On the Expressiveness of Internal Mobility in Name-Passing Calculi*

Michele Boreale
Dipartimento di Scienze dell'Informazione
Università di Roma "La Sapienza"

Abstract

We consider the language πI, a name-passing calculus introduced by Sangiorgi, where only private names can be exchanged among processes (internal mobility). The calculus πI has simple mathematical theory, very close to that of CCS. We provide an encoding from (an asynchronous variant of) the π-calculus to πI, which is fully abstract on the reduction relations of the two calculi. The result shows that, in name-passing calculi, internal mobility is the essential ingredient as far as expressiveness is concerned.

1 Introduction

By now, the π-calculus [13] is generally recognized as *the* prototypical algebraic language for describing concurrent systems with dynamically evolving communication linkage. The latter phenomenon, known as *mobility*, is modelled through the passing of channel names among processes (name-passing). The expressive power of the π-calculus is demonstrated by the existence of simple and fully abstract translations into it for a variety of computational formalisms, including λ-calculus [12], higher-order process calculi [15] and calculi which permits reasoning on the causal or spatial structure of the systems [4, 17].

The price to pay for this expressiveness is a rather complex mathematical theory of the π-calculus. A source of complications is, above all, the need to take *name instantiation* (otherwise called substitution) into account. Input and output at a of a tuple of names $\widetilde{b}$ are written, respectively, as $a(\widetilde{b}).P$ (input prefix) and $\overline{a}\langle\widetilde{b}\rangle.P$ (output prefix), with P representing the continuation of the prefix. An input and an output prefix can be consumed in a communication, where a tuple of names is passed and used to instantiate the formal parameters of the input prefix, thus:

$$a(\widetilde{c}).P \mid \overline{a}\langle\widetilde{b}\rangle.Q \xrightarrow{\tau} P\{\widetilde{b}/\widetilde{c}\} \mid Q \qquad (*)$$

with $\{\widetilde{b}/\widetilde{c}\}$ denoting the instantiation of names in $\widetilde{c}$ with names in $\widetilde{b}$. Name instantiation is a central aspect in the mathematical treatment of certain behavioural relations.

*Work partially supported by EEC, HCM Project Express and by CNR within the project "Specifica ad Alto Livello e Verifica di Sistemi Digitali". Author's e-mail address: michele@dsi.uniroma1.it

E.g., bisimilarity in the π-calculus comes in several different forms (early, late and open), depending on the name instantiation strategy chosen for matching input actions [13, 14, 16], and it is not clear which one should be preferred. Name instantiation also complicates the pragmatics of the π-calculus, since any implementation has to keep track, explicitly (e.g. using environments) or implicitly, of the bindings among names created by communications like (*) as the computation proceeds (see e.g. [8]).

It is therefore natural to try to isolate fragments of the π-calculus enjoying a simpler treatment of name instantiation, while retaining non-trivial expressive power. In this paper, we examine the calculus πI, a sub-language of the π-calculus proposed in [19]. A prominent feature of πI is that it avoids using name instantiation (other than α-conversion). This makes its mathematical treatment and its pragmatics much simpler than those of the π-calculus: indeed the only extra ingredient of πI over CCS is α-conversion of names (see [19]). We show that an "asynchronous" variant of the π-calculus [10, 3, 18] can be translated, in a simple and compositional fashion, into πI. There is a precise operational correspondence, on the reduction relations of the two calculi, between the source process and translated process. The correspondence can be also concisely stated as full abstraction of the translation w.r.t. the *barbed bisimulation* equivalence of [15]. A more precise account of our work follows.

The language πI is obtained from the π-calculus by imposing the constraint that only *private* names be communicated among processes. Output at a of a tuple of private names $\widetilde{b}$ is written as $(\nu\,\widetilde{b})(\overline{a}\langle\widetilde{b}\rangle.P)$, where $(\nu\,\widetilde{b})$ is the *restriction* operator of the π-calculus. After the interaction, the communicated names remain private:

$$a(\widetilde{c}).P \mid (\nu\,\widetilde{b})\overline{a}\langle\widetilde{b}\rangle.Q \xrightarrow{\tau} (\nu\,\widetilde{b})(P\{\widetilde{b}/\widetilde{c}\} \mid Q) \qquad (**)$$

Since both $a(\widetilde{c})$. and $(\nu\,\widetilde{b})$ act as binders for the names $\widetilde{c}$ and $\widetilde{b}$, respectively, up to α-conversion it is possible to assume in (**) that $\widetilde{b} = \widetilde{c}$: thus no name instantiation is needed in πI. The kind of dynamic reconfiguration corresponding to the passing of private names is called *internal mobility* in [19].

