinite set Ψ of constants. Our first result is that the choice of the class of constants is not important, as long as it is nonempty.

Theorem 4.7 *For any nonempty C and C', it holds that $\simeq_C <> \simeq_{C'}$.*

Proof: It suffices to compare $\simeq_d$, where only the constant d is used, with $\simeq_\Psi$, where a countable infinite number of them is used, and show that $\simeq_d < \simeq_\Psi$. Let $A_1, ..., A_n, ...$ be in Λ and be all pairwise related by $\not\precsim$; for instance $A_i = \lambda x_1 ... x_i . \Omega$. If $M \in \Lambda_\Psi$, let M^+ be the term of Λ_d obtained from M by replacing the constant c_i with the term dA_i. Then with an easy transition induction one can prove that $\mathcal{S} = \{(M, N) \mid M^+ \simeq_d N^+\}$ is a $\simeq_\Psi$-bisimulation. This proves $\simeq_d < \simeq_\Psi$ because $\mathcal{S}$ contains $\{(M,N) \mid M, N \in \Lambda$ and $M \simeq_d N\}$. $\Box$

For equivalence checking. The next characterisation is useful for *verifying* λ-observation equivalence. It also suggests an analogy with the verification of equivalences in higher-order calculi: In (Sangiorgi 1992) we show that bisimulation in the Higher-Order π-calculus has a characterisation very close in intent to the following $\simeq'_\Psi$.

Definition 4.8 *The relation $\simeq'_\Psi$ is the largest symmetric relation on $\Lambda_\Psi \times \Lambda_\Psi$ s.t. $M \simeq'_\Psi N$ implies:*

1. *if $M \Rightarrow \lambda x.M'$ then N' exists s.t. $N \Rightarrow \lambda x.N'$ and $M'\{c/x\} \simeq'_\Psi N'\{c/x\}$, for $c \notin ct(M,N)$,*
2. *if $M \Rightarrow cM_1 ... M_n$, for some $n \geq 0$ and $c \in \Psi$, then $N_1, \ldots, N_n$ exist s.t. $N \Rightarrow cN_1 \cdots N_n$ and $M_i \simeq'_\Psi N_i$, $1 \leq i \leq n$.* $\Box$

The difference between the definitions of $\simeq_\Psi$ and $\simeq'_\Psi$ is in clause (1). While $\simeq_\Psi$ requires to test the equality between $\lambda x.M'$ and $\lambda x.N'$ for *all* terms in Λ_Ψ here *one* term — a constant — is enough (also, we do not need clause (3) of Definition 3.1 because the reduction relation is deterministic). The message is that *constants are very powerful*, which will be reinforced by Lemma 6.5.

Theorem 4.9 $\simeq_\Psi <> \simeq'_\Psi$

Proof: There is a simple proof exploiting the encoding into the Higher-Order π-calculus and its theory. A proof inside the λ-calculus is also possible; we shall only sketch it. We have to show that $M\{c/x\} \simeq_\Psi N\{c/x\}$, for $c \notin ct(M,N)$, implies $M\{R/x\} \simeq_\Psi N\{R/x\}$, for any $R \in \Lambda_\Psi$. Without loss of generality assume also 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 stronger version of Lemma 6.5, saying that for each class of symbols P and C and assignment of well-formed operators to the symbols in P, $M \simeq_{P \cup C} N$ implies $M \simeq^P_C N$. The modifications to the proof of Lemma 6.5 are straightforward. Then $M\{c/x\} \simeq_\Psi N\{c/x\}$ implies $M\{c/x\} \simeq^c_{\Psi - \{c\}} N\{c/x\}$, for any choice of operator for c; in particular this holds when $c \Rightarrow R$ is the only behavioural rule of c. Finally, with such a definition of operator for c, we have $c \cong^c_{\Psi - \{c\}} R$; hence $M'\{c/x\} \simeq^c_{\Psi - \{c\}} M'\{R/x\}$ and $N'\{c/x\} \simeq^c_{\Psi - \{c\}} N'\{R/x\}$. From this and $M'\{c/x\} \simeq^c_{\Psi - \{c\}} N'\{c/x\}$, we get $M'\{R/x\} \simeq^c_{\Psi - \{c\}} N'\{R/x\}$ and, by Lemma 3.7, $M'\{R/x\} \simeq_{\Psi - \{c\}} N'\{R/x\}$, as required. $\Box$

On Open Terms. It is a short step to go from $\simeq'_\Psi$ to a characterisation on open terms, with no constants at all. It is perhaps the simplest way to consider the notion of applicative bisimulation on open terms.

Definition 4.10 *The relation* $\overset{o}{\simeq}$ *is the largest symmetric relation on* $\Lambda(\mathcal{X})\times\Lambda(\mathcal{X})$ *s.t.* $M \overset{o}{\simeq} N$ *implies*

1. *if* $M \Rightarrow \lambda x.M'$, $x \notin fv(M,N)$ *then* N' *exists s.t.* $N \Rightarrow \lambda x.N'$ *and* $M' \overset{o}{\simeq} N'$,
2. *if* $M \Rightarrow xM_1 ... M_n$, *for some* $n \geq 0$, *then* $N_1, \ldots, N_n$ *exist s.t.* $N \Rightarrow xN_1 \cdots N_n$ *and* $M_i \overset{o}{\simeq} N_i$, $1 \leq i \leq n$. □

Theorem 4.11 $\simeq'_\Psi <> \overset{o}{\simeq}$. □

The relations $\overset{\sim}{\leftrightarrow}$ and $\overset{o}{\simeq}$ are defined on the same domain of pure λ-terms. From the theorems in this section, by transitivity, we derive that they coincide on closed terms. The correspondence can be strengthened to open terms, so to conclude that:

Corollary 4.12 *The relations* $\overset{\sim}{\leftrightarrow}$ *and* $\overset{o}{\simeq}$ *coincide.* □

5 Full abstraction

In this section we look at $\overset{\sim}{\leftrightarrow}$ denotationally. We show full abstraction for $\overset{\sim}{\leftrightarrow}$ w.r.t. Levy-Longo trees. As a corollary, we get full abstraction w.r.t. the free lazy Plotkin-Scott-Engeler models. Our terminology and notation mainly come from (Ong 1988a). We denote by $\lambda\beta$ the classical (i.e not lazy) formal theory of the λ-calculus, given by the usual axioms α, β and the rules μ, ν, ξ. We use n to range over the set of nonnegative integers $\{0, 1, \ldots\}$ and ω to represent the first ordinal limit. All λ-terms in this section belong to $\Lambda(\mathcal{X})$.

Böhm Trees (BT) are the most popular tree structure in the λ-calculus. However, BT's only correctly express the computational content of λ-terms in a "strict" regime, while they fail to do so in a lazy one. For instance, in a lazy scheme, the terms $\lambda x.\Omega$ and Ω are distinguished, but since *unsolvable* (Barendregt 1984), they have identical BT's. The right notion of tree in a lazy regime is a variant of BT's called Levy-Longo Trees (LT). LT's were introduced in (Longo 1983) — where they were simply called trees — developing an original idea by Levy (1975). For the definition of LT's we need the notions of *proper order* of terms, and of *head reduction*, which we now introduce.

The *order* of a term M expresses the maximum length of the outermost sequence of λ-abstractions to which M is β-convertible; it says how "higher-order" M is. More precisely, M has order n if n is the largest i s.t. $\lambda\beta \vdash M = \lambda x_1 \cdots x_i.N$, for some N. Therefore a term has order 0 if it is not β-convertible to any abstraction. The remaining terms are assigned order ω; they are terms like Ξ which can reduce to an unbounded number of nested abstractions. A term has *proper order* n if it has order of unsolvability n, i.e., after the initial n λ-abstractions it behaves like Ω. Formally,

- M has proper order 0, written $M \in PO_0$, if M has order 0 and there is no $\tilde{N}$ s.t. $\lambda\beta \vdash M = x\tilde{N}$;
- M has proper order n+1, written $M \in PO_{n+1}$, if $N \in PO_n$ exists s.t. $\lambda\beta \vdash M = \lambda x.N$;
- M has proper order ω, written $M \in PO_\omega$, if M has order ω.

