INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Lazy functions and mobile processes

Davide Sangiorgi

N° 2515

April 1995

PROGRAMME 2

Calcul symbolique,
programmation
et génie logiciel

Rapport de recherche

1995

ISSN 0249-6399

Lazy functions and mobile processes

Davide Sangiorgi

November 1994

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet MEIJE

Rapport de recherche n ° 2515 — April 1995 — 28 pages

Abstract: This paper continues the study of Milner's encoding of the lazy λ-calculus into the π-calculus [Mil90]. The encoding is shown to give rise to a λ-model in which, in accordance with the theory of the lazy λ-calculus, conditional extensionality holds. However, the model is not fully abstract. To obtain full abstraction, the operational equivalence on λ-terms (*applicative bisimulation*) is refined. The new relation, called *open applicative bisimulation*, allows us to observe some internal structure of λ-terms, and coincides with the *Lévy-Longo Tree* equality.

Milner's encoding is presented on a sublanguage of the π-calculus similar to those proposed by Boudol [Bou92], Honda and Tokoro [HT92]. Some properties of bisimulation on this sublanguage are demonstrated and used to simplify a few proofs in the paper. For instance, *ground bisimulation*, a form of bisimulation where no name instantiation on input actions is required, is proved to be a congruence relation; as a corollary, various π-calculus bisimilarity equivalences (ground, late, early, open) are shown to coincide on this sublanguage.

Key-words: Bisimulation, π-calculus, λ-calculus, lazy λ-calculus, λ-model, full abstraction, Levy-Longo Trees

(Résumé : tsvp)

Work has been supported by the ESPRIT BRA Project 6454 CONFER.

Unité de recherche INRIA Sophia-Antipolis
2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex (France)
Téléphone : (33) 93 65 77 77 – Télécopie : (33) 93 65 77 65

Fonctions lazy et processus mobiles

Résumé : Cet article reprend l'étude de Milner sur l'encobdage du lazy λ-calculus en π-calculus [Mil90]. Il démontre que l'encodage détermine un λ-modèle pour lequel, conformément à la théorie de lazy λ-calculus l'extensionalité conditionelle vaut. Toutefois, le modèle n'est pas "fully abstract". Pour obtenir une "fully abstraction", l'équivalence opérationelle sur les λ-termes (*bisimulation applicative*) est affinée. La nouvelle relation, appelée *bisimulation applicative ouverte*, permet d'observer certaines structures internes de λ-termes, et coincide avec l'égalité des arbres de Lévy-Longo.

L'encodage de Milner est présenté sur un sous-langage de π-calculus, similaire à ceux proposés par Boudol [Bou92], Honda et Tokoro [HT92]. Certaines propriétés de la bisimulation sur ce sous-langage sont démontrées et utilisées pour simplifier quelques démonstrations dans cet article. Par exemple, nous démontrons que la *ground bisimulation*, une forme de bisimulation pour laquelle aucune instantiation de nom sur les actions d'input n'est requise, est une relation de congruence ; comme corollaire, diverses équivalences de bisimilarité de π-calculus (ground, late, early, open) coincident également sur ce sous-langage.

Mots-clé : Bisimulation, π-calculus, λ-calculus, lazy λ-calculus, λ-modèle, full abstraction, arbres de Lévy-Longo.

1 Introduction

In [Mil90] Milner examines the encoding of the λ-calculus into the π-calculus [MPW92]; the former is the universally accepted basis for computations with *functions*, the latter aims at being its counterpart for computations with *processes*. More precisely, Milner shows how the evaluation strategies of the *lazy λ-calculus* and of (a weak form of) *call-by-value λ-calculus* [Abr89, Plo75] can be faithfully mimicked. The characterisation of the equivalence induced on λ-terms by the encodings is left as an open problem. It also remains to be studied which kind of λ-calculus model — if any — can be constructed from the process terms. These are the main questions tackled in this paper.

A deep comparison between a process calculus and the λ-calculus is interesting for several reasons; indeed, virtually all proposals for process calculi with the capability of treating — directly or indirectly — processes as first class objects have incorporated attempts at embedding the λ-calculus [Bou89, Tho90]. From the process calculus point of view, it is a significant test of expressiveness, and helps in getting deeper insight into its theory. From the λ-calculus point of view, it provides the means to study λ-terms in contexts other than purely sequential ones, and with the instruments available in the process calculus. For example, an important behavioural equivalence upon process terms gives rise to an interesting equivalence upon λ-terms. Moreover, the relevance of those λ-calculus evaluation strategies which can be efficiently encoded is strengthened. More practical motivations for describing functions as processes are to provide a semantic foundation for languages which combine concurrent and functional programming and to develop parallel implementations of functional languages.

We shall focus on Milner's lazy λ-calculus encoding. This is the simplest encoding of the λ-calculus into the π-calculus we are aware of. It also seems "canonical" in the sense of being the "natural" encoding of the lazy strategy. (By contrast, a few variants of the call-by-value strategy have been considered — two of them in Milner's original paper [Mil90] — and it is not clear which one should be preferred.) Below, Milner's encoding of the lazy λ-calculus is simply called "Milner's encoding".

The lazy λ-calculus was proposed by Abramsky [1] and motivated by the practice of functional programming implementations; thus, for instance, reductions inside abstractions are forbidden. Abramsky also equipped the lazy λ-terms with a notion of operational equivalence, called *applicative bisimulation*, which follows the bisimulation idea originally formulated by Park and Milner [Par81, Mil89] in concurrency theory.

---

[1] On closed λ-terms, Abramsky's lazy strategy coincides with Plotkin's *call-by-name* strategy [Plo75].

Briefly, our programme is the following. We begin by examining the operational correspondence between source and target terms of Milner's encoding. We then use the encoding to construct a λ-model from the π-calculus processes. The equality on λ-terms induced by the model is the same as that induced, via the encoding, by the behavioural equality adopted on the π-calculus. In accordance with the theory of the lazy λ-calculus, the model validates conditional extensionality. However, the model is not fully abstract. Not surprisingly so: π-calculus is richer — and hence more discriminanting — than the λ-calculus; the latter is purely sequential, whereas the former can, for instance, express parallelism and non-determinism. To obtain full abstraction, we strengthen the operational equivalence on λ-terms. This is achieved using a refinment of applicative bisimulation, called *open applicative bisimulation*, which allows us to observe some internal structure of λ-terms.

Open applicative bisimulation is perhaps the simplest extension of applicative bisimulation to open terms, and it can be easily shown to coincide with the equality determined by *Lévy-Longo Trees*, the lazy variant of *Böhm Trees*. Open applicative bisimulation has also been studied in [San94a]; the results in [San94a] show how to achieve the same discrimination by remaining with closed terms but enriching the λ-calculus with operators, that is symbols equipped with reduction rules describing their behaviour.

A remark about behavioural equivalences for π-calculus: In this paper, we use (weak[2]) bisimulation. However, the main results presented (construction of the λ-model, full abstraction) should be largely independent of this choice. Bisimulation is widely accepted as the finest extensional behavioural equivalence one would like to impose on processes; on the opposite extreme, as the coarsest equivalence, there is *trace equivalence*. We believe that, on processes encoding λ-terms, bisimulation and trace equivalence coincide. This is suggested by the determinism of the encoded lazy λ-terms. It also confirmed by our results and results by Boudol and Laneve [BL94].[3] We have related bisimulation on these processes to the Lévy-Longo Tree equality; Boudol and Laneve have related the Lévy-Longo Tree equality to the *Morris's context-equivalence* of the *lambda calculus with multiplicities*, a form of enriched lazy λ-calculus. Roughly, Morris's context-equivalence equates two terms if they have the same convergence properties in all contexts; in familiar process algebras, it coincides with trace equivalence.

This paper is an improved version of part of the author's PhD thesis [San92] (from which we extracted the extended abstract [San93]). The proofs are different: In [San92], Milner's encoding was factorised through an encoding into the *Higher-Order π-calculus*, an extension of the π-calculus with higher-order features like term-passing, and then all work was carried out from within the Higher-Order π-calculus. In this paper, by contrast, we work within the π-calculus, mainly to take advantage of recent results about its theory which allow us to drastically simplify a few key proofs. Some of this theory is further developed in this paper: For instance, we prove that *ground bisimulation*, a form of bisimulation where no name instantiation on input actions is required, is a congruence relation on a π-calculus sublanguage similar to those proposed by Boudol [Bou92], Honda and Tokoro [HT92]. An immediate consequence is that various π-calculus bisimilarity equivalences (ground, late,

---

[2] In the concurrency terminology, an equivalence is *weak* if it ignores possible internal moves of processes.

[3] By the time we have decided to make a technical report out of this paper, Boudol and Laneve have indeed proved this result ("The λ-calculus, multiplicities and the π-calculus", draft, March 1995.).

early, open) coincide on this sublanguage. Similar results have been independently obtained by Martin Hansen and Josva Kleist [HaKl94].

Another difference with [San92] is that, there, the λ-calculus had to be enriched with symbols called *constants* in order to prove full abstraction for the λ-model. In this paper, we shall be able to avoid constants; thus the statement of the results is simpler and the correspondence with the Lévy-Longo Tree equality more direct. We also hope that the proofs in the π-calculus — as opposed to the Higher-Order π-calculus — will provide a guideline for the study of the encoding of other λ-calculus evaluation strategies, like call-by-value, where intermediate encodings into the Higher-Order π-calculus might not be so helpful.

The paper is self-contained, but some familiarity with the π-calculus would be useful. We introduce the part of π-calculus sufficient for the encoding of the lazy λ-calculus in Section 2. The theory of this calculus we shall need is introduced in Section 3. In Section 4 we review the lazy λ-calculus and Milner's encoding. In Section 5 we examine the operational correspondence between source and target terms of the encoding. In Section 6 we define the λ-model and present some properties of it. In Section 7 we study full abstraction of the model.

2 The mini π-calculus

Throughout the paper, $\mathcal{R}$ ranges over relations. The composition of two relations $\mathcal{R}$ and $\mathcal{R}'$ is written $\mathcal{R}\ \mathcal{R}'$. We often use infix notation for relations; thus $P\ \mathcal{R}\ Q$ means $(P, Q) \in \mathcal{R}$. A tilde represents a tuple. The i-th elements of a tuple $\widetilde{E}$ is referred to as E_i. Our notations are extended to tuples componentwise. Thus $\widetilde{P}\ \mathcal{R}\ \widetilde{Q}$ means $P_i\ \mathcal{R}\ Q_i$ for all components.

2.1. Syntax Small letters $a, b, \ldots, x, y, \ldots$ range over the infinite set of names, and $P, Q, R, \ldots$ over the set $\mathcal{P}r$ of processes. The part of the polyadic π-calculus [Mil91] we shall use, which we shall refer to as the *mini π-calculus*, is built from the operators of inaction, input prefix, output, parallel composition, restriction, and replication:

$$P \;:=\; \mathbf{0} \;\Big|\; a(\widetilde{b}).\,P \;\Big|\; \overline{a}\langle\widetilde{b}\rangle \;\Big|\; P_1 \,|\, P_2 \;\Big|\; \boldsymbol{\nu}\, a\, P \;\Big|\; !\,P\,.$$

When the tilde is empty, the surrounding brackets () and $\langle\rangle$ will be omitted. $\mathbf{0}$ is the inactive process. An input-prefixed process $a(\widetilde{b}).\,P$, where $\widetilde{b}$ has pairwise distinct components, waits for a tuple of names $\widetilde{c}$ to be sent along a and then behaves like $P\{\widetilde{c}/\widetilde{b}\}$, where $\{\widetilde{c}/\widetilde{b}\}$ is the simultaneous substitution of names $\widetilde{b}$ with names $\widetilde{c}$. An output particle $\overline{a}\langle\widetilde{b}\rangle$ emits names $\widetilde{b}$ at a. Parallel composition is to run two processes in parallel. The restriction $\boldsymbol{\nu}\, a\, P$ makes name a local, or private, to P. A replication $!\,P$ stands for a countable infinite number of copies of P in parallel. If $I = \{i_1, \ldots, i_n\}$, then $\Pi_{i \in I} P_i$ abbreviates $P_{i_1} \mid \ldots \mid P_{i_n}$. We assign parallel composition the lowest precedence among the operators.

The most notable features of this language w.r.t. other formulations of the polyadic π-calculus are the absence of the sum and of the match operators (usually written $P + Q$ and $[a = b]P$, respectively), and the limited form of output guarding available, with a null continuation. We chose this language because it has some useful algebraic properties, some of which are reported in Section 3. Similar languages have been studied by Honda and Tokoro [HT92], who call the language *ν-calculus*, and Boudol [Bou92], who calls it *asynchronous π-calculus* — appropriately so, since the emission of a message does not impose any sequencing constraints. The languages in [HT92] and [Bou92], however, only allow *monadic* communications.

2.2. Terminologies and notations In prefixes $a(\widetilde{b})$ and $\overline{a}\langle\widetilde{b}\rangle$, we call a the *subject* and $\widetilde{b}$ the *object*. We use α to range over prefixes. We often abbreviate $\alpha.\mathbf{0}$ as α, and $\boldsymbol{\nu}\, a\, \boldsymbol{\nu}\, b\, P$ as $\boldsymbol{\nu}\, a, b\, P$. An input prefix $a(\widetilde{b}).P$ and a restriction $\boldsymbol{\nu}\, b\, P$ are binders for names $\widetilde{b}$ and b, respectively, and give rise in the expected way to the definition of *free names* of a term, and *alpha conversion*. We identify processes or actions which only differ on the choice of the bound names. The symbol $=$ will mean "syntactic identity modulo alpha conversion". In a statement, we say that a name is *fresh* to mean that it is different from any other name which occurs in the statement or in objects of the statement like processes and substitutions.