In πI it is impossible to directly describe *external* mobility, i.e., output of public names (or *free output*), as given by (*). In [19], it has been shown that πI is expressive enough to encode in a fully abstract way λ-calculus and certain forms of *strictly* higher-order process calculi. However, neither of these formalisms exhibits external mobility. In particular, in strictly higher- order calculi, no name-passing feature is present, since only processes (or abstraction of processes) can be passed around. It is therefore natural to wonder whether external mobility, at least in some limited form, can be "programmed" via the internal one.

In the paper, we consider asynchronous π-calculus, π_{a}, a language with external mobility introduced by Honda [10] and, independently, by Boudol [3]. This is a variant of the π-calculus where the continuation of a free output prefix is always the empty process (*asynchronous free output*), and the *matching* operator [13], used to test for equalities between names, is omitted. The limitation to asynchronous output prefix is not serious, since it has been shown that the full output prefix can be, in a reasonable sense, programmed in π_{a} [3]. On the other hand, matching plays a secondary role, as far as expressiveness is concerned (e.g. it is not necessary to encode the λ-calculus, higher-order calculi etc.), and its omission leads to a nicer mathematical treatment of many behavioural relations (e.g. on π_{a}, unlike the π-calculus, bisimulation is a full congruence [10, 9, 5]).

We define a compositional encoding, $[\![\,.\,]\!]$, from $\pi_{\rm a}$ to πI. The basic idea is that the output of a free name b at a, $\overline{a}b$, is replaced, in πI, by the output of a private name x, which acts as a pointer to a *link* process from x to b, written $x \rightarrow b$. Intuitively, $x \rightarrow b$ behaves like a buffer with entrance at x and exit at b: however, names transmitted at b are not the same as names received at x (this would require free output), but are, in turn, linked to them (the definition of link processes will be indeed recursive). Since a link $x \rightarrow b$ transforms outputs at x into outputs at b, a process owing x can trigger an output at b by interacting with $x \rightarrow b$.

Thus link processes can be used to naturally encode those $\pi_{\rm a}$-processes in which any receiver, say $a(x).P$, can only use x in P as an output channel. We call $\pi_{\rm a}^{\rm i}$ the subset of $\pi_{\rm a}$ obeying this "inversion of polarity" syntactical condition. We show that the full $\pi_{\rm a}$ can be faithfully encoded in the fragment $\pi_{\rm a}^{\rm i}$. Thus the encoding $[\![\,.\,]\!]$ is actually obtained as the composition of two simple translations: one ($\{\!|\,.\,|\!\}$) from $\pi_{\rm a}$ to $\pi_{\rm a}^{\rm i}$, and the other ($(\!|\,.\,|\!)$) from $\pi_{\rm a}^{\rm i}$ to πI. Each of these two encodings is proven to be fully abstract w.r.t. barbed bisimilarity, a behavioural equivalence which focuses on the reduction relations of process calculi. The meaning of this result is that whatever can be programmed in the π-calculus, it can be programmed in πI. This strengthens the claim of [19], that, in the π-calculus, internal mobility is responsible for most of the expressive power, whereas external mobility is responsible for most of the mathematical complications.

The encoding $\{\!|\,.\,|\!\}$ is also interesting on its own. The underlying idea is that, whenever a name b is passed, the sender keeps for himself the right of using name b as an input. In the translated process, the receiver is hence passed two things: a "polarized" b, which can be only used for output, *plus* the private address of a channel manager, to which all requests of using b as an input channel must be addressed. Thus, all subsequent communications along b will have the channel manager as a receiver. This suggests that, without loosing expressive power, it should be possible to further refine the channel discipline of $\pi_{\rm a}^{\rm i}$ to get a calculus in which each channel, once created, has a single, statically localized receiver; the latter could be understood as an *object*, in the sense of object-oriented programming. This "unique receiver" property is particularly desirable for distributed implementation of concurrent languages: it is, for example, one of the motivations behind the *join-calculus* of Fournet and Gonthier [7].