A λ-term is either of the form $\lambda\tilde{x}.y\tilde{M}$, or of the form $\lambda\tilde{x}.((\lambda x.M_0)M_1\cdots M_n)$, $n \geq 1$. In the latter, the redex $(\lambda x.M_0)M_1$ is called the *head redex.* If M is a term with a head redex, then $M \longrightarrow_h N$ holds if N results from M by β-reducing its head redex; $\Rightarrow_h$ is the reflexive and transitive closure of $\longrightarrow_h$. The *head reduction* $\Rightarrow_h$ is different from the lazy reduction $\Rightarrow$; a lazy redex is also a head redex, but the other way round may be false, for a head redex can also be located underneath an abstraction. However, we have:

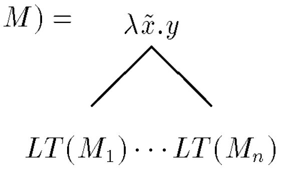
Lemma 5.1

1. *$M \Rightarrow_h \lambda x.N$ iff $M \Rightarrow \lambda x.N'$, for some N' s.t. $N' \Rightarrow_h N$;*
2. *$M \Rightarrow_h x\tilde{N}$ iff $M \Rightarrow x\tilde{N}$;*
3. *$M \Rightarrow_h \lambda x_1.\cdots x_n.y\tilde{N}$ iff there are terms M_i, $1 \leq i \leq n$, s.t. $M \Rightarrow \lambda x_1.M_1\,,\ \ M_i \Rightarrow \lambda x_{i+1}.M_{i+1}$, $1 \leq i < n\,,$ and $M_n \Rightarrow y\tilde{N}$.*

Proof: (1) and (2) hold because both $\Rightarrow$ and $\Rightarrow_h$ progress using the leftmost redex; (3) follows from (1) and (2). $\square$

Definition 5.2 *The* Levy-Longo Tree *of M, written $LT(M)$, is a labelled tree defined inductively as follows:*

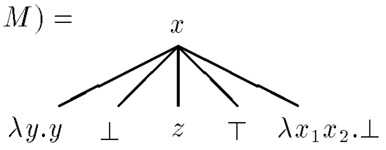
1) $LT(M) = \top$ if $M \in PO_\omega$,
2) $LT(M) = \lambda x_1\cdots x_n.\bot$ if $M \in PO_n$,
3) $LT(M) =$



if $M \Rightarrow_h \lambda\tilde{x}.yM_1\cdots M_n$, $n \geq 0$. $\square$

Example 5.3 *Let $M = x(\lambda y.y)\Omega z\Xi(\lambda x_1x_2.\Omega)$. Then*

$LT(M) =$



$\square$

We shall consider equality of LT's modulo α-conversion.

Theorem 5.4 (full abstraction w.r.t Levy-Longo Trees) *$M \overset{\sim}{\leftrightarrow} N$ iff $LT(M) = LT(N)$.*

Proof: Corollary 4.12 shows that $\overset{\sim}{\leftrightarrow}$ coincides with $\overset{o}{\simeq}$. Using Lemma 5.1 we can show that $\overset{o}{\simeq}$ also coincides with $\overset{o}{\simeq}_h$, defined as the largest relation on $\Lambda(\mathcal{X}) \times \Lambda(\mathcal{X})$ s.t. $M \overset{o}{\simeq}_h N$ implies:

1. $M \in PO_\omega$ iff $N \in PO_\omega$,
2. $M \in PO_n$ iff $N \in PO_n$,

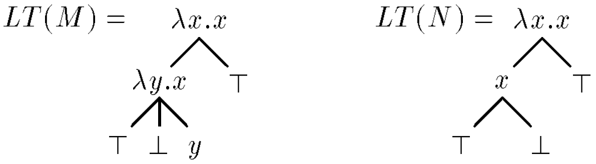
3. If $x_i \notin fv(M,N)$, $1 \leq i \leq n$, then
   $M \Rightarrow_h \lambda x_1 \cdots x_n.yM_1 \cdots M_m$ iff $N \Rightarrow_h \lambda x_1 \cdots x_n.yN_1 \cdots N_m$ and $M_i \overset{o}{\simeq}_h N_i$, $1 \leq i \leq m$.

Finally, it is immediate to see that $\overset{o}{\simeq}_h$ is the LT equality. $\square$

Example 5.5 *Consider the terms*

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

They have been used to prove non full abstraction results for the lazy λ-calculus w.r.t. Abramsky's canonical model (Abramsky and Ong 1989) and w.r.t. Milner's encoding into π-calculus (Milner 1990). This is due to the fact that both in Abramsky's model and in the π-calculus the convergence test ∇ is definable. Such an operator can distinguish between M and N, as $M(\lambda x.\nabla x)$ reduces to an abstraction, whereas $N(\lambda x.\nabla x)$ diverges. However, no pure λ-terms can make the same distinction as can be shown by a case analysis on its order. We can prove $M \not\stackrel{\sim}{\leftrightarrow} N$ by simply observing that their LT's are different:



This gives us a straightforward proof of the non full abstraction of the π-calculus's encoding, i.e., $\simeq \not< \stackrel{\sim}{\leftrightarrow}$. (The analogous result in (Milner 1990) is obtained by implementing ∇ as a π-calculus process, and then exploiting the theory of the π-calculus). $\square$

Observe that the characterisation in terms of $\simeq_C$ and Lemma 3.7 show that λ-observation equivalence is contained in Abramsky's original $\simeq$; the above example shows that the containment is strict.

Theorem 5.4 has an important consequence in terms of λ-models, more precisely the *free lazy Plotkin-Scott-Engeler (PSE) models*. Briefly, a PSE model is defined on top of a PSE algebra. These are combinatory algebras introduced by Engeler (1981), which followed earlier ideas by Plotkin and Scott. PSE algebras are defined in a very natural set theoretic way, the notion of application generalising the classical Myhill-Shepherdson-Roger definition of application in the graph model $\mathcal{P}\omega$. There are two canonical ways to expand a PSE algebra into a PSE model, depending on the choice of the graph function Gr which selects a unique representative $Gr(f)$ for each representable function f: Intuitively, the first choice, called *strict*, selects the "least" representative $Gr^{\perp}(f)$ from the extensionality class of f, and gives rise to the *free strict PSE models*. The second choice selects the "largest" representative, and gives rise to the *free lazy PSE models*. The λ-theory induced by the free strict PSE models is precisely the Böhm Tree theory. The free lazy PSE models enjoy the same property w.r.t. the Levy-Longo Trees. These results have been obtained by Longo (1983) and Ong (1988a). The latter result and Theorem 5.4 show that $\stackrel{\sim}{\leftrightarrow}$ is fully abstract w.r.t. the free lazy PSE models, i.e., if $\Im$ is such a model, then

$$M \stackrel{\sim}{\leftrightarrow} N \text{ iff } \Im \vdash M = N.$$

6 Well-formed operators

We now turn to examine how applicative bisimulation is affected by enrichments of the lambda calculus with well-formed operators. (We recall that an operator is a symbol which comes equipped with reduction rules describing its behaviour; it is well-formed if the rules obey a certain format specified below). In this section, we show that λ-observation equivalent terms cannot be distinguished using well-formed operators. Hence when two pure closed λ-terms are λ-observation equivalent, we can replace them in any context built from well-formed operators without affecting the overall behaviour. In Sections 7 and 8 we shall analyse the complementary question, namely if and when well-formed operators allow us to discriminate terms which are not λ-observation equivalent. First we have to show the format of the rules which define well-formed operators.

6.1 The GV format

We follow Groote and Vaandrager (1992) for the format of the reduction rules for well-formed operators, and hence call it the *Groote-Vaandrager* (GV) format. (It is called *well-founded tyft* format in (Groote and Vaandrager 1992)). Here, we are dealing with a calculus with bound variables, absent in the setting considered by Groote and Vaandrager; therefore their ideas have to be appropriately adapted. We anticipate that the rules describing the operators ∇, $\Diamond$, $\oplus$ and $[]$ in Sections 2 and 3 fit the GV format. Moreover the arguments used in (Groote and Vaandrager 1992) could be reformulated to show that our GV format cannot be generalised in any obvious way without losing the "good behaviour" of the well-formed operators they define, that is to say, without introducing syntactic dependencies of the operators on their arguments. Even though our well-formed operators do not intend to represent *all* such well-behaved operators — for instance the GV format excludes rules with negative premises — they still constitute a very large and interesting class.