Substitutions are of the form $\{\widetilde{b}/\widetilde{c}\}$, and are finite assignments of names to names. We use σ and ρ to range over substitutions. If $\sigma = \{\widetilde{b}/\widetilde{c}\}$, then $\sigma(a)$ is b_i if $a = c_i$, and a if $a \notin \widetilde{c}$; moreover, $P\{\widetilde{b}/\widetilde{c}\}$ is the process obtained from P by replacing the c_i's with the b_i's in parallel. The application of a substitution to a prefix or an action is defined similarly. As usual, bound names of expressions are assumed not to be affected by the application of a substitution. Thus, for all substitutions σ, we have $(\boldsymbol{\nu}\, b\, P)\sigma = \boldsymbol{\nu}\, b\, (P\sigma)$ and, if $\sigma(a) = c$, $(a(\widetilde{b}).\, P)\sigma = c(\widetilde{b}).\, (P\sigma)$. Substitutions have precedence over the operators of the language; $\sigma\rho$ is the composition of substitutions where σ is performed first, therefore $P\sigma\rho$ is $(P\sigma)\rho$.

A context is a process expression with a hole in it. There can be an arbitrary, but finite, number of different holes $[\cdot]_1, \ldots, [\cdot]_n$ in a context, and each of these holes may appear more than once. If C contains at most holes $[\cdot]_1, \ldots, [\cdot]_n$, then we say that C is an n-ary context, and if $\widetilde{P}$ is a vector of n processes, then $C[\widetilde{P}]$ is the process obtained by replacing each occurrence of the hole $[\cdot]_i$ with the i-th component of $\widetilde{P}$.

2.3. Sorting Following Milner [Mil91], we only admit *well-sorted agents*, that is agents which obey a predefined *sorting* discipline in their manipulation of names. The sorting prevents arity mismatching in communications, like in $\overline{a}\langle b, c\rangle \mid a(x).Q$. A sorting is an assignment of *sorts* to names, which specifies the arity of each name and, recursively, of the names carried by that name. We write $a : s$ if name a has sort s. We do not present the formal system of sorting because it is not essential to understand the contents of this paper.

2.4. Transition system and bisimulation We use μ to range over actions. Bound names, free names and names of an action μ are written $\mathrm{bn}(\mu)$, $\mathrm{fn}(\mu)$ and $\mathrm{n}(\mu)$, respectively. The transition system of the calculus is presented in Table 1. We have omitted the symmetric versions of rules `PAR` and `COM`. By our convention for alpha conversion, alpha convertible processes have the same transitions. We often abbreviate $P \xrightarrow{\tau} Q$ with $P \longrightarrow Q$, and write $P \xrightarrow{\widehat{\mu}} Q$ to mean $P \xrightarrow{\mu} Q$ if $\mu \neq \tau$, and $P = Q$ or $P \xrightarrow{\tau} Q$ if $\mu = \tau$.

$$\texttt{INP:}\quad a(\widetilde{b}).\, P \xrightarrow{a(\widetilde{b})} P \qquad\qquad \texttt{OUT:}\quad \overline{a}\langle\widetilde{b}\rangle \xrightarrow{\overline{a}\langle\widetilde{b}\rangle} \mathbf{0}$$

$$\texttt{REP:}\quad \frac{P \mid\, !P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'} \qquad\qquad \texttt{PAR:}\quad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\ \text{if } \mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$$

$$\texttt{COM:}\quad \frac{P \xrightarrow{a(\widetilde{c})} P' \qquad Q \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{d}\,(P'\{\widetilde{b}/\widetilde{c}\} \mid Q')}\ \text{if } \widetilde{d} \cap \mathrm{fn}(P) = \emptyset$$

$$\texttt{RES:}\quad \frac{P \xrightarrow{\mu} P'}{\boldsymbol{\nu}\, a\, P \xrightarrow{\mu} \boldsymbol{\nu}\, a\, P'}\ a \notin \mathrm{n}(\mu) \qquad\qquad \texttt{OPEN:}\quad \frac{P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P'}{\boldsymbol{\nu}\, c\, P \xrightarrow{(\boldsymbol{\nu}\, c, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P'}\ c \in \widetilde{b} - \widetilde{d},\ a \neq c.$$

Table 1: The transition system for the mini-π-calculus

Definition 2.1 (strong ground bisimilarity) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P}r \times \mathcal{P}r$ *is a* strong ground bisimulation *if* $P\ \mathcal{R}\ Q$ *and* $P \xrightarrow{\mu} P'$ *imply that*[4] *there exists* Q' *s.t.* $Q \xrightarrow{\mu} Q'$ *and* $P'\ \mathcal{R}\ Q'$. *Two processes* P *and* Q *are* strongly ground bisimilar, *written* $P \sim Q$, *if* $P\ \mathcal{R}\ Q$ *for some strong bisimulation* $\mathcal{R}$ *.*

Note that in the definition above, no name instantiation is required in the clause for input actions. Therefore, for instance, to check whether two processes $a(\widetilde{b}).\, P$ and $a(\widetilde{b}).\, Q$ are equivalent, we do not have to examine all possible instantiations of names $\widetilde{b}$ in P and Q, but it suffices to check that P and Q alone are equivalent. Surprisingly enough, in the mini π-calculus this form of bisimilarity is preserved by name instantiations and is a congruence relation (Section 3).

We define the weak version of bisimilarity in the usual way. The ‘weak’ arrow $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$, and $\stackrel{\mu}{\Longrightarrow}$ is $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$. Moreover, $P \stackrel{\widehat{\mu}}{\Longrightarrow} Q$ means $P \stackrel{\mu}{\Longrightarrow} Q$ if $\mu \neq \tau$, and $P \Longrightarrow Q$ if $\mu = \tau$.

Definition 2.2 (weak ground bisimilarity) *A symmetric relation* $\mathcal{R} \subseteq \mathcal{P}r \times \mathcal{P}r$ *is a* weak ground bisimulation *if* $P\ \mathcal{R}\ Q$ *and* $P \xrightarrow{\mu} P'$ *imply that there exists* Q' *s.t.* $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ *and* $P'\ \mathcal{R}\ Q'$. *Two processes* P *and* Q *are* weakly bisimilar *(or* observationally equivalent*), written* $P \approx Q$, *if* $P\ \mathcal{R}\ Q$ *for some weak ground bisimulation* $\mathcal{R}$ *.*

Relation $\approx$ is the semantic equality on the π-calculus we are mainly interested in; other relations, like strong bisimilarity and expansion, will serve as auxiliary to $\approx$.

3 Some properties of bisimilarity on the mini π-calculus

3.1. Congruence In this section, $\sim_{\mathrm{L}}$ denotes the original bisimilarity of the π-calculus [MPW92], defined as ground bisimilarity but with the following clause for input actions:

---

[4] We omit the requirement $\mathrm{bn}(\mu) \cap \mathrm{fn}(Q) = \emptyset$, since we work up-to alpha conversion.

"$P \xrightarrow{a(\widetilde{b})} P'$ imply that there exists Q' s.t. $Q \xrightarrow{a(\widetilde{b})} Q'$ and for all $\widetilde{c}$, $P'\{\widetilde{c}/\widetilde{b}\} \mathcal{R} Q'\{\widetilde{c}/\widetilde{b}\}$."

Both strong and weak ground bisimilarity (Definitions 2.1 and 2.2) are preserved by all operators of the language. We only show the argument for weak bisimilarity, whose case is more delicate. Our proof refines, and is inspired by, an idea by Honda, who proved a similar congruence result for the ν-calculus [Hon92]. However, Honda's definition of bisimilarity is not purely ground, since name instantiation is contemplated in the input clause (technically speaking, Honda allows *free input* actions); this makes the congruence w.r.t. parallel composition straightforward. Indeed, Honda's bisimilarity is a variant of the standard bisimilarity $\sim_{\mathrm{L}}$. Moreover, Honda's transition system incorporates certain structural laws, which can then be applied in all contexts independently of the behavioural equivalence adopted.

The crux of the congruence argument is to show that ground bisimilarity is preserved by name instantiation (Proposition 3.5).

Lemma 3.1 *If* $P \xrightarrow{(\boldsymbol{\nu}\, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P'$, *then* $P \sim_L \boldsymbol{\nu}\, \widetilde{d}\,(\overline{a}\langle\widetilde{b}\rangle \mid P')$.

PROOF: By transition induction. One only needs simple algebraic laws for $\sim_{\mathrm{L}}$, like congruence w.r.t. parallel composition. ☐

Lemma 3.2 *If* $P \overset{(\boldsymbol{\nu}\, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle}{\Longrightarrow} P_1 \overset{a(\widetilde{e})}{\Longrightarrow} P_2$, *then* $P \Longrightarrow\sim_L \boldsymbol{\nu}\, \widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\})$.

PROOF: If $P \overset{(\boldsymbol{\nu}\, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle}{\Longrightarrow} P_1$, then there are P_1' and P_1'' s.t. $P \Longrightarrow P_1' \xrightarrow{(\boldsymbol{\nu}\, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P_1'' \Longrightarrow P_1$. By Lemma 3.1,

$$P_1' \sim_{\mathrm{L}} \boldsymbol{\nu}\, \widetilde{d}\,(\overline{a}\langle\widetilde{b}\rangle \mid P_1'') \,. \tag{1}$$

Since $P_1'' \Longrightarrow \overset{a(\widetilde{e})}{\Longrightarrow} P_2$, we have $\boldsymbol{\nu}\, \widetilde{d}\,(\overline{a}\langle\widetilde{b}\rangle \mid P_1'') \Longrightarrow \boldsymbol{\nu}\, \widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\})$; hence, by (1), also $P_1' \Longrightarrow\sim_{\mathrm{L}} \boldsymbol{\nu}\, \widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\})$. From this and $P \Longrightarrow P_1'$, we get $P \Longrightarrow\sim_{\mathrm{L}} \boldsymbol{\nu}\, \widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\})$. ☐

Lemma 3.3

1. *If* $P \xrightarrow{\mu} P'$, *then* $P\sigma \xrightarrow{\mu\sigma} P'\sigma$;
2. *If* $P \overset{\mu}{\Longrightarrow} P'$, *then* $P\sigma \overset{\mu\sigma}{\Longrightarrow} P'\sigma$. ☐

Lemma 3.4

1. *If* $P\sigma \xrightarrow{\mu'} P'$ *with* $\mu' \neq \tau$, *then* $P \xrightarrow{\mu} P''$ *with* $\mu' = \mu\sigma$ *and* $P' = P''\sigma$.
2. *If* $P\sigma \longrightarrow P'$ *then either*
   - *(a)* $P \longrightarrow P''$ *and* $P' = P''\sigma$, *or*
   - *(b)* $P \xrightarrow{(\boldsymbol{\nu}\, \widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} \xrightarrow{c(\widetilde{e})} P''$ *with* $\sigma(a) = \sigma(c)$ *and* $P' \sim_L \boldsymbol{\nu}\, \widetilde{d}\,(P''\{\widetilde{b}/\widetilde{e}\}\sigma)$.

Proof: Another transition induction. We only show the details for assertion (2), in the case when the last rule used is `COM` . Then $P = P_1 \mid P_2$ and $P\sigma = P_1\sigma \mid P_2\sigma$. Moreover, the last step in the derivation of $P\sigma \longrightarrow P'$ is of the form

$$\frac{P_1\sigma \xrightarrow{x(\widetilde{e})} P_1' \qquad P_2\sigma \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{x}\langle\widetilde{z}\rangle} P_2'}{P_1\sigma \mid P_2\sigma \longrightarrow \boldsymbol{\nu}\,\widetilde{d}\,(P_1'\{\widetilde{z}/\widetilde{e}\} \mid P_2')}\,.$$

By assertion (1), there are $a, c, \widetilde{b}$, P_1'' and P_2'' s.t.

$$P_1 \xrightarrow{c(\widetilde{e})} P_1'' \qquad \text{and} \qquad P_2 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P_2''$$

with $\sigma(a) = \sigma(c) = x$, $\sigma(\widetilde{b}) = \widetilde{z}$, $P_1''\sigma = P_1'$ and $P_2''\sigma = P_2'$. If $a = c$, then we can infer

$$P_1 \mid P_2 \longrightarrow \boldsymbol{\nu}\,\widetilde{d}\,(P_1''\{\widetilde{b}/\widetilde{e}\} \mid P_2'') \overset{\text{def}}{=} P''$$

and we have $P''\sigma = P'$, as by assertion (2.a). If $\sigma(a) \neq \sigma(c)$, then we have

$$P_1 \mid P_2 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P_1 \mid P_2'' \xrightarrow{c(\widetilde{e})} P_1'' \mid P_2'' \overset{\text{def}}{=} P''$$

and $\boldsymbol{\nu}\,\widetilde{d}\,(P''\{\widetilde{b}/\widetilde{e}\}\sigma) = \boldsymbol{\nu}\,\widetilde{d}\,(P_1''\{\widetilde{b}/\widetilde{e}\}\sigma \mid P_2''\sigma) = P'\sigma$, as by assertion (2.b). ☐

In the two proofs below, a symmetric relation $\mathcal{R}$ is a *weak ground bisimulation up to restriction and up to* $\sim$ if $P \mathrel{\mathcal{R}} Q$ and $P \xrightarrow{\mu} P''$ imply that there exist $\widetilde{d}$, P' and Q' s.t. $P'' \sim \boldsymbol{\nu}\,\widetilde{d}\,P'$, $Q \overset{\widehat{\mu}}{\Longrightarrow} \sim \boldsymbol{\nu}\,\widetilde{d}\,Q'$ and $P' \mathrel{\mathcal{R}} Q'$. A standard argument shows that if $\mathcal{R}$ is a weak ground bisimulation up to restriction and up to $\sim$ then $\mathcal{R} \subseteq \approx$.

Proposition 3.5 (insensitiveness of $\approx$ to name instantiations) $P \approx Q$ *implies* $P\sigma \approx Q\sigma$.