The rest of the paper is organized as follows. Section 2 contains some background material on $\pi_{\rm a}$, πI and on the behavioural relations used throughout the paper. Section 3 presents the encoding from $\pi_{\rm a}$ to $\pi_{\rm a}^{\rm i}$. Section 4 presents the encoding from $\pi_{\rm a}^{\rm i}$ to πI. The paper ends with a few conclusive remarks in Section 5.

2 Background

In this section we introduce the languages $\pi_{\rm a}$, $\pi_{\rm a}^{\rm i}$ and πI, their basic operational semantics and some behavioural relations on them.

2.1 The languages $\pi_{\rm a}$, $\pi_{\rm a}^{\rm i}$ and πI

The name-passing languages $\pi_{\rm a}$, $\pi_{\rm a}^{\rm i}$ and πI can be regarded as fragments of a common π-calculus subset, which we call $\mathcal{P}$. Below, we shall first describe $\mathcal{P}$ and then isolate out of it the fragments of our interest, by constraining the output constructs.

The countable set $\mathcal{N}$ of *names* is ranged over by $a, b, \ldots, x, y, \ldots$. A countable set of *agent identifiers*, each having a non-negative integer arity, is ranged over by $A, A', \ldots$. *Processes* are ranged over by P, Q and R. The subset of the π-calculus syntax we shall consider is built from the operators of guarded summation, restriction, parallel composition, replication and agent identifier:

$$\begin{array}{lll} P & := \sum_{i \in I} S_i \;\mid\; \nu\, a\, P \;\mid\; P_1 \,|\, P_2 \;\mid\; !\, P \;\mid\; A(a_1, \ldots, a_k) \\ S & := a(\widetilde{b}).P \;\mid\; \overline{a}(\widetilde{b}).P \;\mid\; \overline{a}\langle\widetilde{b}\rangle.\mathbf{0}\,. \end{array}$$

where k is the arity of A. The prefixes $a(\widetilde{b}).$, $\overline{a}(\widetilde{b}).$ and $\overline{a}\langle\widetilde{b}\rangle.$ are called, respectively, input prefix, bound output prefix and (asynchronous) free output prefix; in the input prefix $a(\widetilde{b})$ and in the bound output prefix $\overline{a}(\widetilde{b})$, the components of $\widetilde{b}$ are pairwise distinct. In the free output $\overline{a}\langle\widetilde{b}\rangle$, we omit the surrounding brackets $\langle\rangle$ when $\widetilde{b}$ has one or zero components. In summations, the index-set I is finite; for $\sum_{i \in \emptyset} \alpha_i.P_i$ the symbol $\mathbf{0}$ is also used, while binary summation $\sum_{i \in \{1,2\}} P_i$ is often written as $P_1 + P_2$. We abbreviate $\alpha.\,\mathbf{0}$ as α and $\nu\, a\, \nu\, b\, P$ as $(\nu\, a, b\,)P$.

We only admit *guarded* summation, since, by contrast with full summation, it preserves bisimilarity even for weak relations, where silent moves are partially ignored. Following [19], we have also introduced explicitly the bound output prefix $\overline{a}(\widetilde{b}).P$, that in in the full π-calculus would only be syntactic sugar for $\nu\, \widetilde{b}\, (\overline{a}\langle\widetilde{b}\rangle.P)$.

Input prefix $a(\widetilde{b}).$ and restriction $\nu\, a$ act as *binders* for names $\widetilde{b}$ and a, respectively. *Free names*, *bound names* of a process P, written $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ respectively, arise as expected; the *names* of P, written $\mathrm{n}(P)$ are $\mathrm{fn}(P) \cup \mathrm{bn}(P)$. *Substitutions*, ranged over by $\sigma, \sigma' \ldots$ are functions from $\mathcal{N}$ to $\mathcal{N}$; for any expression E, we write $E\sigma$ for the expression obtained from applying σ to E. Composition of two substitutions σ and σ' is written $\sigma\sigma'$. We assume the following decreasing order of precedence when writing process expressions: substitution, prefix, replication, restriction, parallel composition, summation.

Each agent identifier has an associated defining equation, $A(x_1, \ldots, x_k) \Leftarrow P$, where k is the arity of A, the x_i's are all distinct and $\mathrm{fn}(P) \subseteq \{x_1, \ldots, x_k\}$.