We recall that $X, Y, Z, \cdots$ represent *metavariables*, whereas $x, y, \cdots$ are plain λ-variables. A 'Λ_C^P-term' which may also contain metavariables is called a Λ_C^P-*metaterm* (sometimes simply *metaterm*). Formally, the metaterms — ranged over by T and S — are those produced by the following grammar and which do not have free λ-variables:

$$T = c \quad | \quad pT_1 \cdots T_{r(p)} \quad | \quad X \quad | \quad x \quad | \quad \lambda x.T \quad | \quad T_1 T_2,$$

where c is a constant, p an operator with arity $r(p)$, X a metavariable and x a λ-variable. To introduce the GV format, let us look back at the rules for the convergence test operator ∇, namely

$$\frac{X \Downarrow_\lambda Y}{\nabla X \Rightarrow I} \qquad \text{and} \qquad \frac{X \Rightarrow Y}{\nabla X \Rightarrow \nabla Y},$$

where the symbol $\Downarrow_\lambda$ means reduction to an abstraction, i.e.,

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

Thus, the conclusion of a GV rule is of the form $pX_1 \cdots X_{r(p)} \Rightarrow T$, where p is an operator, $r(p)$ its arity and T a metaterm. The premise of the rule may use the reduction relations $\Rightarrow$ and $\Downarrow_\lambda$. In the above example of ∇, the premises have a metavariable both on the left and on the right-hand side; however, in general the left-hand side could be a metaterm. In addition, a premise of a GV-rule may also employ the reduction relation $\Downarrow_c$, meaning reduction to a term with a constant c in head position, i.e.,

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

We impose that the number of premises of a rule is finite — which is not the case in (Groote and Vaandrager 1992) — because we think it is a reasonable assumption and because it facilitates the proof of Lemma 7.2.

Definition 6.1 (well-formed operators and the GV format) *An operator is* well-formed *if it can be described by behavioural rules in* Groote-Vaandrager *(GV) format. A behavioural rule is in GV format if it can be written as*

$$\frac{\{T_i \; \varrho_i \; S_i \mid i \in I\}}{pX_1 \cdots X_{r(p)} \Rightarrow T} \qquad \textit{where}$$

- *I is a finite index set and p an operator,*
- *for all $i \in I$, T_i, S_i and T are metaterms, $\varrho_i \in \{\Rightarrow, \Downarrow_\lambda\} \cup \{\Downarrow_c$ s.t. c is a constant$\}$ and s.t. if $\varrho_i \in \{\Rightarrow, \Downarrow_\lambda\}$ then $S_i = Y_i$, if $\varrho_i = \Downarrow_c$ then $S_i = cY_{i_1} \cdots Y_{i_n}$, for some $i_n \geq 0$,*
- *all mentioned metavariables X_j, Y_i, and Y_{i_k} are distinct from each other, $1 \leq j \leq r(p)$, $i \in I$ and $1 \leq i_k \leq i_n$.*

Moreover the rule has to be non-circular. $\square$

Therefore, following Groote and Vaandrager, we exclude rules which contain circularity. We explain below what circular rules are and why they can be dangerous. In (Groote and Vaandrager 1992) also rules with conclusion of the form $X \Rightarrow T$ are considered, and called *tyxt* rules. We reject this format because in λ-calculus it is necessary that each non β-redex be motivated by an operator p in head position.

Definition 6.2 *A rule with premises $\{T_i \; \varrho_i \; S_i \mid i \in I\}$ is* non-circular *if it is possible to order the index set I s.t. if some metavariable appears both in S_j and in T_i, then $j < i$ holds. Otherwise, a rule is* circular. $\square$

Here is a circular rule:

$$\frac{p_1 Y_2 \Rightarrow Y_1 \qquad\qquad p_2 Y_1 \Rightarrow Y_2}{p_1 X \Rightarrow Y_1}$$

Circular rules can give rise to "odd" transition systems, where the behaviour of an operand can depend on the *syntax* of its operands. This can lead to an equivalence unnaturally more discriminative than λ-observation equivalence, as the following example shows.

Example 6.3 *Let p be an operator defined by the rules (the first one is circular):*

$$(R1)\,\frac{XY \Rightarrow Y}{pX \Rightarrow Y} \qquad\qquad (R2)\; pX \Rightarrow I$$

We have $\lambda x.\Omega \overset{\sim}{\leftrightarrows} \lambda x.(\Omega x)$, but $p(\lambda x.\Omega) \not\lesssim^p p(\lambda x.(\Omega x))$. The latter because we can infer $p(\lambda x.\Omega) \Rightarrow \Omega$ thus

$$(R1)\,\frac{(\beta)\,\dfrac{- - -}{(\lambda x.\Omega)\Omega \Rightarrow \Omega}}{p(\lambda x.\Omega) \Rightarrow \Omega}\ ,$$

but $p(\lambda x.(\Omega x))$ cannot match this reduction. Indeed $p(\lambda x.(\Omega x))$ cannot evolve using (R1) since, for any R, $(\lambda x.(\Omega x))R \Rightarrow \Omega R \neq R$. The term $p(\lambda x.(\Omega x))$ can only evolve to I using (R2), but then $I \not\simeq^{p} \Omega$ (in the example the presence of (R2) is necessary, otherwise $\Omega \simeq^{p} p(\lambda x.(\Omega x)) \simeq^{p} p(\lambda x.\Omega)$). □

We call the class of all well-formed operators Op; and we call the corresponding bisimulation $\simeq^{Op}$ and congruence $\cong^{Op}$, *rich applicative bisimulation* and *rich applicative congruence*, respectively. The latter represents the most discriminating bisimulation-based equivalence obtained using well-formed operators.

6.2 Constants versus operators

A class P of symbols can be viewed as a class of operators in the language Λ^{P} (when an arity and some behavioural rules are specified for each symbol), or as a class of constants in the language Λ_{P}. The verification of $\simeq_{P}$ is simplified by the characterisations obtained in Section 4.2; Lemma 6.5 below shows that this can be exploited to reason about $\simeq^{P}$. For the proof of Lemma 6.5 we use an auxiliary lemma. The *depth* of the proof of a derivation of a transition $M \Rightarrow M'$ is the maximal number of nested steps in such a derivation.

Lemma 6.4 *Let $M, M' \in \Lambda^{P}$. Suppose $M \Rightarrow M'$ is inferred with a derivation of depth n in which some behavioural rule is used. Then there are an operator p and terms $\tilde{M}$ s.t.*

$$M \Rightarrow p\tilde{M} \Rightarrow M' \quad \textit{and}$$

- *the derivation of $M \Rightarrow p\tilde{M}$ does not use any behavioural rule;*
- *the derivation of $p\tilde{M} \Rightarrow M'$ has depth not greater than n.*

Proof: By induction on n. For the inductive part, proceed by a case analysis on the last rule used, which may be *App*, *Trans* or a behavioural rule. We omit the details, which are simple. □

Lemma 6.5 *$M \simeq_{P} N$ implies $M \simeq^{P} N$, for any assignment of well-formed operators to the symbols in P.*

Proof: We are working here with two languages, Λ_{P} and Λ^{P}. Therefore we have two reduction relations, $\Rightarrow_1$ for Λ_{P} and $\Rightarrow_2$ for Λ^{P}, with the corresponding convergence predicates $\Downarrow^{i}_{\lambda}$, $i \in 1, 2$ (since Λ^{P} has no constants, we shall never have to use the other convergence predicate, $\Downarrow_{c}$). An easy fact to prove is that everything derivable with $\Rightarrow_1$ is also derivable with $\Rightarrow_2$, i.e.,

$$M \Rightarrow_1 M' \text{ implies } M \Rightarrow_2 M' . \tag{1}$$

We shall prove that $\mathcal{S} = \{(M, N) \mid M \simeq_{P} N\}$ is a $\simeq^{P}$-bisimulation. We have to check the clauses 3.1(1) and 3.1(3) of Definition 3.1. The two clauses are similar, so we only consider the latter, and proceed by transition induction. So, suppose $M \Rightarrow_2 M'$; we have to find some N' s.t.

$$N \Rightarrow_2 N' \text{ and } M' \simeq_{P} N' . \tag{2}$$

Basic case: The depth of the derivation of $M \Rightarrow_2 M'$ is equal to one. The only interesting case is when a behavioural rule is used, say

$$pX_1 \cdots X_r \Rightarrow_2 T\,. \tag{3}$$

The term T may contain the metavariables $\tilde{X} = \{X_1, \cdots, X_r\}$ plus some other metavariables $\tilde{Z} = \{Z_1, \cdots, Z_m\}$. Now, if rule (3) has been applied to M, there must be terms $\tilde{M}_X = \{M_1, \cdots, M_r\}$ and $\tilde{R} = \{R_1, \cdots, R_m\}$ s.t. $M = pM_1 \cdots M_r$ and $M' = T\{\tilde{M}_X/\tilde{X}, \tilde{R}/\tilde{Z}\}$ (substitution between tuples is defined componentwise). From $M \simeq_P N$ and definition of $\simeq_P$, we infer that $N \Rightarrow_1 pN_1 \cdots N_r$, with $M_i \simeq_P N_i$ for each $1 \le i \le r$. Therefore, by (1), also $N \Rightarrow_2 pN_1 \cdots N_r$ and, using rule (3), if $\tilde{N}_X = \{N_1, \cdots, N_r\}$ then $N \Rightarrow_2 T\{\tilde{N}_X/\tilde{X}, \tilde{R}/\tilde{Z}\}$. Since $\simeq_P$ is a congruence (Proposition 4.6), we have $T\{\tilde{M}_X/\tilde{X}, \tilde{R}/\tilde{Z}\} \simeq_P T\{\tilde{N}_X/\tilde{X}, \tilde{R}/\tilde{Z}\}$, as required by (2).