Proof: We show that

$$\mathcal{R} = \{(P\sigma, Q\sigma) \; : \; P \approx Q\}$$

is a weak ground bisimulation up to restriction and up to $\sim$. Suppose $P\sigma \xrightarrow{\mu'} P''$. We have to find $\widetilde{d}$, P' and Q' s.t.

$$P'' \sim \boldsymbol{\nu}\,\widetilde{d}\,P', \qquad Q\sigma \overset{\widehat{\mu'}}{\Longrightarrow} \sim \boldsymbol{\nu}\,\widetilde{d}\,Q' \qquad \text{and} \qquad P' \mathrel{\mathcal{R}} Q'\,. \tag{2}$$

We distinguish the case when $\mu' \neq \tau$ and $\mu' = \tau$. If $\mu' \neq \tau$, then, by Lemma 3.4(1), $P \xrightarrow{\mu} P_1$ with $\mu' = \mu\sigma$ and $P'' = P_1\sigma$. Since $P \approx Q$, we have $Q \overset{\mu}{\Longrightarrow} Q_1$ with $P_1 \approx Q_1$. Now, by Lemma 3.3, we have $Q\sigma \overset{\mu\sigma}{\Longrightarrow} Q_1\sigma$; for $\widetilde{d} = \emptyset$, $P' \overset{\text{def}}{=} P''$ and $Q' \overset{\text{def}}{=} Q_1\sigma$ this proves (2).

Now, suppose $\mu' = \tau$. According to Lemma 3.4(2), there are two subcases to consider. In the first, we have $P \longrightarrow P_1$ and $P'' = P_1\sigma$; this can be handled as the case $\mu' \neq \tau$ above. In the other subcase, we have $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} P_1 \xrightarrow{c(\widetilde{e})} P_2$ with $\sigma(a) = \sigma(c)$ and $P'' \sim_{\mathrm{L}} \boldsymbol{\nu}\,\widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\}\sigma)$; since $\sim_{\mathrm{L}} \subseteq \sim$, also $P'' \sim \boldsymbol{\nu}\,\widetilde{d}\,(P_2\{\widetilde{b}/\widetilde{e}\}\sigma)$. Since $P \approx Q$, we have $Q \overset{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle}{\Longrightarrow} Q_1 \overset{c(\widetilde{e})}{\Longrightarrow} Q_2$ with $P_2 \approx Q_2$. Let $x = \sigma(a)$ and $\widetilde{z} = \sigma(\widetilde{b})$; by Lemma 3.3, we also have $Q\sigma \overset{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{x}\langle\widetilde{z}\rangle}{\Longrightarrow} Q_1\sigma \overset{x(\widetilde{e})}{\Longrightarrow} Q_2\sigma$. Finally, by Lemmas 3.2 and the inclusion $\sim_{\mathrm{L}} \subseteq \sim$, $Q\sigma \Longrightarrow \sim \boldsymbol{\nu}\,\widetilde{d}\,(Q_2\sigma\{\widetilde{z}/\widetilde{e}\}) = \boldsymbol{\nu}\,\widetilde{d}\,(Q_2\{\widetilde{b}/\widetilde{e}\}\sigma)$. For $P' \overset{\text{def}}{=} P_2\{\widetilde{b}/\widetilde{e}\}\sigma$ and $Q' \overset{\text{def}}{=} Q_2\{\widetilde{b}/\widetilde{e}\}\sigma$, this proves (2). ☐

Corollary 3.6 (congruence of $\approx$) *$P \approx Q$ implies:*

1. $P \mid R \approx Q \mid R$;
2. $a(\widetilde{b}).\, P \approx a(\widetilde{b}).\, Q$;
3. $!P \approx !Q$;
4. $\boldsymbol{\nu}\, b P \approx \boldsymbol{\nu}\, b Q$.

PROOF: By exhibiting the appropriate bisimulation relations. We only sketch the argument for assertion (1). One shows that the the set of all pairs of the form $(P \mid R, Q \mid R)$, with $P \approx Q$, is a weak ground bisimulation up to restriction. The most interesting case to look at is how $Q \mid R$ can match an interaction between P and R where P performs the input. Thus, suppose $P \mid R \longrightarrow \boldsymbol{\nu}\, \widetilde{d}\,(P'\{\widetilde{b}/\widetilde{e}\} \mid R')$, for some P' and R' s.t. $P \xrightarrow{a(\widetilde{e})} P'$ and $R \xrightarrow{(\boldsymbol{\nu}\,\widetilde{d})\,\overline{a}\langle\widetilde{b}\rangle} R'$. Since $P \approx Q$, there is Q' s.t. $Q \overset{a(\widetilde{e})}{\Longrightarrow} Q' \approx P'$. Hence $Q \mid R \Longrightarrow \boldsymbol{\nu}\, \widetilde{d}\,(Q'\{\widetilde{b}/\widetilde{e}\} \mid R')$. Since, by Proposition 3.5, $\approx$ is preserved by substitutions, we have $P'\{\widetilde{b}/\widetilde{e}\} \approx Q'\{\widetilde{b}/\widetilde{e}\}$. This is enough, because $\mathcal{R}$ is a bisimulation up to restriction. ☐

Let $\approx_{\mathrm{L}}$ be the weak version of $\sim_{\mathrm{L}}$, defined using the weak arrow $\overset{\widehat{\mu}}{\Longrightarrow}$ in the usual way. This is the weak bisimilarity proposed in [MPW92].

Corollary 3.7 *Relations $\approx_L$ and $\approx$ coincide.*

PROOF: For the containment $\approx_{\mathrm{L}} \subseteq \approx$, one shows that $\approx_{\mathrm{L}}$ is a weak ground bisimulation; the opposite containment can be established similarly, if one uses the fact that $\approx$ is closed under substitutions. ☐

In π-calculus literature, relations $\sim_{\mathrm{L}}$ and $\approx_{\mathrm{L}}$ are sometimes called *late* bisimilarities, to distinguish them from other formulations of bisimilarity like the *early* and *open* ones (see [FMQ94]); these differ from the former because name instantiation is used in a different position in the bisimilarity clauses. An argument similar to that in Corollary 3.7 shows that in the mini π-calculus ground bisimilarity also coincides with early and open bisimilarities. In view of these results, in the remainder of the paper ground bisimulation will be simply called *bisimulation.*

3.2. Proof techniques

For the proof of one of the main results in the paper, namely Theorem 7.15, we shall use a proof technique for bisimulation in order to reduce the size of the relation to exhibit. In the bisimilarity clause, this technique allows us to manipulate the derivatives of two processes with the *expansion relation* and to cancel a common context. It extends a technique in [San94b] where contexts can only be monadic and static (i.e., they can only be of the form $\boldsymbol{\nu}\, \widetilde{b}\,(P \mid [\cdot])$). The expansion relation [AKH92, SM92], written $\lesssim$, is an asymmetric variant of $\approx$ which takes into account the number of τ-actions performed by processes. Thus, $P \lesssim Q$ holds if $P \approx Q$ but also Q has at least as many τ-moves as P. The expansion relation provides us with better ‘control’ on τ-moves of processes than the ordinary $\approx$.

Definition 3.8 (expansion) *A relation $\mathcal{R} \subseteq \mathcal{P}r \times \mathcal{P}r$ is an* expansion *if $P \,\mathcal{R}\, Q$ implies:*

1. *Whenever $P \xrightarrow{\mu} P'$, there exists Q' s.t. $Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \,\mathcal{R}\, Q'$;*

2. whenever $Q \xrightarrow{\mu} Q'$*, there exists* P' *s.t.* $P \xrightarrow{\widehat{\mu}} P'$ *and* $P' \mathcal{R}\, Q'$*.*

We say that Q expands P*, written* $P \lesssim Q$*, if* $P \mathcal{R}\, Q$*, for some expansion* $\mathcal{R}$ *.*

Relation $\lesssim$ is a preorder and enjoys the same congruence properties as $\approx$, including the closure w.r.t. substitutions.

Proposition 3.9 *The following is a chain of strict inclusions:* $\sim \; \subset \; \lesssim \; \subset \; \approx$ *.* ☐

Definition 3.10 (weak bisimulation up-to context and up-to $\gtrsim$) *A symmetric relation* $\mathcal{R}$ *is a* weak bisimulation up-to context and up-to $\gtrsim$ *if* $P \mathcal{R}\, Q$ *and* $P \xrightarrow{\mu} P''$ *imply that there are a context* C *and processes* $\widetilde{P'}$ *and* $\widetilde{Q'}$ *s.t.* $P'' \gtrsim C[\widetilde{P'}]$*,* $Q \overset{\widehat{\mu}}{\Longrightarrow} \gtrsim C[\widetilde{Q'}]$ *and* $\widetilde{P'} \mathcal{R}\, \widetilde{Q'}$*.*

Definition 3.11 *A relation* $\mathcal{R}$ *is* closed under substitutions *if* $(P, Q) \in \mathcal{R}$ *implies* $(P\sigma, Q\sigma) \in \mathcal{R}$ *, for all substitution* σ*.*

Lemma 3.12 *Suppose that* $\mathcal{R}$ *is is closed under substitutions and is a weak bisimulation up-to context and up-to* $\gtrsim$*. If* $(\widetilde{P}, \widetilde{Q}) \in \mathcal{R}$ *and* $C[\widetilde{P}] \xrightarrow{\mu} P'$*, then there are a context* C' *and processes* $\widetilde{P'}$ *and* $\widetilde{Q'}$ *s.t.* $P' \gtrsim C'[\widetilde{P'}]$*,* $C[Q] \overset{\widehat{\mu}}{\Longrightarrow} \gtrsim C'[\widetilde{Q'}]$ *and* $\widetilde{P'} \mathcal{R}\, \widetilde{Q'}$*.*

PROOF: Proceeding by induction on the structure of C. ☐

Theorem 3.13 *If* $\mathcal{R}$ *is is closed under substitutions and is a weak bisimulation up-to context and up-to* $\gtrsim$*, then* $\mathcal{R} \subseteq \approx$ *.*

PROOF: Use the previous lemma to show that

$$\mathcal{R}^{c} = \{(P, Q) \; : \; \begin{array}{l} \text{for some context } C \text{ and processes } \widetilde{P}, \widetilde{Q} \\ \text{it holds that } P \gtrsim C[\widetilde{P}], Q \gtrsim C[\widetilde{Q}] \text{ and } \widetilde{P} \mathcal{R}\, \widetilde{Q} \end{array} \}$$

is a weak bisimulation. ☐

3.3. Some laws for bisimilarity

For ease of reference, in this section we have collected some simple laws for bisimilarity which we shall apply several times in the paper. First, two laws for restriction and parallel composition:

L1: $\boldsymbol{\nu}\, a\, (P \mid Q) \sim P \mid \boldsymbol{\nu}\, a\, Q$, if $a \notin \mathrm{fn}(P)$;

L2: $\boldsymbol{\nu}\, a\, (a(\widetilde{e}).\, P \mid \overline{a}\langle\widetilde{b}\rangle.\, Q) \gtrsim \boldsymbol{\nu}\, a\, (P\{\widetilde{b}/\widetilde{e}\} \mid Q)$.

Next, we report some distributivity laws for private replications, i.e., systems of the form

$$\boldsymbol{\nu}\, y\, (P \mid \; !\, y(\widetilde{q}).\, Q)$$

in which y may occur free in P and Q only in output subject position. One should think of Q as a private resource of P, for P is the only process who can access Q; indeed P can activate as many copies of Q as needed.

Lemma 3.14 *Suppose a occurs free in $P, R, \alpha.P, Q$ only in output subject position. Then:*

1. $\boldsymbol{\nu}\, a\,(P \mid R \mid\, !\,a(\widetilde{b}).\,Q) \sim \boldsymbol{\nu}\, a\,(P \mid\, !\,a(\widetilde{b}).\,Q) \mid \boldsymbol{\nu}\, a\,(R \mid\, !\,a(\widetilde{b}).\,Q)$;
2. $\boldsymbol{\nu}\, a\,((\,!\,P) \mid\, !\,a(\widetilde{b}).\,Q) \sim\, !\,\boldsymbol{\nu}\, a\,(P \mid\, !\,a(\widetilde{b}).\,Q)$;
3. $\boldsymbol{\nu}\, a\,(\alpha.\,P \mid\, !\,a(\widetilde{b}).\,Q) \sim \alpha.\,\boldsymbol{\nu}\, a\,(P \mid\, !\,a(\widetilde{b}).\,Q)$, *if* $bn(\alpha) \cap fn(a(\widetilde{b}).\,Q) = \emptyset$;
4. $\boldsymbol{\nu}\, a\,((\boldsymbol{\nu}\, c\, P) \mid\, !\,a(\widetilde{b}).\,Q) \sim \boldsymbol{\nu}\, c\, \boldsymbol{\nu}\, a\,(P \mid\, !\,a(\widetilde{b}).\,Q)$ *if* $c \notin fn(a(\widetilde{b}).\,Q)$;
5. $\boldsymbol{\nu}\, a\,(P \mid\, !\,a(\widetilde{b}).\,Q) \sim P$, *if* $a \notin fn(P)$;
6. $\boldsymbol{\nu}\, a\,(\overline{a}\langle\widetilde{d}\rangle \mid\, !\,a(\widetilde{b}).\,Q) \gtrsim Q\{\widetilde{d}/\widetilde{b}\}$, *if* $a \notin fn(Q\{\widetilde{d}/\widetilde{b}\})$.

PROOF: Assertions (3-6) are easy. Assertions (1) and (2) have been first proved by Milner (see [Mil91]) by exhibiting appropriate strong bisimulations (Milner proved the result for $\sim_{\mathrm{L}}$, but the same proof works for $\sim$). $\square$

3.4. Abstractions In Milner's encoding of the lazy λ-calculus into the π-calculus, described in the next section, the encoding of a λ-term is parametric on a name, that is, is a function from names to π-calculus processes. We call such expressions *abstractions.* For the purposes of this paper unary abstractions, i.e., with only one parameter, suffice. An abstraction with parameter a and body P is written $(a)\,P$, and is a binder for a of the same nature as the input prefix binder $b(a).\,P$. If $F \stackrel{\mathrm{def}}{=} (a)\,P$, then $F\langle b\rangle$, called *application*, abbreviates $P\{b/a\}$ — the actual parameter b substitutes the formal parameter a in the body of F. Application has the same syntactic precedence as substitution (i.e., the highest), whereas abstraction has the lowest precedence; thus $(a)\,\alpha_1.\,(F_1\langle q\rangle\sigma \,|\, \alpha_2.\,F_2\langle r\rangle)$ means $(a)\,\big(\big(\alpha_1.\,((F\langle q\rangle)\sigma)\big) \mid \alpha_2.\,(F_2\langle r\rangle)\big)$.