The transition rules for the language operators are given in Table 1. *Actions*, ranged over by μ, can be of four forms: τ (interaction), $a(\widetilde{b})$ (input), or $\nu\, \widetilde{b}'\, \overline{a}\langle\widetilde{b}\rangle$ (output), $\overline{a}(\widetilde{b})$ (bound output). By convention, we shall identify actions $\nu\, \widetilde{b}\, \overline{a}\langle\widetilde{b}\rangle$ and $\overline{a}(\widetilde{b})$. Functions $\mathrm{bn}(\cdot)$, $\mathrm{fn}(\cdot)$ and $\mathrm{n}(\cdot)$ are extended to actions as expected, once we set $\mathrm{bn}(a(\widetilde{b})) = \widetilde{b}$ and $\mathrm{bn}(\nu\, \widetilde{b}'\, \overline{a}\langle\widetilde{b}\rangle) = \widetilde{b}'$.

Throughout the paper, we work up to α-conversion on names — that is, we implicitly take an underlying representation of names based on de Bruijn indices [6] — so to avoid tedious side conditions in transition rules and bisimulation clauses. Therefore, for instance, in a process bound names are assumed different from each other and from the free names, and α-equivalent processes are assumed to have the same transitions. All our notations are extended to tuples componentwise.

Following Milner [11], we only admit *well-sorted processes*: the sorting prevents arity mismatching in communications, like in $\overline{a}\langle b, c\rangle.\, P \mid a(x).\, Q$. Moreover, substitutions must map names onto names of the same sort. We do not present the sorting system because it is not essential to understand the contents of this paper.

We say that a name a occurs in P in *input- (resp. output-) subject position* if P

$$\texttt{Sum}: \ \textstyle\sum_{i\in I}\alpha_i.P_i \xrightarrow{\alpha_j} P_j,\ j\in I \qquad \texttt{Rep}: \ \dfrac{P \mid !\,P \xrightarrow{\mu} P'}{!\,P \xrightarrow{\mu} P'}$$

$$\texttt{Par}: \ \dfrac{P_1 \xrightarrow{\mu} P_1'}{P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2} \qquad \texttt{Com}: \ \dfrac{P_1 \xrightarrow{(\nu\,\widetilde{b}')\overline{a}\langle\widetilde{b}\rangle} P_1' \qquad P_2 \xrightarrow{a(\widetilde{c})} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} \nu\,\widetilde{b}'\,(P_1' \mid P_2'\{\widetilde{b}/\widetilde{c}\})}$$

$$\texttt{Res}: \ \dfrac{P \xrightarrow{\mu} P'}{\nu\, c\, P \xrightarrow{\mu} \nu\, c\, P'},\ c \notin \mathrm{n}(\mu) \qquad \texttt{Open}: \ \dfrac{P \xrightarrow{(\nu\,\widetilde{b}')\overline{a}\langle\widetilde{b}\rangle} P'}{\nu\, c\, P \xrightarrow{(\nu\,\widetilde{b}'c)\overline{a}\langle\widetilde{b}\rangle} P'},\ c \neq a, c \in \widetilde{b} - \widetilde{b}'$$

$$\texttt{Ide}: \ \dfrac{P\{\widetilde{b}/\widetilde{x}\} \xrightarrow{\mu} P'}{A(\widetilde{b}) \xrightarrow{\mu} P'} \ \text{if } A(\widetilde{x}) \Leftarrow P$$

Table 1: Operational semantics of $\mathcal{P}$ (symmetric versions of `Par, Com, Close` omitted).

contains a prefix $a(\widetilde{b})$ (resp. $\overline{a}\langle\widetilde{b}\rangle$ or $\overline{a}(\widetilde{b})$) not inside the scope of a binder for a. We call:

- $\mathcal{P}$ the above defined set of π-calculus processes;
- π_{a} the subset of $\mathcal{P}$ with no bound output prefixes and no agent identifiers;
- $\pi_{\mathrm{a}}^{\mathrm{i}}$ the subset of π_{a} in which, for terms of the form $a(\widetilde{b}).P$, no $b_i \in \widetilde{b}$ occurs in P in input subject position;
- $\pi\mathrm{I}$ the subset of $\mathcal{P}$ without free output prefix.