Inductive case: The depth of the derivation of $M \Rightarrow_2 M'$ is $n > 1$. We distinguish two cases: whether or not the derivation uses a behavioural rule. If it does not, then also $M \Rightarrow_1 M'$ holds; since $M \simeq_P N$, for some N', $N \Rightarrow_1 N' \simeq_P M'$; by (1), we get $N \Rightarrow_2 N' \simeq_P M'$.

The case in which the derivation of $M \Rightarrow_2 M'$ does use behavioural rules is more delicate. By Lemma 6.4, for some p and $\tilde{M}$, we can split the reduction $M \Rightarrow_2 M'$ into

$$M \Rightarrow_2 p\tilde{M} \Rightarrow_2 M'\,, \tag{4}$$

where $M \Rightarrow_2 p\tilde{M}$ does not use any behavioural rule, and $p\tilde{M} \Rightarrow_2 M'$ has a derivation of depth less than or equal to n. Since $M \Rightarrow_2 p\tilde{M}$ does not use any behavioural rule, we also have

$$M \Rightarrow_1 p\tilde{M}\,. \tag{5}$$

Let $\tilde{M} = M_1 \cdots M_r$. From $M \simeq_P N$ and (5), we deduce that there are $N_1 \cdots N_r$ s.t. $N \Rightarrow_1 pN_1 \cdots N_r$ and $M_i \simeq_P N_i$, for all i. Moreover, by (1),

$$N \Rightarrow_2 pN_1 \cdots N_r\,. \tag{6}$$

This matches the reduction $M \Rightarrow_2 pM_1 \cdots M_r$ in (4). We are left with the other reduction in (4), namely $pM_1 \cdots M_r \Rightarrow_2 M'$. We have to find N' s.t.

$$pN_1 \cdots N_r \Rightarrow_2 N' \simeq_P M' \tag{7}$$

This would conclude the proof, because, from (6), (7) and the rule *Trans*, we can infer $N \Rightarrow_2 N' \simeq_P M'$, as required by (2).

We find N' to satisfy (7) by a case analysis on the last rule used in the derivation of $pM_1 \cdots M_r \Rightarrow_2 M'$.

(Rule *App*). In this case, the premise of the rule is $pM_1 \cdots M_{r-1} \Rightarrow_2 M''$ and M' is $M''M_r$. Since $M_i \simeq_P N_i$ and $\simeq_P$ is a congruence, we have $pM_1 \cdots M_{r-1} \simeq_P pN_1 \cdots N_{r-1}$. Since the derivation of $pM_1 \cdots M_{r-1} \Rightarrow_2 M''$ has depth less than or equal to $n-1$, by induction $pN_1 \cdots N_{r-1} \Rightarrow_2 N''$ and $N'' \simeq_P M''$. Using *App* we infer $pN_1 \cdots N_r \Rightarrow_2 N''N_r$ and, by the congruence of $\simeq_P$, $M''M_r \simeq_P N''N_P$. For $N' = N''N_r$ in (7), this concludes the case.

(Rule *Trans*). The premises of the rule are $pM_1 \cdots M_r \Rightarrow_2 M''$ and $M'' \Rightarrow_2 M'$. This case simply requires the application of the inductive hypothesis of the lemma twice.

(Behavioural rules). Let r represent the arity of p. Since Λ^P has no constants, the behavioural rule used is of the form

$$\frac{\{T_i \; \varrho_i \, Y_i \mid i \in I\}}{pX_1 \cdots X_r \Rightarrow T} \tag{8}$$

for $\varrho_i \in \{\Rightarrow, \Downarrow_\lambda\}$. Let $\tilde{Z} = \{Z_1, \cdots, Z_m\}$ be the ordered set of the metavariables different from X_i $(1 \leq i \leq r)$ and Y_i $(i \in I)$, which appear in the terms $\{T_i\}_{i \in I}$ and T; similarly, let $\tilde{X} = \{X_1, \cdots, X_r\}$ and $\tilde{Y} = \{Y_i\}_{i \in I}$. Therefore, the last step of the derivation of $M_1 \cdots M_r \Rightarrow_2 M'$ is

$$\frac{\{M_{T_i} \; \varrho_i \, M_{Y_i} \mid i \in I\}}{pM_1 \cdots M_r \Rightarrow M'} \, .$$

The terms $\tilde{M}_X = \{M_1, \cdots, M_r\}$ have instantiated the metavariables $\tilde{X}$ of (8), the terms $\tilde{M}_Y = \{M_{Y_i}\}_{i \in I}$ the metavariables $\tilde{Y}$ and there are terms $\tilde{R} = \{R_1, \cdots, R_m\}$ which have instantiated the metavariables $\tilde{Z}$. The terms M_{T_i} and M' are obtained from T_i and T, respectively, by substituting their metavariables with the appropriate terms, i.e.,

$$\begin{array}{rrcl} & M_{T_i} & = & T_i\{\tilde{M}_X/\tilde{X}, \tilde{M}_Y/\tilde{Y}, \tilde{R}/\tilde{Z}\} \\ \text{and} & M' & = & T\{\tilde{M}_X/\tilde{X}, \tilde{M}_Y/\tilde{Y}, \tilde{R}/\tilde{Z}\} \, . \end{array}$$

We shall show that there are $\tilde{N}_Y = \{N_{Y_i}\}_{i \in I}$ s.t. if $\tilde{N}_X = \{N_1, \cdots, N_r\}$,

$$\begin{array}{rrcl} & N_{T_i} & = & T_i\{\tilde{N}_X/\tilde{X}, \tilde{N}_Y/\tilde{Y}, \tilde{R}/\tilde{Z}\} \\ \text{and} & N' & = & T\{\tilde{N}_X/\tilde{X}, \tilde{N}_Y/\tilde{Y}, \tilde{R}/\tilde{Z}\} \, , \end{array}$$

then

$$N_{T_i} \; \varrho_i \, N_{Y_i} \text{ and } M_{Y_i} \simeq_P N_{Y_i} \, , \text{ for all } i \in I. \tag{9}$$

This would allow us to use rule (8) also on $pN_1 \cdots N_r$ and infer

$$\frac{\{N_{T_i} \; \varrho_i \, N_{Y_i} \mid i \in I\}}{pN_1 \cdots N_r \Rightarrow N'} \, .$$

Moreover, since M' and N' are obtained from T by instantiating its metavariables with $\simeq_P$ terms, we get $M' \simeq_P N'$, as required by (7). We prove (9) by induction on i; here the non-circularity hypothesis for rule (8) is crucial.

- case i=1. By the non-circularity assumption, T_1 contains at most the metavariables $\tilde{X}$ and $\tilde{Z}$, and $M_{T_1} = T_1\{\tilde{M}/\tilde{X}, \tilde{R}/\tilde{Z}\}$. Take the term $N_{T_1} = T_1\{\tilde{N}/\tilde{X}, \tilde{R}/\tilde{Z}\}$; by congruence for $\simeq_P$, $M_{T_1} \simeq_P N_{T_1}$. Since $M_{T_1} \, \varrho_1 \, M_{Y_1}$ has a shorter inference, by the inductive hypothesis of the lemma N_{Y_1} exists s.t. $N_{T_1} \varrho_1 N_{Y_1}$ and $M_{Y_1} \simeq_P N_{Y_1}$.
- case i+1 . In this case also variables Y_j, for $j \leq i$ may appear in T_{i+1}. But by the induction on i, we know that $M_{Y_j} \simeq_P N_{Y_j}$, so everything follows as in the previous case. □

Now we can conclude that λ-observation equivalence is at least as discriminating as rich applicative congruence.