Processes and abstractions form the class $\mathcal{P}r^*$ of *agents.* F and G range over abstractions; A over agents. To distinguish logically different agents, we assign them a sort: A process takes the sort (); and, if name a has sort s, then an abstraction $(a)\,P$ takes sort (s). We extend bisimulation to abstractions and set $(a)\,P \sim (a)\,Q$ (resp. $(a)\,P \approx (a)\,Q$) if $P \sim Q$ (resp. $P \approx Q$); note that, as for input prefixes, so for abstractions instantiations of the bound name are not not needed. The congruence of $\sim$ and $\approx$ is preserved by the abstraction and application constructs; the latter because both bisimulations are preserved by name instantiation.

4 Milner's encoding of the lazy lambda calculus

4.1. The lazy λ-calculus We let x and y range over the set of λ-calculus variables. The set Λ^{o} of λ-terms is defined by the grammar

$$M := x \;\Big|\; \lambda x.\,M \;\Big|\; M_1 M_2\,.$$

Free variables, closed terms, substitution, alpha-conversion etc. are defined as usual [Bar84, HS86]. We identify alpha-convertible terms, and write $M = N$ if M and N are alpha-convertible. The set of free variables in the term M is $\mathrm{fv}(M)$, and the subclass of Λ^{o} only

$$\begin{array}{rcl}\mathcal{E}[\![\lambda x.M]\!] & \overset{\text{def}}{=} & (p)\, p(x,q).\, \mathcal{E}[\![M]\!]\langle q\rangle \\ \mathcal{E}[\![x]\!] & \overset{\text{def}}{=} & (p)\, \overline{x}\langle p\rangle \\ \mathcal{E}[\![MN]\!] & \overset{\text{def}}{=} & (p)\, \boldsymbol{\nu}\, r,x\, \big(\mathcal{E}[\![M]\!]\langle r\rangle \mid \overline{r}\langle x,p\rangle \mid\, !\, x(q).\, \mathcal{E}[\![N]\!]\langle q\rangle\big),\ x \text{ fresh.}\end{array}$$

Table 2: The encoding of the lazy λ-calculus

containing the closed terms is Λ. We group brackets on the left; therefore MNL is $(MN)L$. 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. Symbol Ω is the always-divergent term $(\lambda x.xx)(\lambda x.xx)$.

In Abramsky's *lazy lambda calculus* [Abr89], the redex is always at the extreme left of a term. The reduction relation $\longrightarrow \subseteq \Lambda^{\rm o} \times \Lambda^{\rm o}$ (for our purposes we need it defined on open terms) is determined by the two rules:

$$\textsc{Beta}\text{:}\ (\lambda x.M)N \longrightarrow M\{^N\!/x\}\ , \qquad\qquad \textsc{App}\text{:}\ \frac{M \longrightarrow M'}{MN \longrightarrow M'N}\ .$$

The reflexive and transitive closure of $\longrightarrow$ is $\Longrightarrow$. We write $M \Downarrow$ if M is convergent, i.e., it can reduce to an abstraction. In the remainder of the paper, unless otherwise specified, M, N, L are from $\Lambda^{\rm o}$.

4.2. Milner's encoding

We informally explain Milner's encoding $\mathcal{E}$ of the lazy λ-calculus into the π-calculus. The core of any encoding of the λ-calculus into a process calculus is the translation of function application. This becomes a particular form of parallel combination of two agents, the function and its argument; beta-reduction is then modeled as process interaction. Since the syntax of the π-calculus only allows for the transmission of names along channels, the communication of a term is simulated by the communication of a *trigger* for it.

In the λ-calculus, λ is the only port; a λ-terms receives its argument at λ. In the π-calculus, there are infinitely-many ports, so the encoding of a λ-term M is parametric over a port p. This can be thought of as the *location* of M, for p represents the unique port along which M interacts with its environment. M receives two names along p: The first is a trigger for its argument and the second is the location to be used for the next interaction. The encoding is presented in Table 2. It is slightly different from Milner's original encoding [Mil90] in the rule for application: In Milner's encoding, the particle $\overline{r}\langle x,q\rangle$ guards the process $!\, x(q).\, \mathcal{E}[\![N]\!]\langle q\rangle$. We could transform the guard into a parallel composition, so to use only the operators of the mini π-calculus, because x is restricted and hence $\overline{r}\langle x,q\rangle$ is blocking for $!\, x(q).\, \mathcal{E}[\![N]\!]\langle q\rangle$.

Two sorts of names are used in the encoding: *Location names* like by p, q and r, and *trigger names* like by x, y and z. For simplicity, we have assumed that the set of trigger names is the same as the set of λ-variables. If $s_{\rm loc}$ denotes the sort of the location names, then the encoding of a λ-term is an abstraction of sort $(s_{\rm loc})$. In the remainder of the paper, all abstractions we write have sort $(s_{\rm loc})$, names $p, q, r, \ldots$ are location names, and $x, y, z, \ldots$ are trigger names.

5 Operational correspondence for the encoding

In this section, we carefully examine the operational correspondence between source and target terms of the encoding. This will also be the basis for the study of full abstraction in Section 7. First, we introduce a process notation which allows us to give a simpler description of encodings of λ-terms with a variable in head position. We recall that $\sim$ is strong bisimilarity and $\lesssim$ is the expansion relation.

Definition 5.1 *For $n > 0$, we define:*

$$\mathcal{O}_n\langle r_o, r_n, F_1, \ldots, F_n\rangle \overset{def}{=}$$
$$\boldsymbol{\nu}\, r_1, \ldots, r_{n-1}, x_1, \ldots, x_n \left(\overline{r}_o\langle x_1, r_1\rangle \mid \ldots \mid \overline{r}_{n-1}\langle x_n, r_n\rangle \mid\ !\, x_1(q).\, F_1\langle q\rangle \mid \ldots \mid\ !\, x_n(q_n).\, F_n\langle q_n\rangle\right)$$

where names $r_1, \ldots, r_{n-1}, x_1 \ldots, x_n$, q are fresh.

The i-th output of $\mathcal{O}_n\langle r_o, r_n, F_1, \ldots, F_n\rangle$, namely $\overline{r}_i\langle x_i, r_{i+1}\rangle$, liberates the agent $x_i(q).\, F_i\langle q\rangle$ and the successive output at r_{i+1}.

Lemma 5.2

1. *If $n > 1$ and $\mathcal{O}_n\langle r_o, r_n, F_1, \ldots, F_n\rangle \xrightarrow{\mu} P$, then $\mu = (\boldsymbol{\nu}\, x_1, r_1)\overline{r}_o\langle x_1, r_1\rangle$ and $P \sim \mathcal{O}_{n-1}\langle r_1, r_n, F_2, \ldots, F_n\rangle \mid\ !\, x_1(q).\, F_1\langle q\rangle$.*
2. *If $n > 0$, then $\mathcal{E}[\![xM_1 \ldots M_n]\!]\langle r_n\rangle \sim \boldsymbol{\nu}\, r_o\, (\overline{x}\langle r_o\rangle \mid \mathcal{O}_n\langle r_o, r_n, \mathcal{E}[\![M_1]\!], \ldots, \mathcal{E}[\![M_n]\!]\rangle)$.*

PROOF: Proceed by induction on n. □

Lemma 5.3

1. *If $M \longrightarrow N$, then $\mathcal{E}[\![M]\!]\langle p\rangle \longrightarrow\gtrsim \mathcal{E}[\![N]\!]\langle p\rangle$.*
2. *If $M = \lambda x.\, N$, then $\mathcal{E}[\![M]\!]\langle p\rangle \xrightarrow{p(x,q)} \mathcal{E}[\![N]\!]\langle q\rangle$.*
3. *If $M = x$ then $\mathcal{E}[\![M]\!]\langle p\rangle \xrightarrow{\overline{x}\langle p\rangle} \mathbf{0}$.*
4. *If $M = xM_1 \ldots M_n$, $n > 0$, then $\mathcal{E}[\![M]\!]\langle p\rangle \xrightarrow{(\boldsymbol{\nu}\, q)\, \overline{x}\langle q\rangle} \sim \mathcal{O}_n\langle q, p, \mathcal{E}[\![M_1]\!], \ldots, \mathcal{E}[\![M_n]\!]\rangle$.*

PROOF: Assertions (2) and (3) are immediate from the definition of the encoding. Assertion (4) follows from Lemma 5.2(2). Assertion (1) is proved by induction on the structure of M. The most interesting case is when $M = (\lambda x.\, M_1)M_2$ and $N = M_1\{^{M_2}/x\}$. We have

$$\mathcal{E}[\![(\lambda x.M)N]\!]\langle p\rangle \longrightarrow\sim \boldsymbol{\nu}\, x\, (\mathcal{E}[\![M]\!]\langle p\rangle \mid\ !\, x(r).\, \mathcal{E}[\![N]\!]\langle r\rangle)\, .$$

Then

$$\boldsymbol{\nu}\, x\, (\mathcal{E}[\![M]\!]\langle p\rangle \mid\ !\, x(r).\, \mathcal{E}[\![N]\!]\langle r\rangle) \gtrsim \mathcal{E}[\![M\{^{N}/x\}]\!]\langle p\rangle \tag{3}$$

can be proved proceeding by induction on the structure of M, and using the distributivity properties of private replications in Lemma 3.14; alternatively, (3) can be inferred as an instance of Lemma 6.3. □

Proposition 5.4 (operational correspondence on the reductions of M)

1. *If* $M \Longrightarrow N$, *then* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \Longrightarrow\gtrsim \mathcal{E}\llbracket N\rrbracket\langle p\rangle$.
2. *If* $M \Longrightarrow \lambda x.N$, *then* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \overset{p(x,q)}{\Longrightarrow} \gtrsim \mathcal{E}\llbracket N\rrbracket\langle q\rangle$.
3. *If* $M \Longrightarrow x$ *then* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \overset{\overline{x}\langle q\rangle}{\Longrightarrow} \gtrsim \mathbf{0}$.
4. *If* $M \Longrightarrow xM_1\dots M_n$, $n > 0$, *then* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\,\overline{x}\langle q\rangle}{\Longrightarrow} \gtrsim \mathcal{O}_n\langle q, p, \mathcal{E}\llbracket M\rrbracket_1, \dots, \mathcal{E}\llbracket M_n\rrbracket\rangle$.

PROOF: Assertion (1) is proved using induction on the number of steps made by M and Lemma 5.3(1). The other assertions are consequences of (1) and Lemma 5.3(2-4). ☐

Proposition 5.5 (operational correspondence on the weak transitions of $\mathcal{E}\llbracket M\rrbracket\langle p\rangle$)

1. *If* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \Longrightarrow P$, *then there is* N *s.t.* $M \Longrightarrow N$ *and* $P \gtrsim \mathcal{E}\llbracket N\rrbracket\langle p\rangle$.
2. *If* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \Longrightarrow \xrightarrow{\mu} P$ *and* μ *is an input action, then* $\mu = p(x,q)$ *and there is* N *s.t.* $M \Longrightarrow \lambda x.N$, $P \gtrsim \mathcal{E}\llbracket N\rrbracket\langle q\rangle$.
3. *If* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \overset{\mu}{\Longrightarrow} P$ *and* μ *is a free output, then there is* x *s.t.* $\mu = \overline{x}\langle p\rangle$ *and* $P \gtrsim \mathbf{0}$.
4. *If* $\mathcal{E}\llbracket M\rrbracket\langle p\rangle \overset{\mu}{\Longrightarrow} P$ *and* μ *is not a free output, then there are* x *and* $M_1, \dots, M_n$, $n > 0$, *s.t.* $\mu = (\boldsymbol{\nu}\, q)\,\overline{x}\langle q\rangle$, $M \Longrightarrow xM_1\dots M_n$, *and* $P \gtrsim \mathcal{O}_n\langle q, p, \mathcal{E}\llbracket M\rrbracket_1, \dots, \mathcal{E}\llbracket M_n\rrbracket\rangle$.

PROOF: By induction on the length of the transition of $\mathcal{E}\llbracket M\rrbracket\langle p\rangle$. For the basic case, note that for each M and p, process $\mathcal{E}\llbracket M\rrbracket\langle p\rangle$ has only one possible transition, and hence the thesis follows from Lemma 5.3. ☐

6 A λ-model from the process terms

A *partial function* φ from a set D_1 to a set D_2 can be undefined on some elements of D_1; the subset of D_1 on which φ is defined is $\mathrm{dom}(\varphi)$. If $\mathrm{dom}(\varphi)$ is finite, then φ is *finite.* We write $[d_2/d_1]\varphi$ for the function which maps d_1 to d_2 and which behaves like φ on the remaining elements of D_1. To ease readability, we sometimes abbreviate $\varphi(d)$ as φ_d.

There are simple syntax-free definitions of λ-model. However, since we already have the mapping from λ to process terms, it is more convenient to use a definition where we can use such a mapping explicitly. A *finite valuation* is a finite function from the set of λ-variables to the domain D of the λ-model. Below, *finite* evaluations are enough because the set of free variables of a λ-term is finite.