Corollary 6.6 $\stackrel{\sim}{\leftrightarrow} \, < \, \cong^{Op}$

Proof: Let $M, N \in \Lambda$ with $M \stackrel{\sim}{\leftrightarrow} N$ and C be any Λ^{Op}-context. From Theorems 4.2 and 4.7, we get $M \simeq_{Op} N$ and, from Proposition 4.6, $C(M) \simeq_{Op} C(N)$. Finally, from Lemma 6.5 we derive $C(M) \simeq^{Op} C(N)$. □

7 The power of non-determinism

Corollary 6.6 left open the question whether $\stackrel{\sim}{\leftrightarrow}$ is strictly finer than $\cong^{Op}$, which if true, would signify that after all $\stackrel{\sim}{\leftrightarrow}$ is introducing some superfluous discriminations on λ-terms. We shall prove that this is not the case and that $\stackrel{\sim}{\leftrightarrow}$ and $\cong^{Op}$ represent the same relation (on Λ terms). It therefore inherits simple characterisations from λ-observation equivalence, and a measure of its fineness in term of well-formed operators from rich applicative congruence. The proof shows that it is sufficient to add non-determinism to achieve the same maximal discrimination; later we shall see that non-determinism is also necessary. Observe that the characterisations of $\stackrel{\sim}{\leftrightarrow}$ in terms of constants and open terms in Section 4.2, did not give any indication of what the λ-calculus is lacking in terms of *control mechanism*; indeed, none of them seemed to suggest non-determinism. We shall use an even simpler form of non-determinism than $\oplus$ (unconditional choice), namely the unary operator $\uplus = \oplus\Omega$ described by the rules:

$$(\uplus 1) \quad \uplus M \Rightarrow M \qquad \text{and} \qquad (\uplus 2) \quad \uplus M \Rightarrow \Omega\,.$$

First, we observe that $\simeq^{\uplus}$ is a congruence:

Lemma 7.1 *If P is a class of operators whose behavioural rules have empty premises (i.e., the rules are of the simple form $pX_1 \cdots X_{r(p)} \Rightarrow T$), then $\simeq^{P}$ coincides with its congruence $\cong^{P}$.*

Proof: Following the same proof schema as in Proposition 4.6. □

To prove that $\simeq^{\uplus}$ discriminates at least as much as λ-observation equivalence, the key result is Lemma 7.7. For this, we need the sequence $\{\simeq_{d_n}\}_n$ of approximations of $\simeq_d$ which are inductively defined as follows (we recall that in $\simeq_d$ only one constant is used and that $\simeq_d$ is a characterisation of $\stackrel{\sim}{\leftrightarrow}$ by virtue of Theorem 4.7):

- $\simeq_{d_0} = \Lambda_d \times \Lambda_d$
- $M \simeq_{d_{n+1}} N$ when
  1. if $M \Rightarrow \lambda x.M'$, then N' exists s.t. $N \Rightarrow \lambda x.N'$ and $M'\{R/x\} \simeq_{d_n} N'\{R/x\}$, for all $R \in \Lambda_d$,
  2. if $M \Rightarrow dM_1...M_m$, then $N_1, ..., N_m$ exist s.t. $N \Rightarrow dN_1...N_m$ and $M_i \simeq_{d_n} N_i$, for $1 \leq i \leq m$.

Lemma 7.2 *The relation $\simeq_d$ coincides with $\bigcap_n \simeq_{d_n}$.*

Proof: We only have to show that $\bigcap_n \simeq_{d_n}$ is contained in $\simeq_d$. Take $\mathcal{S} = \{(M, N) \mid (M, N) \in \bigcap_n \simeq_{d_n}\}$; the proof that $\mathcal{S}$ is a $\simeq_d$-bisimulation is straightforward because the terms are deterministic. □

To facilitate the proof of Lemma 7.7, it is also convenient to introduce the equivalence $\approxeq^{\uplus}$, a weakening of $\simeq^{\uplus}$ where the *first* step of the bisimulation only considers convergence. Formally, $M \approxeq^{\uplus} N$ holds when

1. if $M \Rightarrow \lambda x.M'$, then N' exists s.t. $N \Rightarrow \lambda x.N'$ and $\lambda x.M' \simeq^{\uplus} \lambda x.N'$;
2. the converse, i.e., if $N \Rightarrow \lambda x.N'$, then M' exists s.t. $M \Rightarrow \lambda x.M'$ and $\lambda x.M' \simeq^{\uplus} \lambda x.N'$.

The relation $\approxeq^{\uplus}$ is used to prove negative results about $\simeq^{\uplus}$, as $\not\approxeq^{\uplus}$ implies $\not\simeq^{\uplus}$. For a term $M \in \Lambda^{\uplus}$, we write $M \Rightarrow_{\star} N$ if there exists a reduction $M \Rightarrow N$ in which the rule ($\uplus 2$) is not used. Moreover, we say that a generic term M is *divergent*, or *diverges*, if M cannot reduce to a term of the form $\lambda x.N$, $x\tilde{N}$ or $c\tilde{N}$; that is, M behaves like Ω.

Lemma 7.3 *If* $M \Rightarrow_{\star} M'$, *then* $M \approxeq^{\uplus} M'$.

Proof: By showing that M and M' reduce to the same abstractions. Clearly, if $M' \Rightarrow \lambda x.M''$, then also $M \Rightarrow M' \Rightarrow \lambda x.M''$. On the other hand, suppose $M \Rightarrow \lambda x.M''$: Then rule ($\uplus 2$) has not been used because, otherwise, we would have reached a divergent term. This means that $M \Rightarrow_{\star} \lambda x.M''$ and, since $\Rightarrow_{\star}$ is deterministic, also $M' \Rightarrow_{\star} \lambda x.M''$. $\square$

The above lemma fails for $\simeq^{\uplus}$; for instance, $\uplus I \Rightarrow_{\star} I$, but $\uplus I \not\simeq^{\uplus} I$.

Corollary 7.4 *Suppose* $M \Rightarrow_{\star} M'$ *and* $N \Rightarrow_{\star} N'$. *Then*

1. $M \not\approxeq^{\uplus} N$ *implies* $M' \not\approxeq^{\uplus} N'$;
2. $M' \not\approxeq^{\uplus} N'$ *implies* $M \not\approxeq^{\uplus} N$.

Proof: Both assertions follow from Lemma 7.3 and the transitivity of $\approxeq^{\uplus}$. $\square$

In the crucial Lemma 7.7 below, we shall use $\approxeq^{\uplus}$ precisely to exloit Corollary 7.4 in the proof.

Lemma 7.5 $MR \not\approxeq^{\uplus} NR$ *implies* $M \not\approxeq^{\uplus} N$.

Proof: M and N cannot be both divergent, for otherwise $MR \approxeq^{\uplus} \Omega \approxeq^{\uplus} NR$. Therefore one of them converges, say $M \Rightarrow \lambda x.M'$. Suppose N converges too, say $N \Rightarrow \lambda x.N'$, and feed R as input to $\lambda x.M'$ and $\lambda x.N'$. By the definition of $\approxeq^{\uplus}$, we have to show that $M'\{R/x\} \not\simeq^{\uplus} N'\{R/x\}$. We derive this from the hypothesis of the lemma. We have:

$$\begin{array}{lllll} MR & \Rightarrow_{\star} & (\lambda x.M')R & \Rightarrow_{\star} & M'\{R/x\} \\ NR & \Rightarrow_{\star} & (\lambda x.N')R & \Rightarrow_{\star} & N'\{R/x\} \end{array}$$

Since $MR \not\approxeq^{\uplus} NR$, from Corollary 7.4(1), we get $M'\{R/x\} \not\approxeq^{\uplus} N'\{R/x\}$, which implies $M'\{R/x\} \not\simeq^{\uplus} N'\{R/x\}$. $\square$

Lemma 7.6 *Let* $M, N \in \Lambda^{\uplus}(\mathcal{X})$ *with* $fv(M,N) \subseteq \{x\}$; *if* $N \Rightarrow \Omega$ *and* $M \Rightarrow \lambda y.M'$, *and in the latter derivation neither rule* ($\uplus 1$) *nor* ($\uplus 2$) *are 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 input is given, $\lambda x.M$ cannot diverge, whereas $\lambda x.N$ can. $\square$

Some notations. We use $(\lambda x)^n.M$ to abbreviate $\lambda x_1.\cdots.\lambda x_n.M$. We denote by q_n, $n > 0$, the term $(\lambda x)^n.x_n x_1 x_2 \cdots x_{n-1}$. This is usually called the *Böhm permutator* of degree n. Böhm permutators play a pivotal rôle in the so-called *Böhm-out* technique. A variant of them, the $\uplus$-permutators, play a pivotal rôle in Lemma 7.7 below. A term $M \in \Lambda^{\uplus}$ is a *$\uplus$-permutator of degree n* if $M = q_n$ or there exists $0 \leq r < n$ s.t.

$$M = (\lambda x)^r . \uplus \lambda x_{r+1} \cdots \lambda x_n . x_n x_1 \cdots x_{n-1} \, .$$

We also call a function f from the positive integers to $\Lambda^{\uplus}$ a *$\uplus$-permutator function*, if for all n, $f(n)$ is a $\uplus$-permutator of degree n.