Definition 6.1 (λ-model) *A* λ-model *is a triple* $< D, \cdot, \mathcal{M} >$, *where* D *is a set with at least two elements, '$\cdot$' is a mapping from* $D \times D$ *to* D *and* $\mathcal{M}$ *is a mapping which assigns, to* *each* λ*-term* M *and finite valuation* ρ *with fv(M)* $\subseteq$ *dom*(ρ)*, a member* $\mathcal{M}[\![M]\!]_\rho \in D$ *such that:*

1. $\mathcal{M}[\![x]\!]_\rho = \rho(x)$
2. $\mathcal{M}[\![MN]\!]_\rho = \mathcal{M}[\![M]\!]_\rho \cdot \mathcal{M}[\![N]\!]_\rho$
3. $\mathcal{M}[\![\lambda x.M]\!]_\rho \cdot \boldsymbol{d} = \mathcal{M}[\![M]\!]_{[\boldsymbol{d}/x]\rho}$ *for all* $\boldsymbol{d} \in D$
4. $\mathcal{M}[\![M]\!]_\rho = \mathcal{M}[\![M]\!]_\sigma$ *if* $\rho(x) = \sigma(x)$ *for all* x *free in* M
5. $\mathcal{M}[\![\lambda x.M]\!]_\rho = \mathcal{M}[\![\lambda y.M\{y/x\}]\!]_\rho$, y *not free in* M
6. *if* $\mathcal{M}[\![M]\!]_{[\boldsymbol{d}/x]\rho} = \mathcal{M}[\![N]\!]_{[\boldsymbol{d}/x]\rho}$ *for all* $\boldsymbol{d} \in D$, *then* $\mathcal{M}[\![\lambda x.M]\!]_\rho = \mathcal{M}[\![\lambda x.N]\!]_\rho$.

We wish to construct a λ-model using π-calculus agents and the encoding $\mathcal{E}$ of the λ-calculus into the π-calculus. It is reasonable that the model should respect observation equivalence ($\approx$), which is the semantic equality adopted in the π-calculus. So, for a π-calculus agent A, let $[\,A\,]_\approx$ be the equivalence class of A, namely

$$[\,A\,]_\approx \stackrel{\text{def}}{=} \{A' \;:\; A' \in \mathcal{P}r^* \text{ and } A \approx A'\}\,.$$

The elements of the domain D of the model will be the equivalence classes of the π-calculus agents with the same sort (s_{loc}) as the agents encoding λ-terms:

$$D \stackrel{\text{def}}{=} \{[\,F\,]_\approx \;:\; F \in \mathcal{P}r^* \text{ and } F \text{ has sort } (s_{\text{loc}})\}\,. \tag{4}$$

The definition of application on these elements follows the translation of λ-application in $\mathcal{E}$:

$$[\,G\,]_\approx \cdot [\,F\,]_\approx \stackrel{\text{def}}{=} [\,(p)\boldsymbol{\nu}\, r,x\,\big(G\langle r\rangle \mid \overline{r}\langle x,p\rangle \mid\, !\,x(q).\,F\langle q\rangle\big)\,]_\approx \qquad \text{for} \quad p,x,r,q \text{ not free in } F,G. \tag{5}$$

The definition of application is consistent since, by the congruence properties of $\approx$, the result of the application does not depend upon the representatives G and F chosen from the equivalence classes. We are left with the definition of $\mathcal{M}[\![M]\!]_\rho$. The valuation ρ maps a λ-variable x to $\rho(x)$, which is a set of bisimilar π-calculus agents. Given a valuation ρ, we denote by $\hat{\rho}$ the "conversion" of ρ which operates on a π-calculus name x and selects a representative out of the equivalence class of $\rho(x)$; that is, $\hat{\rho}(x) \in \rho(x)$ if $x \in \text{dom}(\rho)$, and $\hat{\rho}(x)$ is undefined if $x \notin \text{dom}(\rho)$. Now, the mapping $\mathcal{M}$ of the λ-model is defined in terms of $\mathcal{E}$ as follows:

$$\mathcal{M}[\![M]\!]_\rho \stackrel{\text{def}}{=} [\;(p)\,\boldsymbol{\nu}\,\widetilde{y}\,\big(\mathcal{E}[\![M]\!]\langle p\rangle\{\widetilde{y}/\widetilde{x}\} \mid \prod_i !\,y_i(q).\,\hat{\rho}_{x_i}\langle q\rangle\big)\,]_\approx \tag{6}$$

where $\widetilde{y}$ and q are fresh names, $\widetilde{x} = \text{dom}(\rho)$, and x_i (resp. y_i) is the i-th component of $\widetilde{x}$ (resp. $\widetilde{y}$). The use of fresh names $\widetilde{y}$ in the outermost restriction — and hence the substitution $\{\widetilde{y}/\widetilde{x}\}$ — is needed to avoid that free names of $\hat{\rho}_{x_i}$ become bound, for names $\widetilde{x}$ might occur free in $\hat{\rho}_{x_i}$. Definition (6) is independent of the representatives of the equivalence classes selected by $\hat{\rho}$, since $\approx$ is a congruence.

Agent $\hat{\rho}_{x_i}$ is used in (6) as a private resource for $\mathcal{E}[\![M]\!]\langle p\rangle\{\widetilde{y}/\widetilde{x}\}$, accessible through name y_i. Behaviourally, this amounts to replacing the translation of the λ-calculus variable x_i with the agent $\hat{\rho}_{x_i}$. This idea is formalised in Lemma 6.3 using the encoding $\widehat{\mathcal{E}}$ below.

Definition 6.2 *Let* φ *be a finite function from trigger names to abstractions of sort* (s_{loc})*. The mapping* $\widehat{\mathcal{E}}[\![-]\!]_\varphi$*, from* λ*-terms to* π*-calculus abstractions of sort* (s_{loc})*, is defined induc-*

tively thus:

$$\begin{array}{rcl}
\widehat{\mathcal{E}}\llbracket \lambda x.M\rrbracket_{\varphi} & \stackrel{def}{=} & (p)\, p(x,q).\, \widehat{\mathcal{E}}\llbracket M\rrbracket_{\varphi}\,\langle q\rangle \\
\widehat{\mathcal{E}}\llbracket x\rrbracket_{\varphi} & \stackrel{def}{=} & \begin{cases} \varphi_x & \textit{if } x \in \textit{dom}(\varphi) \\ (p)\, \overline{x}\langle p\rangle & \textit{if } x \notin \textit{dom}(\varphi) \end{cases} \\
\widehat{\mathcal{E}}\llbracket MN\rrbracket_{\varphi} & \stackrel{def}{=} & (p)\, \boldsymbol{\nu}\, r, x\left(\widehat{\mathcal{E}}\llbracket M\rrbracket_{\varphi}\,\langle r\rangle \mid \overline{r}\langle x,p\rangle \mid !\, x(q).\, \widehat{\mathcal{E}}\llbracket N\rrbracket_{\varphi}\,\langle q\rangle\right)
\end{array}$$

where p, q, r and x are fresh.

Note that, since x is fresh w.r.t. φ, in $p(x,q).\, \widehat{\mathcal{E}}\llbracket M\rrbracket_{\varphi}\,\langle q\rangle$ we have $x \neq a$ and $x \notin \mathrm{fn}(\varphi_a)$, for all $a \in \mathrm{dom}(\varphi)$.

Lemma 6.3 *If ρ is a finite evaluation with $\widetilde{x} = dom(\rho)$, and $\widetilde{y}, q$ are fresh, then*

$$\boldsymbol{\nu}\, \widetilde{y}\, (\mathcal{E}\llbracket M\rrbracket\langle p\rangle\{\widetilde{y}/\widetilde{x}\} \mid \prod_i !\, y_i(q).\, \hat{\rho}_{x_i}\langle q\rangle) \gtrsim \widehat{\mathcal{E}}\llbracket M\rrbracket_{\hat{\rho}}$$

PROOF: By induction on the structure of M. We consider two cases and write $P_{\tilde{y}}$ for $\prod_i !\, y_i(q).\, \hat{\rho}_{x_i}\langle q\rangle$ as $P_{\tilde{y}}$.

(1) $M = y$ and $y \in \mathrm{dom}(\rho)$. We have:

$$\begin{array}{rcl}
\boldsymbol{\nu}\, \widetilde{y}\, (\mathcal{E}\llbracket M\rrbracket\langle p\rangle\{\widetilde{y}/\widetilde{x}\} \mid P_{\tilde{y}}) & = & (\text{definition of } \mathcal{E}) \\
\boldsymbol{\nu}\, \widetilde{y}\, (\overline{y}\langle p\rangle \mid P_{\tilde{y}}) & \sim & (\text{Lemma 3.14(5)}) \\
\boldsymbol{\nu}\, y\, (\overline{y}\langle p\rangle \mid !\, y(q).\, \hat{\rho}_x\langle q\rangle) & \gtrsim & (\text{Lemma 3.14(6)}) \\
\hat{\rho}_x\langle p\rangle & = & \widehat{\mathcal{E}}\llbracket M\rrbracket_{\hat{\rho}}\,\langle p\rangle
\end{array}$$

(2) $M = NL$. We have:

$$\begin{array}{rcl}
\boldsymbol{\nu}\, \widetilde{y}\, (\mathcal{E}\llbracket M\rrbracket\langle p\rangle\{\widetilde{y}/\widetilde{x}\} \mid P_{\tilde{y}}) & = & (\text{definition of } \mathcal{E}) \\
\boldsymbol{\nu}\, \widetilde{y}\left(\boldsymbol{\nu}\, r, z\left(\mathcal{E}\llbracket N\rrbracket\langle r\rangle\{\widetilde{y}/\widetilde{x}\} \mid \overline{r}\langle z,p\rangle \mid !\, z(q).\, (\mathcal{E}\llbracket L\rrbracket\langle q\rangle\{\widetilde{y}/\widetilde{x}\})\right) \mid P_{\tilde{y}}\right) & \sim & (\text{law } \mathbf{L1}) \\
\boldsymbol{\nu}\, r, z, \widetilde{y}\left(\mathcal{E}\llbracket N\rrbracket\langle r\rangle\{\widetilde{y}/\widetilde{x}\} \mid \overline{r}\langle z,p\rangle \mid !\, z(q).\, (\mathcal{E}\llbracket L\rrbracket\langle q\rangle\{\widetilde{y}/\widetilde{x}\}) \mid P_{\tilde{y}}\right) & \sim & (*) \\
\boldsymbol{\nu}\, r, z\left(\boldsymbol{\nu}\, \widetilde{y}\left(\mathcal{E}\llbracket N\rrbracket\langle r\rangle\{\widetilde{y}/\widetilde{x}\} \mid P_{\tilde{y}}\right) \mid \overline{r}\langle z,p\rangle \mid !\, z(q).\, \boldsymbol{\nu}\, \widetilde{y}\left(\mathcal{E}\llbracket L\rrbracket\langle q\rangle\{\widetilde{y}/\widetilde{x}\} \mid P_{\tilde{y}}\right)\right) & \gtrsim & (\text{induction}) \\
\boldsymbol{\nu}\, r, z\left(\widehat{\mathcal{E}}\llbracket M\rrbracket_{\hat{\rho}}\,\langle r\rangle \mid \overline{r}\langle z,p\rangle \mid !\, z(q).\, \widehat{\mathcal{E}}\llbracket L\rrbracket_{\hat{\rho}}\,\langle q\rangle\right) & = & \widehat{\mathcal{E}}\llbracket NL\rrbracket_{\hat{\rho}}\,\langle p\rangle
\end{array}$$

where $(*)$ is derived from the distributivity laws for private replications in Lemma 3.14. □

Corollary 6.4 shows the relationship between mapping $\mathcal{M}$ and encoding $\widehat{\mathcal{E}}$.

Corollary 6.4 $\mathcal{M}\llbracket M\rrbracket_{\rho} = [\, \widehat{\mathcal{E}}\llbracket M\rrbracket_{\hat{\rho}}\,]_{\approx}$

PROOF: From the definitions of $\widehat{\mathcal{E}}$ and $\mathcal{M}$, and Lemma 6.3. □

We can now define the λ-model.

Definition 6.5 (model $\mathcal{D}$) *We set $\mathcal{D} \stackrel{def}{=} < D, \cdot, \mathcal{M} >$, for $D, \cdot,$ and $\mathcal{M}$ as given in (4), (5) and (6).*

Lemma 6.6 *If* $\mathcal{M}[\![M]\!]_{[\,\widetilde{\boldsymbol{d}}\,/\,\widetilde{x}\,]\rho} = \mathcal{M}[\![N]\!]_{[\,\widetilde{\boldsymbol{d}}\,/\,\widetilde{x}\,]\rho}$ *for all* $\widetilde{\mathbf{d}}$, *then* $\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}} \approx \widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}$.

Proof: Take $F_x \overset{\text{def}}{=} (p)\,\overline{x}\langle p\rangle$ and $\boldsymbol{d}_x \overset{\text{def}}{=} [\,F_x\,]_{\approx}$. By Corollary 6.4, $\mathcal{M}[\![M]\!]_{[\,\widetilde{\boldsymbol{d}_x}\,/\,\widetilde{x}\,]\rho} = \mathcal{M}[\![N]\!]_{[\,\widetilde{\boldsymbol{d}_x}\,/\,\widetilde{x}\,]\rho}$ implies

$$\widehat{\mathcal{E}}[\![M]\!]_{[\,\widetilde{F_x}\,/\,\widetilde{x}\,]\hat{\rho}} \approx \widehat{\mathcal{E}}[\![N]\!]_{[\,\widetilde{F_x}\,/\,\widetilde{x}\,]\hat{\rho}}\,. \tag{7}$$

Moreover, by definitions of $\widehat{\mathcal{E}}$ and $\widetilde{F_x}$,

$$\text{for all } L \quad \widehat{\mathcal{E}}[\![L]\!]_{[\,\widetilde{F_x}\,/\,\widetilde{x}\,]\hat{\rho}} = \widehat{\mathcal{E}}[\![L]\!]_{\hat{\rho}}\,. \tag{8}$$

Now, (7) and (8) prove that $\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}} \approx \widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}$. $\square$

Corollary 6.7 $\mathcal{E}[\![M]\!] \approx \mathcal{E}[\![N]\!]$ *iff, for all* ρ, $\mathcal{M}[\![M]\!]_\rho = \mathcal{M}[\![N]\!]_\rho$.