Lemma 7.7 *Suppose $M \not\simeq_{d_n} N$, for some n; then there are positive integers m_o, k and a $\uplus$-permutator function f s.t. for all $m > k$, $M\{f(m+m_o)/d\} \not\approxeq^{\uplus} N\{f(m+m_o)/d\}$.*

PROOF: By induction on the least n s.t. $M \not\simeq_{d_n} N$. We shall use M^f to stand for $M\{f(m+m_o)/d\}$. We shall also write Ω^m for a sequence of m occurrences of Ω; so, e.g., $M\Omega^3$ is $M\Omega\Omega\Omega$.

Basic case: $M \not\simeq_{d_1} N$. The only interesting situation is where neither term diverges. There are two cases to consider (their symmetric ones are analogous):

- $M \Rightarrow dM_1 \cdots M_r$ and $N \Rightarrow dN_1 \cdots N_{r'}$ with $r < r'$.

  Take $m_o = r+1$, $k = r'$, and $f(n) = q_n$. We derive $M^f \not\approxeq^{\uplus} N^f$, for all $m > k$, from Lemma 7.5: We prove that $M^f \tilde{R} \not\approxeq^{\uplus} N^f \tilde{R}$, where

  $$\tilde{R} = \Omega^m (\lambda x)^{m+r} . I$$

  We have

  $$\begin{aligned} M^f \tilde{R} \; &\Rightarrow \; q_{(m+r+1)} M_1^f \cdots M_r^f \Omega^m (\lambda x)^{m+r} . I \Rightarrow \\ &\qquad\qquad [\text{after } m+r+1 \text{ applications of the } \beta \text{ rule}] \\ ((\lambda x)^{m+r} . I) M_1^f \cdots M_r^f \Omega^m \; &\Rightarrow \; I \end{aligned}$$

  On the other hand,

  $$\begin{aligned} &N^f \tilde{R} \Rightarrow q_{(m+r+1)} N_1^f \cdots N_{r'}^f \tilde{R} \; = \\ &q_{(m+r+1)} N_1^f \cdots N_{r'}^f \Omega^m (\lambda x)^{m+r} . I \; \Rightarrow \simeq^{\uplus} \Omega \end{aligned}$$

  because being $m + r' \geq m + r + 1 > r'$, the variable x_{m+r+1} of $q_{(m+r+1)}$ is instantiated with Ω.

- $M \Rightarrow \lambda x . M'$ and $N \Rightarrow dN_1 \cdots N_r$, for some r. Fix $m_o = r+1$. The choice of k is irrelevant. Suppose M' is divergent and take $f(n) = q_n$. For any term R, we have

  $$\begin{aligned} M^f R \quad &\Rightarrow \quad (\lambda x . M')^f R \approxeq^{\uplus} (\lambda x . \Omega) R \quad [\text{because } M' \text{ is divergent}] \\ &\Rightarrow \quad \Omega \end{aligned}$$

  whereas

  $$N^f R \Rightarrow q_{(m+r+1)} N_1^f \cdots N_r^f R \Rightarrow \lambda x . L, \quad \text{for some } L,$$

since $m > 0$. Hence $M^f R \not\approxeq^{\uplus} N^f R$ and, by Lemma 7.5, also $M^f \not\approxeq^{\uplus} N^f$.

If M' is not divergent, it reduces to a term which is either of the form $d\tilde{M}$ or $\lambda y.M''$ or $x\tilde{M}$. We show that in each of these cases, we can choose f s.t. N^f reduces to an abstraction which is $\not\simeq^{\uplus}$ with $\lambda x.(M')^f$; hence, by definition of $\approxeq^{\uplus}$, $M^f \not\approxeq^{\uplus} N^f$.

- If $M' \Rightarrow d\tilde{M}$, take $f(n) = \uplus q_n$. We have:

  $$\begin{array}{lll} N^f \Rightarrow \uplus q_{(m+r+1)} N_1^f \cdots N_r^f & \Rightarrow & [\text{using } (\uplus 1)] \\ q_{(m+r+1)} N_1^f \cdots N_r^f & \Rightarrow \lambda x.(\lambda y.N') & [\text{since } m > 0] \end{array}$$

  for some N'. Now we can derive $\lambda x.(\lambda y.N') \not\simeq^{\uplus} \lambda x.(M')^f$ applying Lemma 7.6, since

  $$(M')^f = \uplus q_{(m+r+1)} \tilde{M}^f \Rightarrow \Omega \qquad [\text{using } (\uplus 2)].$$

- If $M' \Rightarrow \lambda y.M''$, define for $n > r$

  $$f(n) = (\lambda x)^{r+1}.\uplus \lambda x_{(r+2)} ... \lambda x_n . x_n x_1 ... x_{(n-1)}.$$

  We have $N^f \Rightarrow \lambda x.\uplus \tilde{N}'$, for some $\tilde{N}'$. Again, we can use Lemma 7.6 to derive $\lambda x.(M')^f \not\simeq^{\uplus} \lambda x.\uplus \tilde{N}'$, since $\uplus \tilde{N}' \Rightarrow \Omega$ and $(M')^f \Rightarrow \lambda y.(M'')^f$ without requiring the rule $(\uplus 1)$ or $(\uplus 2)$.
- Finally, if $M' \Rightarrow x\tilde{M}$, take $f(n) = q_n$. We have $N^f \Rightarrow \lambda x.(\lambda y.N')$, for some N'. Then $\lambda x.(x\tilde{M}^f)$ and $\lambda x.(\lambda y.N')$ can be distinguished on input Ω, as the former will diverge, whereas the latter will not.

Inductive case: $M \not\simeq_{d_{n+1}} N$. There are two cases to look at.

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

  By induction there are m'_o, k', f' s.t.

  $$\text{for all } m > k', \quad M_i\{f'(m + m'_0)/d\} \not\approxeq^{\uplus} N_i\{f'(m + m'_0)/d\} \tag{10}$$

  Let $m_o = max\{m'_o, r + 1\}$, $k = k'$, $f = f'$. We have to demonstrate that $M^f \not\approxeq^{\uplus} N^f$. For this, we use Lemma 7.5 and we prove that $M^f \tilde{R} \not\approxeq^{\uplus} N^f \tilde{R}$, where

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

  By Corollary 7.4(2), it is enough to show that $M^f \tilde{R} \Rightarrow_\star M_i^f$ and $N^f \tilde{R} \Rightarrow_\star N_i^f$ since, from (10) and definitions of m_o, k, and f we can infer $M_i^f \not\approxeq^{\uplus} N_i^f$. We have:

  $$\begin{array}{l} M^f \tilde{R} \Rightarrow f(m + m_o) M_1^f \cdots M_r^f \Omega^{(m+m_o-r-1)} (\lambda x)^{(m+m_o-1)}.x_i \Rightarrow_\star \\ \qquad\qquad [\text{after } m + m_o\ \beta\text{-reductions and possibly one application of the rule } (\uplus 1)] \\ ((\lambda x)^{(m+m_o-1)}.x_i) M_1^f \cdots M_r^f \Omega^{(m+m_o-r-1)} \Rightarrow_\star M_i^f \quad [\text{after } m + m_o - 1\ \beta\text{-reductions}] \end{array}$$

  and similarly:

  $$\begin{array}{l} N^f \tilde{R} \Rightarrow f(m + m_o) N_1^f \cdots N_r^f \tilde{R}_m \Rightarrow_\star \\ ((\lambda x)^{(m+m_o-1)}.x_i) N_1^f \cdots N_r^f \Omega^{(m+m_o-r-1)} \Rightarrow_\star N_i^f . \end{array}$$

- $M \Rightarrow \lambda x.M'$, $N \Rightarrow \lambda x.N'$ and R exists s.t. $M'\{R/x\} \not\simeq_{d_n} N'\{R/x\}$.

  By hypothesis m'_o, k' and f' exist s.t.

$$\text{for all } m > k', \quad (M'\{R/x\})\{f'(m+m'_o)/d\} \not\cong^{\uplus} (N'\{R/x\})\{f'(m+m'_o)/d\}\,. \tag{11}$$

  Take $m_o = m'_o$, $k = k'$ and $f = f'$. Again, we show $M^f \not\cong^{\uplus} N^f$ using Lemma 7.5. In this case, we prove that $M^f R^f \not\cong^{\uplus} N^f R^f$. We have:

$$\begin{array}{lllll} M^f R^f & \Rightarrow_\star & (\lambda x.M')^f R^f & \Rightarrow_\star & (M'\{R/x\})^f \\ N^f R^f & \Rightarrow_\star & (\lambda x.N')^f R^f & \Rightarrow_\star & (N'\{R/x\})^f \end{array}$$

  From (11) and definitions of m_o, k and f, we get $(M'\{R/x\})^f \not\cong^{\uplus} (N'\{R/x\})^f$, hence, by Corollary 7.4(2), $M^f R^f \not\cong^{\uplus} N^f R^f$. □

Theorem 7.8 $\overset{\sim}{\leftrightarrow} <> \simeq^{\uplus}$.