Proof: The implication from left to right follows from the definition of $\mathcal{M}$ and congruence of $\approx$. Now, the implication from right to left. By Lemma 6.6, $\mathcal{M}[\![M]\!]_\rho = \mathcal{M}[\![N]\!]_\rho$ for all ρ implies $\widehat{\mathcal{E}}[\![M]\!]_\emptyset \approx \widehat{\mathcal{E}}[\![N]\!]_\emptyset$. This proves the result since, for all L, $\widehat{\mathcal{E}}[\![L]\!]_\emptyset = \mathcal{E}[\![L]\!]$. $\square$

Theorem 6.8 $\mathcal{D}$ *is a* λ*-model.*

Proof: We look at the main clauses of Definition 6.1.

(Clause 2) We have:

$$\begin{array}{rcl}
\mathcal{M}[\![MN]\!]_\rho & = & \text{(Corollary 6.4)} \\
[\,\widehat{\mathcal{E}}[\![MN]\!]_{\hat{\rho}}\,]_{\approx} & = & \text{(definition of } \widehat{\mathcal{E}}\text{)} \\
[\,(p)\boldsymbol{\nu}\,r,x\left(\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}}\,\langle r\rangle \mid \overline{r}\langle x,p\rangle \mid\, !\,x(q).\,\widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}\,\langle q\rangle\right)]_{\approx} & = & \text{(definition of application} \\
 & & \text{in the model)} \\
[\,\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}}\,]_{\approx}\cdot[\,\widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}\,]_{\approx} & = & \text{(Corollary 6.4)} \\
\mathcal{M}[\![M]\!]_\rho\cdot\mathcal{M}[\![N]\!]_\rho & &
\end{array}$$

(Clause 3) Let $\boldsymbol{d} = [\,F\,]_{\approx}$. By Corollary 6.4,

$$\begin{array}{rrcl}
 & \mathcal{M}[\![\lambda x.M]\!]_\rho\cdot\boldsymbol{d} & = & [\,(p)\boldsymbol{\nu}\,r,z\left(\widehat{\mathcal{E}}[\![\lambda x.\,M]\!]_{\hat{\rho}}\,\langle r\rangle \mid \overline{r}\langle z,p\rangle \mid\, !\,z(q).\,F\langle q\rangle\right)]_{\approx}\,, \\
\text{and} & \mathcal{M}[\![M]\!]_{[\boldsymbol{d}/x]\rho} & = & [\,\widehat{\mathcal{E}}[\![M]\!]_{[F/x]\hat{\rho}}\,]_{\approx}\,.
\end{array}$$

We have

$$\begin{array}{rcl}
\boldsymbol{\nu}\,r,z\left(\widehat{\mathcal{E}}[\![\lambda x.\,M]\!]_{\hat{\rho}}\,\langle r\rangle \mid\ \mid \overline{r}\langle z,p\rangle \mid\, !\,z(q).\,F\langle q\rangle\right) & = & \\
\boldsymbol{\nu}\,r,z\left(r(x,q).\,\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}}\,\langle q\rangle \mid \overline{r}\langle z,p\rangle \mid\, !\,z(q).\,F\langle q\rangle\right) & \approx & \text{(laws } \mathbf{L1},\mathbf{L2}\text{)} \\
\boldsymbol{\nu}\,z\left(\widehat{\mathcal{E}}[\![M]\!]_\rho\,\langle p\rangle\{z/x\} \mid\, !\,z(q).\,F\langle q\rangle\right) & \approx & (*) \\
\widehat{\mathcal{E}}[\![M]\!]_{[F/x]\hat{\rho}} & &
\end{array}$$

where $(*)$ is obtained using Lemma 6.3 to expand the definition of $\widehat{\mathcal{E}}$.

(Clause 6) By Lemma 6.6, if $\mathcal{M}[\![M]\!]_{[\boldsymbol{d}/x]\rho} = \mathcal{M}[\![N]\!]_{[\boldsymbol{d}/x]\rho}$ for all $\boldsymbol{d}$, then $\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}} \approx \widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}$. Since $\approx$ is a congruence, we get

$$(p)\,p(x,q).\,\widehat{\mathcal{E}}[\![M]\!]_{\hat{\rho}}\,\langle q\rangle \approx (p)\,p(x,q).\,\widehat{\mathcal{E}}[\![N]\!]_{\hat{\rho}}\,\langle q\rangle\,.$$

By Corollary 6.4 and the definition of $\widehat{\mathcal{E}}$, this means $\mathcal{M}[\![\lambda x.M]\!]_\rho = \mathcal{M}[\![\lambda x.N]\!]_\rho$. $\square$

We could have tried to be more selective in the definition of model $\mathcal{D}$, and take $D^* = \{[\,\mathcal{E}[\![M]\!]\,]_{\approx} \ : \ M \in \Lambda\}$ as its domain; then $\mathcal{D}^* =< D^*, \cdot, \mathcal{M} >$ represents the *interior* of $\mathcal{D}$ [HS86]. But it turns out that $\mathcal{D}^*$ is *not* a λ-model. Clause (6) in Definition 6.1 fails. As counterexample, take the terms $\lambda x.\, xx$ and $\lambda x.\,(x\lambda y.\,(xy))$. Their encodings into π-calculus are not behaviourally equivalent (see Lemma 7.5). Hence $\mathcal{M}[\![\lambda x.\, xx]\!]_{\emptyset} \neq \mathcal{M}[\![\lambda x.\,(x\lambda y.\,(xy))]\!]_{\emptyset}$ However, for all closed N, if $\boldsymbol{d} = [\,\mathcal{E}[\![N]\!]\,]_{\approx}$, we have

$$\mathcal{M}[\![xx]\!]_{[\boldsymbol{d}/x]} = [\,\mathcal{E}[\![NN]\!]\,]_{\approx} = [\,\mathcal{E}[\![N(\lambda y.\,(Ny))]\!]\,]_{\approx} = \mathcal{M}[\![x(\lambda y.\,(xy))]\!]_{[\boldsymbol{d}/x]}.$$

Therefore $\mathcal{D}$ is an example of a λ-model whose interior is not a λ-model; see [HL80] for more examples.

Since $\mathcal{D}$ is a λ-model, we can infer all properties of λ-models for it and, hence, the two corollaries below (for the proof of the first, one also needs Corollary 6.7). We write $\lambda\beta \vdash M = N$ if $M = N$ is an equation in the formal theory given by the alpha and beta axioms plus the rules of inference for equivalence and congruence.

Corollary 6.9 (validity of beta equality for $\mathcal{E}$) *If* $\lambda\beta \vdash M = N$*, then we have* $\mathcal{E}[\![M]\!] \approx \mathcal{E}[\![N]\!]$. ☐

Corollary 6.10 $< D, \cdot >$ *is a combinatory algebra where the two distinguished elements* $\boldsymbol{k}$ *and* $\boldsymbol{s}$ *can be defined as* $\boldsymbol{k} = [\mathcal{E}[\![\lambda xy.x]\!]\,]_{\approx}$*, and* $\boldsymbol{s} = [\mathcal{E}[\![\lambda xyz.(xz(yz))]\!]\,]_{\approx}$. ☐

However model $\mathcal{D}$ is *not* extensional, i.e. it is not a $\lambda\eta$ model. As counterexample, take Ω and $\lambda x.(\Omega x)$. Then $\mathcal{E}[\![\Omega]\!]\langle p\rangle \not\approx \mathcal{E}[\![\lambda x.(\Omega x)]\!]\langle p\rangle$, since $\mathcal{E}[\![\Omega]\!]\langle p\rangle \approx \mathbf{0}$, whereas $\mathcal{E}[\![\lambda x.(\Omega x)]\!]\langle p\rangle$ can perform a visible action at p. This failure is not too surprising, since our encoding mimics the lazy λ-calculus, in which the η rule is not valid. However, as in the lazy λ-calculus, the η rule holds if M is convergent:

Theorem 6.11 (conditional extensionality) *If* $x \notin fv(M)$*, then* $M \Downarrow$ *implies* $\mathcal{E}[\![\lambda x.\,(Mx)]\!] \approx \mathcal{E}[\![M]\!]$.

Proof: If $M \Downarrow$, then $M \Longrightarrow \lambda z.\, N$, for some z and N. Therefore also $Mx \Longrightarrow N\{^x/z\}$. By validity of beta equality for $\mathcal{E}$ (Corollary 6.9)

$$\mathcal{E}[\![M]\!] \approx \mathcal{E}[\![\lambda z.\, N]\!] \tag{9}$$

and $\mathcal{E}[\![Mx]\!] \approx \mathcal{E}[\![N\{^x/z\}]\!]$. Applying the latter equality in $\mathcal{E}[\![\lambda x.\,(Mx)]\!]$ one gets

$$\mathcal{E}[\![\lambda x.\,(Mx)]\!] \approx \mathcal{E}[\![\lambda x.\, N\{^x/z\}]\!] \approx \mathcal{E}[\![\lambda z.\, N]\!]\,. \tag{10}$$

Equalities (9) and (10) prove the theorem. ☐

7 Full abstraction

Full abstraction, first studied by Milner [Mil77] and Plotkin [Plo77], is the problem of finding a denotational interpretation for a programming language such that the resulting semantic equality coincides with a notion of operational indistinguishability.

Inspired by the work of Milner and Park in concurrency [Par81, Mil89], Abramsky [Abr89] introduced an operational equivalence on closed lazy λ-terms called *applicative bisimulation.*

Definition 7.1 *A symmetric relation* $\mathcal{R} \subseteq \Lambda \times \Lambda$ *is an* applicative bisimulation *if* $M \mathcal{R} N$ *and* $M \Longrightarrow \lambda x.\, M'$ *imply that there is an* N' *such that* $N \Longrightarrow \lambda x.\, N'$ *and* $M'\{L/x\} \mathcal{R} N'\{L/x\}$*, for all* $L \in \Lambda$*. Two terms* $M, N \in \Lambda \times \Lambda$ *are* applicative bisimilar*, written* $M \simeq N$*, if* $M \mathcal{R} N$ *holds, for some applicative bisimulation* $\mathcal{R}$ *.*

Applicative bisimulation can then be extended to open terms: If $M, N \in \Lambda^o$ *with* $fv(M,N) \subseteq \{\widetilde{x}\}$*, then* $M \simeq N$ *if for all* $\widetilde{L} \subseteq \Lambda$*, we have* $M\{\widetilde{L}/\widetilde{x}\} \simeq N\{\widetilde{L}/\widetilde{x}\}$*.*

Applicative bisimulation has been extensively studied by Abramsky and Ong [AO93]; in particular, they have showed that it is a congruence relation.

The classical setting in which the full abstraction problem has been developed is the simply typed λ-calculus. With the introduction of the operational equivalence resulting from applicative bisimulation, it can be neatly transferred to the untyped λ-calculus and it has motivated elegant works by Abramsky, Ong and Boudol ([AO93, Bou94]). A denotational interpretation is said to be *sound* if it only equates operationally equivalent terms, *complete* if it equates all operationally equivalent terms, and *fully abstract* if it is sound and complete.

We call the equality on λ-terms induced by model $\mathcal{D}$ of the previous section λ-*observation equivalence.*

Definition 7.2 (λ-observation equivalence) *For* $M, N \in \Lambda^o$*, we say that* M *and* N *are* λ-observationally equivalent*, written* $M \approx_\lambda N$*, if for all valuation* ρ *with* $dom(\rho) = fv(M,N)$*, it holds that* $\mathcal{M}[\![M]\!]_\rho = \mathcal{M}[\![N]\!]_\rho$*.*

By Corollary 6.7, λ-observation equivalence coincides with the equivalence induced, via encoding $\mathcal{E}$, by π-calculus observation equivalence:

Proposition 7.3 *For all* $M, N \in \Lambda^o$*, it holds that* $M \approx_\lambda N$ *iff* $\mathcal{E}[\![M]\!] \approx \mathcal{E}[\![N]\!]$*.* $\square$

Model $\mathcal{D}$ of Definition 6.5 is sound but not complete w.r.t. applicative bisimulation. To show this, we prove that bisimilarity between the encoding of two λ-terms implies (applicative) bisimilarity between the two original λ-terms, whereas the opposite implication fails; similar results have been proved by Milner in [Mil90].

Proposition 7.4 (soundness of $\mathcal{D}$ w.r.t. $\simeq$) $M \approx_\lambda N$ *implies* $M \simeq N$*.*

Proof: It suffices to show that for all $M, N \in \Lambda$, $\mathcal{E}[\![M]\!] \approx \mathcal{E}[\![N]\!]$ implies $M \simeq N$.

Abramsky and Ong have proved that applicative bisimulation can be described in terms of the convergence predicate $M \Downarrow$: For terms $M, N \in \Lambda$, the property

"in all closed λ-context C, $C[M] \Downarrow$ iff $C[N] \Downarrow$"

holds if and only if $M \simeq N$.

Let us first extend the convergence predicate to π-calculus processes: We set $P \Downarrow$ if P can perform a visible action, i.e., there is $\mu \neq \tau$ and P' s.t. $P \overset{\mu}{\Longrightarrow} P'$. Propositions 5.4 and 5.5 show that for all closed λ-terms and names p,

$$M \Downarrow \qquad \text{iff} \qquad \mathcal{E}[\![M]\!]\langle p\rangle \Downarrow . \tag{11}$$

Now, $\mathcal{E}[\![M]\!]\langle p\rangle \approx \mathcal{E}[\![N]\!]\langle p\rangle$ implies that $\mathcal{E}[\![M]\!]\langle p\rangle \Downarrow$ iff $\mathcal{E}[\![N]\!]\langle p\rangle \Downarrow$. Therefore, since $\mathcal{E}$ is compositional and $\approx$ is a congruence, $\mathcal{E}[\![M]\!]\langle p\rangle \approx \mathcal{E}[\![N]\!]\langle p\rangle$ also implies that for all closed λ-calculus contexts C, $\mathcal{E}[\![C[M]]\!]\langle p\rangle \Downarrow$ iff $\mathcal{E}[\![C[N]]\!]\langle p\rangle \Downarrow$. This and (11) give $M \simeq N$. ☐

Proposition 7.5 (non-completeness of $\mathcal{D}$ w.r.t. $\simeq$) *There are $M_1, M_2 \in \Lambda$ s.t. $M_1 \simeq M_2$ but $M_1 \not\approx_\lambda M_2$.*

Proof: Take $M_1 \stackrel{\text{def}}{=} \lambda x.\, xx$ and $M_2 \stackrel{\text{def}}{=} \lambda x.\, (x\lambda y.\, (xy))$. We have $M_1 \approx_\lambda M_2$ if for all $N \in \Lambda$, $NN \simeq N\lambda y.\, (Ny)$. If N is convergent, then, by conditional extensionality, $N \simeq \lambda y.\, (Ny)$ and hence, since $\simeq$ is a congruence relation, $NN \simeq N\lambda y.\, (Ny)$. If N is not convergent, then $NN \simeq \Omega \simeq N\lambda y.\, (Ny)$.

On the other hand, we have $\mathcal{E}[\![M_1]\!] \not\approx_\lambda \mathcal{E}[\![M_2]\!]$; the difference between the two processes can be detected by an external observer after 5 interactions. ☐

However, model $\mathcal{D}$ is fully abstract on the *affine* λ-calculus, which collects those λ-terms in which a variable may occur free at most once in any subterm. This is a consequence of results in the next section and of results by Boudol and Laneve [BL94].

Theorem 7.6 (full abstraction for $\mathcal{D}$ on the affine λ-calculus) *If M and N are affine λ-terms, then $M \approx_\lambda N$ iff $M \simeq N$.*

Proof: Boudol and Laneve [BL94] have proved that on the affine λ-calculus applicative bisimulation coincides with the Lévy-Longo Tree equality (see below). Then the result follows from Corollary 7.16. ☐

But we seek full abstraction on the whole class of λ-terms. Given a denotational interpretation which is not fully abstract, there are two natural directions to achieve full abstraction:

- to cut down the existing "over-generous" semantic domain (*restrictive approach*);
- to enrich the language (*expansive approach*).

The two approaches are exemplified by the solutions to the full abstraction problem for PCF (a typed λ-calculus extended with fixed points, boolean and arithmetic features) proposed by Milner [Mil77] and Plotkin [Plo77]; in the latter, PCF is augmented with a 'parallel or' operator.

Also in the case of our model $\mathcal{D}$ we can attempt both directions. In this paper we examine the expansive approach. We first summarise the study of the restrictive approach, reported in [San92] (but the process calculus used is the Higher-Order π-calculus rather than π-calculus). Two cuts of the model are made: Only the interior of the model is used; the behavioural equivalence on process terms is weakened. Intuitively, the latter is achieved by restricting the class of contexts in which two terms can be tested: As λ-terms are *only* used in λ-calculus contexts, so we require that their encodings be used only in encodings of λ-calculus contexts. Technically, this is expressed using *barbed bisimulation* [MS92], a bisimilarity equivalence which can be relativised on a class of contexts. Barbed bisimulation coincides with $\approx$ if powerful enough contexts are allowed, but it is coarser otherwise. The model finally obtained is not only fully abstract, but also *fully expressive*, in the sense that all objects of the domain of interpretation are λ-definable.

7.1. Expansive approach

In the expansive approach, we study λ-observation equivalence, i.e, the equivalence induced on λ-terms by model $\mathcal{D}$ and, hence, by the π-calculus encoding. The goal is to derive a *direct* characterisation of λ-observation equivalence, i.e. a characterisation not mentioning the encoding.

Propositions 7.4 and 7.5 show that applicative bisimulation is strictly coarser than λ-observation equivalence. The counterexample in Proposition 7.5 indicates that there is structure in a λ-term which is observable in a concurrency setting but not in a purely-functional setting. In particular, in concurrency we can observe *when* the input of a function is used in its body. To achieve the same discrimination, we refine applicative bisimulation. The new relation, called *open applicative bisimulation*, represents perhaps the simplest way to extend applicative bisimulation to open terms.

Definition 7.7 *A symmetric relation* $\mathcal{R} \subseteq \Lambda^o \times \Lambda^o$ *is an* open applicative bisimulation *if* $M \mathcal{R} N$ *implies:*

1. *if* $M \Longrightarrow \lambda x. M'$, *then there exists* N' *s.t.* $N \Longrightarrow \lambda x. N'$ *and* $M' \mathcal{R} N'$;
2. *if* $M \Longrightarrow x M_1 \ldots M_n$, *for some* $n \geq 0$, *then then there exist* $N_1, \ldots, N_n$ *s.t.* $N \Longrightarrow x N_1 \ldots N_n$ *and* $M_i \mathcal{R} N_i$, *for all* $1 \leq i \leq n$.

Two terms $M, N \in \Lambda^o$ *are* open applicative bisimilar, *written* $M \simeq^o N$, *if* $M \mathcal{R} N$, *for some open applicative bisimulation* $\mathcal{R}$ *.*

Clause (2), which takes care of terms with a variable in head position, was not present in the definition of applicative bisimulation, where all terms are closed. Moreover, by contrast with applicative bisimulation, in clause (1) no term instantiation on λ-abstractions is required. This simplification is possible because we work on open terms and can be justified with the congruence of $\simeq^o$. (A straightforward proof of the congruence of $\simeq^o$ utilises the full abstraction Theorems 7.14 and 7.15 and the congruence of π-calculus bisimilarity $\approx$.) Two useful facts to know are:

Lemma 7.8 *If* $M \Longrightarrow N$, *then* $M \simeq^o N$. ☐

Lemma 7.9 *Let* ρ *be a substitution from* λ*-variables to* λ*-variables. If* $M \simeq^o N$, *then* $M\rho \simeq^o N\rho$. ☐

Open applicative bisimulation is reminiscent of a tree representation of λ-terms. Indeed, it is easy to prove that it coincides with the Lévy-Longo Tree equality, which equates two terms $M, N \in \Lambda^o$ if they have the same Lévy-Longo Tree; see [San94a] (which gives a direct proof) or [Ong88] (which goes through preorders).

Lévy-Longo Tree (briefly LT) are the lazy variant of *Böhm Trees* (briefly BT), the most popular tree structure in the λ-calculus. BT's only correctly express the computational content of λ-terms in a "strict" regime, while they fail to do so in a lazy regime. For instance, in a lazy scheme, the terms $\lambda x.\Omega$ and Ω are distinguished, but since *unsolvable* [Bar84], they have identical BT's. These terms have different LT's, because LT's take into account the order of unsolvability of a term, i.e., the maximal number of λ-abstractions which the term can exhibit. LT's were introduced in [Lon83] — where they were simply called trees — developing an original idea by Levy [Lev75]; see [Lon83, San94a, Ong88] for a precise definition. We write $LT(M)$ for the Lévy-Longo Tree of M.

Theorem 7.10 (correspondence with Lévy-Longo Trees, from [San94a])
For all $M, N \in \Lambda^o$, we have $M \simeq^o N$ iff $LT(M) = LT(N)$. $\square$

7.2. The full abstraction theorems

We need a few lemmas before tackling the full abstraction theorems. Lemma 7.11 shows a decomposition property for weak bisimulation; Lemmas 7.12 and 7.13 show properties of the processes $\mathcal{O}_n\langle r_o, r_n, F_1, \ldots, F_n\rangle$, introduced in Section 5 to represent the encoding of λ-terms with a variable in head position.

For a process P, we let $\mathcal{N}_P$ be the set of names along which P can perform an action, i.e.,

$$\mathcal{N}_P = \{a \ : \ \text{for some and } P' \text{ and } \mu \text{ with subject } a,\ P \stackrel{\mu}{\Longrightarrow} P'\}\,.$$

Lemma 7.11

1. *Suppose $fn(P_1, P_2) \cap \big(\mathcal{N}_{Q_1} \cup \mathcal{N}_{Q_2}\big) = \emptyset$. Then $P_1 \mid Q_1 \approx P_2 \mid Q_2$ implies $P_1 \approx P_2$.*
2. *Let $x, q \notin fn(F, G)$. Then $!\,x(q).\,F\langle q\rangle \approx\, !\,x(q).\,G\langle q\rangle$ implies $F \approx G$.*

Proof: We first prove (1). Relation

$$\begin{aligned}\mathcal{R} = \{(P_1, P_2) \ : \ & P_1 \mid Q_1 \approx P_2 \mid Q_2 \\ & \text{for some } Q_1, Q_2 \text{ with } fn(P_1, P_2) \cap \big(\mathcal{N}_{Q_1} \cup \mathcal{N}_{Q_2}\big) = \emptyset \ \}\end{aligned}$$

is a weak bisimulation. The proof is straightforward: Since $\mathrm{fn}(P_1, P_2) \cap (\mathcal{N}_{Q_1} \cup \mathcal{N}_{Q_1}) = \emptyset$, no interaction between P_i and Q_i is possible, $i = 1, 2$. Moreover, if all bound names of actions of P_1 and P_2 are fresh, then the side condition of $\mathcal{R}$ is preserved.

Now assertion (2). We have to show that $F\langle q\rangle \approx G\langle q\rangle$. We can assume, without loss of generality, that q is different from x. Since $!\,x(q).F\langle q\rangle \approx\, !\,x(q).G\langle q\rangle$ and $!\,x(q).F\langle q\rangle \xrightarrow{x(q)} F\langle q\rangle \mid\, !\,x(q).F\langle q\rangle$, we have, for some P,

$$!\,x(q).G\langle q\rangle \stackrel{x(q)}{\Longrightarrow} P \approx F\langle q\rangle \mid\, !\,x(q).F\langle q\rangle\,. \tag{12}$$

Since x does not occur in $G\langle q\rangle$, no interaction between $G\langle q\rangle$ and $!\,x(q).G\langle q\rangle$ may have occurred; therefore P is of the form $P_G \mid\, !\,x(q).G\langle q\rangle$, for some P_G s.t.

$$G\langle q\rangle \Longrightarrow P_G. \tag{13}$$

Thus (12) can be written as $P_G \mid\, !\,x(q).G\langle q\rangle \approx F\langle q\rangle \mid\, !\,x(q).F\langle q\rangle$. From this, we get

$$P_G \approx F\langle q\rangle \tag{14}$$

using the assertion (1) of the lemma, since $\mathcal{N}_{!\,x(q).G\langle q\rangle} = \mathcal{N}_{!\,x(q).F\langle q\rangle} = \{x\}$ and x is not free in $F\langle q\rangle$ and P_G. In a symmetric way (just exchange F and G), we can derive, for some P_F,

$$F\langle q\rangle \Longrightarrow P_F\,, \qquad \text{and} \qquad P_F \approx G\langle q\rangle\,. \tag{15}$$

Now, we exploit (13-15) to show that $F\langle q\rangle \approx G\langle q\rangle$: For this, we take

$$\mathcal{R} = \{(F\langle q\rangle, G\langle q\rangle), (G\langle q\rangle, F\langle q\rangle)\} \cup \approx$$

and show that it is a weak bisimulation. Suppose $F\langle q\rangle \xrightarrow{\mu} Q_F$. We show how $G\langle q\rangle$ can match this move: Since $F\langle q\rangle \approx P_G$, we have $P_G \overset{\widehat{\mu}}{\Longrightarrow} Q_G \approx Q_F$; therefore, using (13), $G\langle q\rangle \Longrightarrow P_G \overset{\widehat{\mu}}{\Longrightarrow} Q_G \approx Q_F$, which closes the bisimulation. The case in which $G\langle q\rangle$ moves first is analogous. $\square$

Lemma 7.12 *If* $\mathcal{O}_n\langle r_o, r_n, F_1, \dots, F_n\rangle \approx \mathcal{O}_m\langle r_o, r_m, G_1, \dots, G_m\rangle$*, then* $n = m$.

PROOF: If $n \neq m$, then one of the two processes can perform more consecutive output actions than the other. $\square$

Lemma 7.13 *It holds that*

$$\mathcal{O}_n\langle r_o, r_n, F_1, \dots, F_n\rangle \approx \mathcal{O}_n\langle r_o, r_n, G_1, \dots, G_n\rangle \textit{ iff } F_i \approx G_i \textit{ for all } 1 \leq i \leq n.$$

PROOF: The implication from right to left can be inferred from the congruence properties of $\approx$. Thus, we only have to consider the implication from left to right. We proceed by induction on n. We only consider the inductive case. Let

$$P \overset{\text{def}}{=} \mathcal{O}_n\langle r_o, r_n, F_1, \dots, F_n\rangle, \qquad\qquad Q \overset{\text{def}}{=} \mathcal{O}_n\langle r_o, r_n, G_1, \dots, G_n\rangle.$$

By Lemma 5.2(1), the only transitions they can perform are

$$\begin{array}{lll} P & \xrightarrow{(\boldsymbol{\nu}\, x_1, r_1)\overline{r}_o\langle x_1, r_1\rangle} \sim & \mathcal{O}_{n-1}\langle r_1, r_n, F_2, \dots, F_n\rangle \mid !\,x_1(q).\,F_1\langle q\rangle, \\ Q & \xrightarrow{(\boldsymbol{\nu}\, x_1, r_1)\overline{r}_o(x_1, r_1)} \sim & \mathcal{O}_{n-1}\langle r_1, r_n, G_2, \dots, G_n\rangle \mid !\,x_1(q).\,G_1\langle q\rangle. \end{array}$$

Let $P_1 \overset{\text{def}}{=} \mathcal{O}_{n-1}\langle r_1, r_n, F_2, \dots, F_n\rangle$, and $Q_1 \overset{\text{def}}{=} \mathcal{O}_{n-1}\langle r_1, r_n, G_2, \dots, G_n\rangle$. The only actions that $!\,x_1(q).\,F_1\langle q\rangle$ and $!\,x_1(q).\,G_1\langle q\rangle$ can perform are at x_1 and, by Lemma 5.2(1), the only actions that P_1 and Q_1 can perform are at r_1. Since r_1 is not free in $!\,x_1(q).\,F_1\langle q\rangle$ and $!\,x_1(q).\,G_1\langle q\rangle$, and x_1 is not free in P_1 and Q_1, using Lemma 7.11(1) twice we infer

$$!\,x_1(q).F_1\langle q\rangle \approx\, !\,x_1(q).G_1\langle q\rangle \quad \text{and} \quad P_1 \approx Q_1$$

Now from the former we get $F_1 \approx G_1$ using Lemma 7.11(2); from the latter we get $F_i \approx G_i$, $2 \leq i \leq n$ using the inductive assumption. $\square$

We are now ready to prove that open applicative bisimilarity coincides with λ-observation equivalence.