PROOF: From Corollary 6.6 we know that $\overset{\sim}{\leftrightarrow} < \simeq^{Op}$, and by Lemma 3.7, $\simeq^{Op} < \simeq^{\uplus}$. It remains to prove that $\not\overset{\sim}{\leftrightarrow} < \not\simeq^{\uplus}$. Suppose $M, N \in \Lambda$ and $M \not\overset{\sim}{\leftrightarrow} N$. Since $\overset{\sim}{\leftrightarrow} <> \simeq_d$ (Theorems 4.2 and 4.7), also $M \not\simeq_d N$. By Lemma 7.2 there is an n s.t. $M \not\simeq_{d_n} N$; hence by Lemma 7.7 $M \not\cong^{\uplus} N$ (remember that $M, N \in \Lambda$, therefore they cannot contain constants), which implies $M \not\simeq^{\uplus} N$. □

Example 7.9 *We show how non-determinism can provide the discriminating power given by the convergence test. Let M, N be the terms in Example 5.5. We have seen that $M \simeq N$, but $M \not\simeq^{\nabla} N$. We show that $M \not\simeq^{\uplus} N$. When feeding $R = \uplus K$ as input, for $K = \lambda xy.x$, we have*

$$\begin{array}{rl} MR & \Rightarrow \\ R(\lambda y.(R\Xi\Omega y))\Xi & \Rightarrow \\ K(\lambda y.(R\Xi\Omega y))\Xi & \Rightarrow \\ \lambda y.(R\Xi\Omega y)\,. & \end{array}$$

Then the only possible reduction for NR to match this move — leading to an abstraction — is:

$$\begin{array}{rl} NR & \Rightarrow \\ R(R\Xi\Omega)\Xi & \Rightarrow \\ K(R\Xi\Omega)\Xi & \Rightarrow \\ R\Xi\Omega & \Rightarrow \\ K\Xi\Omega & \Rightarrow \Xi\,. \end{array}$$

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 converges. □

From Lemma 3.7, Corollary 6.6 and Theorem 7.8 we can conclude that:

Corollary 7.10 *Suppose that P is a class of operators containing $\uplus$; then $\overset{\sim}{\leftrightarrow} <> \simeq^P <> \cong^P$.* □

In particular, each pair of elements from $S = \{\overset{\sim}{\leftrightarrow},\ \cong^{Op},\ \simeq^{Op},\ \cong^{\uplus},\ \simeq^{\uplus}\}$ are in the relation $<>$. We can also add $\simeq^{\oplus}$, $\cong^{\oplus}$, $\simeq^{\|}$ and $\cong^{\|}$ to the set S, since $\uplus$ is definable from $\oplus$ or $\|$. For the former, take $\uplus M = \Omega \oplus M$. For the latter, take $\uplus M = ((\lambda x.\Omega)\|(\lambda x.M))R$, where R is any term and x is not free in M; we have:

$$\begin{array}{lll} ((\lambda x.\Omega)\|(\lambda x.M))R & \Rightarrow\ (\lambda x.\Omega)R & \Rightarrow \Omega \\ ((\lambda x.\Omega)\|(\lambda x.M))R & \Rightarrow\ (\lambda x.M)R & \Rightarrow M \end{array}$$

which are the same derivatives as those of $\uplus M$.

Remark 7.11 The non-determinism of $\simeq^{\uplus}$ only offers the extra possibility of divergence. Therefore, the equivalence $\simeq^{\uplus}$ and, consequently, the assertion of Corollary 7.10, do not change if we replace clause (3) of Definition 3.1 with the weaker clause

$$M \Uparrow \text{ implies } N \Uparrow,$$

where $M \Uparrow$ should be read "M can reduce to a divergent term". This new definition makes $\simeq^{\oplus}$ appear fairly similar to the equivalence adopted in (Ong 1993).

8 Church-Rosser operators

Definition 8.1

- *The terms M and N are* confluent *if there is R s.t. $M \Rightarrow R$ and $N \Rightarrow R$;*
- *A term M is* Church-Rosser *if whenever $M \Rightarrow N$ and $M \Rightarrow R$, then N and R are confluent;*
- *A class P of operators is* Church-Rosser *if each $M \in \Lambda^P$ is Church-Rosser.*

Observe that the class of *deterministic* operators, roughly the operators for which the one-step reduction relation is deterministic, is Church-Rosser. We shall show that Church-Rosser operators do not give full discriminating power. In the remainder of this section, we denote a generic Church-Rosser class of operators by CR.

Lemma 8.2 *Let $M, N \in \Lambda^{CR}$ be confluent and $M \Rightarrow \lambda x.R$; then also $N \Rightarrow \lambda x.R$.*

PROOF: By filling a simple diagram. □

Lemma 8.3 *Let $M \in \Lambda^{CR}$ and $M \Rightarrow M'$; then $M \cong^{CR} M'$.*

PROOF: Take

$$\mathcal{S} = \{(C(M), C(N)) \mid C \textit{ is a } \Lambda^{CR}\textit{-context and } M, N \text{ are confluent } \}$$

The proof that $\mathcal{S}$ is a $\simeq^{CR}$ bisimulation is by induction on the length of the inference used to derive a reduction for $C(M)$ or $C(N)$. In the case when $C = [\,]C_1 \cdots C_n$, for $n \geq 0$, Lemma 8.2 is needed. □

Corollary 8.4 *In Λ^{CR} the following conditional η-rule holds:*

$$M \Downarrow_{\lambda} \textit{ implies } \lambda y.(My) \cong^{CR} M$$

Proof: By hypothesis, $M \Rightarrow \lambda x.M'$. By repeated applications of Lemma 8.3 we have $\lambda y.(My) \cong^{CR} \lambda y.((\lambda x.M')y) \cong^{CR} \lambda y.M'\{y/x\} = \lambda x.M' \cong^{CR} M$. $\square$.

Theorem 8.5 $\simeq^{CR} \not< \simeq^{Op}$

Proof: Consider $M = \lambda x.(xx)$ and $N = \lambda x.(x\lambda y.(xy))$; M and N have different LT's and therefore $M \not\overset{\sim}{\leftrightarrow} N$. By Corollary 7.10, also $M \not\simeq^{Op} N$. However $M \simeq^{CR} N$. For this, we have to prove that for each $R \in \Lambda^{CR}$, $RR \simeq^{CR} R\lambda y.(Ry)$. There are two cases to consider according to whether R is convergent or not. If R is convergent, then $RR \simeq^{CR} R\lambda y.(Ry)$ using Corollary 8.4; if not then $RR \simeq^{CR} \Omega \simeq^{CR} R\lambda y.(Ry)$. $\square$

Since the parallel convergence operator $\Diamond$ is Church-Rosser, Theorem 8.5 also proves that $\overset{\sim}{\leftrightarrow}$ is finer than the relation $\simeq^{\Diamond}$ considered by Abramsky (1987). This means that Abramsky's canonical domain, which is fully abstract for $\simeq^{\Diamond}$, is not a sound model for $\overset{\sim}{\leftrightarrow}$. The same thing is true for the equivalences and their fully abstract canonical domains defined by Boudol (1990 and 1991), since on pure λ-terms they coincide with $\simeq^{\Diamond}$.

9 Conclusions and future work

The purpose of this paper was to study the effect, on the equivalence between λ-terms, of the use of contexts richer than the pure λ-calculus contexts, and possibly involving parallelism. We have concentrated on Abramsky's *lazy λ-calculus* (Abramsky 1989), following two directions. In the first, the λ-calculus has been studied within a process calculus by examining the equivalence $\overset{\sim}{\leftrightarrow}$ induced by Milner's encoding into the π-calculus. We have started from the characterisation of $\overset{\sim}{\leftrightarrow}$ in (Sangiorgi 1992), which uses a variant of Abramsky's applicative bisimulation on a λ-calculus enriched with a countable infinite set of constants. We have derived a few equivalent operational characterisations, from which we have been able to prove full abstraction of $\overset{\sim}{\leftrightarrow}$ w.r.t. Levy-Longo Trees.

In the second direction, we have examined applicative bisimulation when the λ-calculus is augmented with well-formed operators, symbols equipped with behavioural rules obeying the GV format. The maximal discrimination on pure λ-terms induced by such operators is achieved when all of them are present in the calculus. We have proved that a simple non-deterministic operator is enough to obtain the same discrimination, whereas Church-Rosser operators are not. Moreover, we have showed that it coincides with the discrimination given by $\overset{\sim}{\leftrightarrow}$. We conclude that:

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

Thus, because $\oplus$ and $[]$ are natural non-deterministic operators, $\Lambda^{\oplus}$ and $\Lambda^{[]}$ become particularly attractive languages. All this is relevant when investigating the integration of concurrent and functional programming, the embedding of the λ-calculus into a concurrent language or simply the addition of parallel operators to the λ-calculus. For instance, it tells you that it is sound to replace $\simeq^{\oplus}$- or $\simeq^{[]}$-terms in contexts containing concurrent features. Moreover, when applied to the problem of the encoding of the lazy λ-calculus as tackled in (Milner 1990, 1991), this leads to our claim that:

$\Lambda^{\oplus}$ and $\Lambda^{[\,]}$ are very appealing forms of λ-calculus to encode within a process calculus.

For instance, extending Milner's encoding in (Milner 1991) to an encoding for $\Lambda^{\oplus}$ is immediate, simply add the clause[1]

$$[\![M \oplus N]\!] = (a)\big([\![M]\!]\langle a\rangle \oplus [\![N]\!]\langle a\rangle\big)\,.$$

We believe that, following our study conducted here and in (Sangiorgi 1992), it is possible to prove an exact correspondence on $\Lambda^{\oplus}$ between applicative bisimulation and π-calculus's observation equivalence, i.e.,

$$\text{for all } \ M, N \in \Lambda^{\oplus}, \ \ M \simeq^{\oplus} N \Leftrightarrow [\![M]\!] \approx [\![N]\!]$$

(for the pure closed λ-terms the result follows from Corollary 7.10).

In Section 3 we pointed out the problem of the congruence of $\simeq^{P}_{C}$, which we could only prove in a few special cases. Unfortunately, the general congruence theorem proved by Ong (1992) does not help, because the treatment of non-determinism in (Ong 1992) is different from ours. We also leave for future research the examination of classes of operators other than those describable with GV rules. (It is intended, however, that such operators should always be well-behaved, i.e., their behaviour should not depend upon the syntax of their operands). For instance, one might consider rules with negative premises, as done by Bloom, Istrail and Meyer (1988) and Groote (1989), for process algebras. We would like to know whether $\overset{\sim}{\leftrightarrow}$ remains invariant for operators defined from these rules, as we have shown to happen with the GV rules.

An interesting issue is whether the work on canonical models and domain logics carried out in (Abramsky 1987, Boudol 1990 and 1991) can be reformulated in terms of $\overset{\sim}{\leftrightarrow}$ or $\simeq^{\oplus}$. For instance, would the addition of a negation operator turn the logics considered by Abramsky and Boudol into suitable logics for $\simeq^{\oplus}$?

This paper shows what should be added to the (lazy) λ-calculus so that its applicative bisimulation is preserved by the encoding into the π-calculus. The natural complement of such a study is to examine what *it is necessary to eliminate from the π-calculus* to bring it down to the same level as the (pure) λ-calculus. The question is: what exactly makes the π-calculus more discriminating?

Finally we would like to extend the work to other evaluation strategies; in particular the call-by-value, the other strategy encoded by Milner (1990) into π-calculus.

Acknowledgments

I would like to thank Gerard Boudol, Eugenio Moggi, Luke Ong, Peter Sewell, David N. Turner and Andrew Wilson for their comments. A special thank to Robin Milner, for many interesting discussions on the topics of the paper. The suggestions of the two anonymous referees have been very useful.

---

[1] The encoding of a λ-term is parametric over a name a, which represents the channel along which the (encoding of the) λ-term will be accessed; the construct (a) means "abstraction on a", whereas $[\![M]\!]\langle a\rangle$ means "a is applied to $[\![M]\!]$".
The sum operator in the original syntax of π-calculus (Milner *et al.* 1992) is different from $\oplus$ and $[\,]$; however, the process $P \oplus Q$ is definable from π-calculus's sum $+$ and silent prefix τ, as $\tau.P + \tau.Q$.

References

ABRAMSKY, S. (1987) *Domain Theory and the Logic of Observable Properties*, Ph.D. thesis, University of London.

ABRAMSKY, S. (1989) The Lazy Lambda Calculus, in *Research Topics in Functional Programming*, Turner, D. editor, 65-116, Addison-Wesley.

ABRAMSKY, S. AND ONG, L. (1989) Fully Abstraction in the Lazy Lambda Calculus, Research Report, Dep. of Computing, Imperial College, to appear in *Information and Computation.*

ASTESIANO, E. AND COSTA, G. (1980) Nondeterminism and fully abstract models, *RAIRO*, **14**(4), 323-347.

BARENDREGT, H. (1984) *The Lambda Calculus: Its Syntax and Semantics*, Studies in Logic 103, North-Holland, Revised Edition.

BLOOM, B. AND ISTRAIL, S. AND MEYER, A.R. (1988) Bisimulation can't' be Traced: preliminary report, in *Conference Record of the 15th ACM Symposium on Principle of Programming Languages (POPL)*, 229-239.

BOUDOL, G. (1990) A Lambda Calculus for Parallel Functions, Rapport de Recherche 1231, INRIA-Sophia Antipolis.

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

DE NICOLA, R. AND HENNESSY, M. (1984) Testing Equivalences for Processes, *Theor. Comput. Sci.* **34**, 89-133.

DE NICOLA, R. AND HENNESSY, M. (1987) CCS without τ's, in *Proc. TAPSOFT 87*, LNCS 249, 138-152, Springer Verlag.

ENGELER, E. (1981) Algebras and Combinators, *Algebra Universalis*, **13**, 389-392.

GROOTE, J.F. (1989) Transition System Specifications with negative Premises, in *Proc. CONCUR '90*, Baeten, J.C.M. and Klop, J.W. editors, LNCS 458, 332-341, Springer Verlag.

GROOTE, J.F., AND VAANDRAGER F.W. (1992) Structured Operational Semantics and Bisimulation as a Congruence, *Information and Computation*, **100**, 202-260.

JAGADEESAN, R. AND PANANGADEN, P. (1990) A Domain Theoretic Model for a Higher-order Process Calculus, in *Proc. ICALP*, LNCS 443, 181-194, Springer Verlag.

LEVY, J.-J (1975) An Algebraic Interpretation of Equality in Some Models of the Lambda Calculus, in *Lambda Calculus and Computer Science Theory*, Böhm, C. editor, LNCS 37, 147-165, Springer Verlag.

DE'LIGUORO U., AND PIPERNO, A. (1992) Must Preorder in Non-deterministic Untyped λ-calculus, in *Proc. CAAP*, Raoult, E.C. editor, LNCS 581, 203-220, Springer Verlag.

LONGO, G. (1983) Set Theoretical Models of Lambda Calculus: Theory, Expansions and Isomorphisms, *Annales of Pure and Applied Logic*, **24**, 153-188.

Milner, R. (1989) *Communication and Concurrency*, Prentice Hall.

Milner, R. (1990) Functions as Processes, TR. 1154, INRIA, Sophia Antipolis, Final version in *Journal of Math. Struct. in Comp. Science*, **2**, 119-141, 1992.

Milner, R. (1991), The polyadic π-calculus: a tutorial, TR. ECS–LFCS–91–180, Dep. of Computer Science, University of Edinburgh. To appear in the *Proceedings of the International Summer School on Logic and Algebra of Specification*, Marktoberdorf, August 1991

Milner, R. and Parrow, J.G. and Walker, D.J. (1992) A Calculus of Mobile Processes, Parts I and II, *Information and Computation*, **100**, 1-77.

Ong, L. (1988) Fully Abstract Models of the Lazy Lambda Calculus, in proc. *29th Conf. Foundations of Computer Science*, 368-376, The Computer Society Press.

Ong, L. (1988a) *The Lazy Lambda Calculus: an Investigation into the Foundations of Functional Programming*, PhD thesis, University of London. Also Prize Fellowship Dissertation, Trinity College, Cambridge.

Ong, L. (1992) Concurrent Lambda Calculus and a General Precongruence Theorem for Applicative Bisimulation, working draft.

Ong, L. (1993) Non-Determinism in a Functional Setting, in *Proc. of eight annual IEEE Symposium on Logic in Computer Science* (LICS '93), IEEE Computer Society Press.

Park, D. (1981) Concurrency and Automata on Infinite Sequences, LNCS 104, Springer Verlag.

Sangiorgi, D. (1992) *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*, Ph.D. thesis CST–99–93, Department of Computer Science, University of Edinburgh. Also published as ECS–LFCS–93–266.

Sangiorgi, D. (1993) An Investigation into Functions as Processes, to appear in *Proc. Math. Foundations of Program Semantics '93* (New Orleans, April 1993).

Sieber, K. (1993) Call-by-Value and Nondeterminism, in *Proc. International Conference on Typed Lambda Calculi and Applications*, Bezem, M. and Groote, J.F. editors, LNCS 664, Springer Verlag.