Theorem 7.14 (soundness of $\mathcal{D}$ w.r.t. $\simeq^{\mathbf{o}}$) $M \approx_\lambda N$ *implies* $M \simeq^o N$.

PROOF: By Proposition 7.3, it suffices to prove that $\mathcal{E}\llbracket M \rrbracket \approx \mathcal{E}\llbracket N \rrbracket$ implies $M \simeq^{\mathrm{o}} N$. If $\mathcal{E}\llbracket M \rrbracket \approx \mathcal{E}\llbracket N \rrbracket$ then, for any p, $\mathcal{E}\llbracket M \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N \rrbracket\langle p\rangle$. We prove that

$$\mathcal{R} = \{(M,N) \;:\; \mathcal{E}\llbracket M \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N \rrbracket\langle p\rangle, \;\text{ for some } p\}$$

is an open applicative bisimulation. First, suppose $M \Longrightarrow \lambda x. M'$. We have to find N' s.t. $N \Longrightarrow \lambda x. N'$ and $(M', N') \in \mathcal{R}$. From $M \Longrightarrow \lambda x. M'$ and Proposition 5.4(2), we get

$$\mathcal{E}\llbracket M \rrbracket\langle p\rangle \overset{p(x,q)}{\Longrightarrow} \gtrsim \mathcal{E}\llbracket M' \rrbracket\langle q\rangle\,.$$

Since $\mathcal{E}\llbracket M \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N \rrbracket\langle p\rangle$, there is P'' s.t.

$$\mathcal{E}\llbracket N \rrbracket\langle p\rangle \overset{p(x,q)}{\Longrightarrow} P'' \approx \mathcal{E}\llbracket M' \rrbracket\langle q\rangle\,. \tag{16}$$

We can decompose $\mathcal{E}\llbracket N \rrbracket\langle p\rangle \overset{p(x,q)}{\Longrightarrow} P''$ into $\mathcal{E}\llbracket N \rrbracket\langle p\rangle \Longrightarrow \overset{p(x,q)}{\longrightarrow} P' \Longrightarrow P''$, for some P'. Then, using Proposition 5.5(1-2), we infer that there are N' and N'' s.t. $N \Longrightarrow \lambda x.\, N'$ and $N' \Longrightarrow N''$ with $P' \gtrsim \mathcal{E}\llbracket N' \rrbracket\langle q\rangle$ and

$$P'' \gtrsim \mathcal{E}\llbracket N'' \rrbracket\langle q\rangle\,. \tag{17}$$

Moreover, by validity of beta equality (Corollary 6.9),

$$\mathcal{E}\llbracket N' \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N'' \rrbracket\langle p\rangle\,. \tag{18}$$

Since $\lesssim \subseteq \approx$, we can combine (16), (17) and (18) and derive $\mathcal{E}\llbracket M' \rrbracket\langle q\rangle \approx \mathcal{E}\llbracket N' \rrbracket\langle q\rangle$; hence $(M', N') \in \mathcal{R}$, which concludes the case.

Now, suppose $M \Longrightarrow x M_1 \ldots M_n$, for some x and $M_1, \ldots, M_n$. We suppose $n > 0$; the case $n = 0$ is simpler. We have to find $N_1, \ldots, N_n$ s.t. $N \Longrightarrow x N_1 \ldots N_n$ and $M_i \;\mathcal{R}\; N_i$, for all i.

From Proposition 5.4(4), we get

$$\mathcal{E}\llbracket M \rrbracket\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\overline{x}\langle q\rangle}{\Longrightarrow} \gtrsim \mathcal{O}_n\langle q, p, \mathcal{E}\llbracket M_1 \rrbracket, \ldots, \mathcal{E}\llbracket M_n \rrbracket\rangle$$

and, from $\mathcal{E}\llbracket M \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N \rrbracket\langle p\rangle$ and Proposition 5.5(4), for some m and $N_1, \ldots, N_m$,

$$\mathcal{E}\llbracket N \rrbracket\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\overline{x}\langle q\rangle}{\Longrightarrow} \gtrsim \mathcal{O}_m\langle q, p, \mathcal{E}\llbracket N_1 \rrbracket, \ldots, \mathcal{E}\llbracket N_m \rrbracket\rangle \tag{19}$$

with $\mathcal{O}_n\langle q, p, \mathcal{E}\llbracket M_1 \rrbracket, \ldots, \mathcal{E}\llbracket M_n \rrbracket\rangle \approx \mathcal{O}_m\langle q, p, \mathcal{E}\llbracket N_1 \rrbracket, \ldots, \mathcal{E}\llbracket N_m \rrbracket\rangle$. From this and Lemmas 7.12 and 7.13 we infer that $m = n$ and that $\mathcal{E}\llbracket M_i \rrbracket \approx \mathcal{E}\llbracket N_i \rrbracket$, for all i. Moreover, from (19) and Proposition 5.5(4) we infer that

$$N \Longrightarrow x N_1 \ldots N_n\,.$$

Since $\mathcal{E}\llbracket M_i \rrbracket \approx \mathcal{E}\llbracket N_i \rrbracket$, for any p, $\mathcal{E}\llbracket M_i \rrbracket\langle p\rangle \approx \mathcal{E}\llbracket N_i \rrbracket\langle p\rangle$; hence $(M_i, N_i) \in \mathcal{R}$. ☐

Theorem 7.15 (completeness of $\mathcal{D}$ w.r.t. $\simeq^{\mathrm{o}}$) *$M \simeq^o N$ implies $M \approx_\lambda N$.*

PROOF: We show that

$$\mathcal{R} = \bigcup_p \{(\mathcal{E}\llbracket M \rrbracket\langle p\rangle, \mathcal{E}\llbracket N \rrbracket\langle p\rangle) \;:\; M \simeq^{\mathrm{o}} N\}\,.$$

is closed under substitutions and is a weak bisimulation up-to context and up-to $\gtrsim$. By Theorem 3.13, this implies $\mathcal{R} \subseteq \approx$.

First, we show that $\mathcal{R}$ is closed under substitutions. The free names of a process $\mathcal{E}[\![M]\!]\langle p\rangle$ are $\{p\} \cup \{x : x \in \mathrm{fv}(M)\}$. Therefore, for all name substitution σ there is a variable substitution ρ s.t.

$$\mathcal{E}[\![M]\!]\langle p\rangle\sigma = \mathcal{E}[\![M\rho]\!]\langle \sigma(p)\rangle\,.$$

Lemma 7.9 shows that $\simeq^{\mathrm{o}}$ is closed under variable substitutions; hence $\mathcal{R}$ is closed under name substitutions.

Now we show that $\mathcal{R}$ is a weak bisimulation up-to context and up-to $\gtrsim$. Suppose that $(\mathcal{E}[\![M]\!]\langle p\rangle, \mathcal{E}[\![N]\!]\langle p\rangle) \in \mathcal{R}$ and $\mathcal{E}[\![M]\!]\langle p\rangle \xrightarrow{\mu} P$; there are four cases to consider, according to whether μ is a silent, an input, a free output or a non-free output. The first three cases are simpler, so we only show the argument for the last case (to handle the last case we shall need both up-to context and up-to $\gtrsim$, whereas for the first three cases up-to $\gtrsim$ is enough).

Thus, suppose μ is a non-free output. By Proposition 5.5(4), $\mu = (\boldsymbol{\nu}\, q)\, \overline{x}\langle q\rangle$, and there are x and $M_1, \ldots, M_n$ s.t. $M = xM_1 \ldots M_n$ and $P \gtrsim \mathcal{O}_n\langle q, p, \mathcal{E}[\![M_1]\!], \ldots, \mathcal{E}[\![M_n]\!]\rangle$. Since $M \simeq^{\mathrm{o}} N$, there are $N_1, \ldots, N_n$ s.t.

$$N \Longrightarrow xN_1 \ldots N_n \tag{20}$$

and

$$M_i \simeq^{\mathrm{o}} N_i, \qquad \text{for all } 1 \le i \le n. \tag{21}$$

Now, from (20), by Proposition 5.4 we get

$$\mathcal{E}[\![N]\!]\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\, \overline{x}\langle q\rangle}{\Longrightarrow} \gtrsim \mathcal{O}_n\langle q, p, \mathcal{E}[\![N_1]\!], \ldots, \mathcal{E}[\![N_n]\!]\rangle$$

and from (21) we get

$$(\mathcal{E}[\![M_i]\!]\langle r\rangle, \mathcal{E}[\![N_i]\!]\langle r\rangle) \in \mathcal{R}\ ,$$

for all r. Summarising, we have obtained that

$$\begin{array}{rcl} \mathcal{E}[\![M]\!]\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\, \overline{x}\langle q\rangle}{\Longrightarrow} & \gtrsim & \mathcal{O}_n\langle q, p, \mathcal{E}[\![M_1]\!], \ldots, \mathcal{E}[\![M_n]\!]\rangle\,, \\ \mathcal{E}[\![N]\!]\langle p\rangle \overset{(\boldsymbol{\nu}\, q)\, \overline{x}\langle q\rangle}{\Longrightarrow} & \gtrsim & \mathcal{O}_n\langle q, p, \mathcal{E}[\![N_1]\!], \ldots, \mathcal{E}[\![N_n]\!]\rangle\,, \text{ and} \\ (\mathcal{E}[\![M_i]\!]\langle r\rangle, \mathcal{E}[\![N_i]\!]\langle r\rangle) & \in & \mathcal{R}\,, \qquad \text{for all } r \text{ and } 1 \le i \le n. \end{array}$$

This is enough, because $\mathcal{R}$ is a bisimulation up-to context and up-to $\gtrsim$. □

The "up-to context and up-to $\gtrsim$" technique simplifies a lot the proof of Theorem 7.15, which would have been almost untreatable otherwise.

We can summarise the results of Proposition 7.3, Theorems 7.10, 7.14 and 7.15:

Corollary 7.16 *For all $M, N \in \Lambda^o$ it holds that*

$$M \approx_\lambda N \textit{ iff } \mathcal{E}[\![M]\!] \approx \mathcal{E}[\![N]\!] \textit{ iff } M \simeq^o N \textit{ iff } LT(M) = LT(N).$$ □

Acknowledgements

A series of discussions with Robin Milner has inspired the work reported in the paper. I also benefited from comments by Matthew Hennessy and Gordon Plotkin.

References

[Abr89] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pp65–116. Addison-Wesley, 1989.

[AKH92] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29:737–760, 1992.

[AO93] S. Abramsky and L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993.

[Bar84] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North Holland, 1984. Revised edition.

[BL94] G. Boudol and C. Laneve. The discriminanting power of the λ-calculus with multiplicities. Draft, 1994.

[Bou89] G. Boudol. Towards a lambda calculus for concurrent and communicating systems. In *TAPSOFT '89*, *LNCS 351*, pp149–161, 1989.

[Bou94] G. Boudol. A lambda calculus for (strict) parallel functions. *Information and Computation*, 108(1):51–127, 1994.

[Bou92] G. Boudol. Asynchrony and the π-calculus. Notes., 1992.

[FMQ94] G. Ferrari, U. Montanari, and P. Quaglia. A π-calculus with explicit substitutions: the late semantics. In *Proc. MFCS'94*, LNCS . Springer Verlag, 1994.

[HL80] J.R. Hindley and G. Longo. Lambda calculus models and extensionality. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 26:289–310, 1980.

[Hon92] K. Honda. Two bisimilarities for the ν-calculus. Technical Report 92-002, Keio University, 1992.

[HS86] J.R. Hindley and J.P. Seldin. *Introduction to Combinators and λ-calculus.* Cambridge University Press, 1986.

[HT92] K. Honda and M. Tokoro. On asynchronous communication semantics. In *ECOOP '91*, *LNCS 612*. Springer Verlag, 1992.

[HaKl94] M. Hansen and J. Kleist *Process Calculi with Asynchronous Communication.* Master thesis (supervisor Hans Hüttel.). Aalborg University, August 1994.

[Lev75] J.-J. Lévy. An algebraic interpretation of equality in some models of the lambda calculus. In *Lambda Calculus and Computer Science Theory*, *LNCS 37*, pp147–165. Springer Verlag, 1975.

[Lon83] G. Longo. Set theoretical models of lambda calculus: Theory, expansions and isomorphisms. *Annales of Pure and Applied Logic*, 24:153–188, 1983.

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

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mil90] R. Milner. Functions as processes. Research Report 1154, INRIA, Sophia Antipolis, 1990. Final version in *Journal of Mathem. Structures in Computer Science* 2(2):119–141, 1992.

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. In *Logic and Algebra of Specification*, ed. F.L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag, 1993.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th ICALP*, *LNCS 623*, pp685–695. Springer Verlag, 1992.

[Ong88] 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.

[Par81] D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, *LNCS 104*. Springer Verlag, 1981.

[Plo75] G.D. Plotkin. Call by name, call by value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

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

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992.

[San93] D. Sangiorgi. An investigation into functions as processes. In *Proc. Ninth International Conference on the Mathematical Foundations of Programming Semantics (MFPS'93)*, *LNCS 802*, pp143–159. Springer Verlag, 1993.

[San94a] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Information and Computation*, 111(1):120–153, 1994.

[San94b] D. Sangiorgi. Locality and non-interleaving semantics in calculi for mobile processes. To appear in *Theoretical Computer Science.* An extract appeared in *Proc. TACS '94*, *LNCS 789*, Springer Verlag.

[SM92] D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up to". In W.R. Cleveland, editor, *Proceedings of CONCUR '92*, *LNCS 630*, pp32–46. Springer Verlag, 1992.

[Tho90] B. Thomsen. *Calculi for Higher Order Communicating Systems.* PhD thesis, Department of Computing, Imperial College, 1990.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399