Note that the language $\pi\mathrm{I}$ contains both replication and agent identifiers: contrary to what happens in the π-calculus, these two primitives are not equivalent (see [19]) in $\pi\mathrm{I}$. Even though replication can be derived from identifiers, we decided to keep it for notational convenience.

2.2 Barbed bisimilarity, standard bisimilarity and expansion preorder

Weak barbed bisimilarity [15] is the relation we are most interested in. We will, however, use in some of the proofs a few auxiliary relations: standard (strong and weak) bisimilarities and the expansion preorder.

In the sequel, we let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$, let $\overset{\mu}{\Longrightarrow}$ be $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$, and let $P \overset{\widehat{\mu}}{\Longrightarrow} Q$ be $P \overset{\mu}{\Longrightarrow} Q$, if $\mu \neq \tau$, and $P \Longrightarrow Q$, if $\mu = \tau$.

2.2.1 Barbed bisimilarity

Barbed bisimilarity [15] represents a uniform mechanism for defining behavioural equivalences, which focuses on two concepts common to different process calculi: the *reduction relation* $\xrightarrow{\tau}$ and an *observability predicate* $\downarrow a$. In π-calculus, we say that

P *commits* to a, and write $P \downarrow a$, if P contains a prefix $a(\widetilde{b})$, or $\overline{a}\langle\widetilde{b}\rangle$ or $\overline{a}(\widetilde{b})$ which is not underneath another prefix or in the scope of a $\nu\, a$ restriction operator. This means that P is capable of interacting immediately on channel a. We write $P \Downarrow a$ if P is capable of interacting on a possibly after a few invisible steps, i.e. if $P \Longrightarrow\downarrow a$.

Definition 2.1 (weak barbed bisimilarity) *A symmetric binary relation $\mathcal{R}$ on $\mathcal{P} \times \mathcal{P}$ is a* weak barbed bisimulation *if and only if, whenever $P\,\mathcal{R}\,Q$:*

1. *$P \xrightarrow{\tau} P'$ implies there exists Q' s.t. $Q \Longrightarrow Q'$ and $P'\,\mathcal{R}\,Q'$, and*

2. *$P \downarrow a$ implies $Q \Downarrow a$, for any a.*

We say that P and Q are barbed bisimilar*, written $P \simeq Q$ if and only if $P\,\mathcal{R}\,Q$ for some barbed bisimulation $\mathcal{R}$.*

2.2.2 Standard bisimilarities

A few forms of (standard) bisimilarity have been proposed for the π-calculus, notably the *late, early* and *open* bisimilarities [13, 16], depending on the specific name instantiation strategy adopted for input actions. Here, we take advantage of the fact that, over the subsets of the π-calculus we are interested in ($\pi\mathrm{I}$, π_{a} and hence $\pi_{\mathrm{a}}^{\mathrm{i}}$), these forms coincide with each other and with another, simpler form of bisimilarity, called *ground* bisimilarity (see [9, 10, 18, 5]). In the latter, no name instantiation of the input formal parameter is required, apart from α-conversion. We recall its definition below.

Definition 2.2 (strong ground bisimilarity) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a* strong ground bisimulation *if $P\,\mathcal{R}\,Q$ and $P \xrightarrow{\mu} P'$ imply that 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 $\mathcal{R}$.*

The weak versions of this bisimulation, where one ignores silent steps in matching transitions, is obtained in the usual way: weak ground bisimilarity is defined by replacing in Definition 2.2 the transition $Q \xrightarrow{\mu} Q'$ with $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$. We use $\approx$ for weak ground bisimilarity.

Since we are only interested in the π_{a} and $\pi\mathrm{I}$ fragments of $\mathcal{P}$, where all mentioned forms of standard bisimilarity coincide, in the sequel we shall omit the adjective 'ground' when referring to $\sim$ and $\approx$.

2.2.3 Expansion preorder

The expansion relation $\lesssim$ [1, 20] is an asymmetric variant of $\approx$ which allows us to count the number of τ-actions performed by the processes. Thus, $P \lesssim Q$ holds if $P \approx Q$ but also Q has at least as many τ-moves as P. As for standard bisimilarities, different (ground, early, late, open) forms of expansion can be defined on the π-calculus, depending on the chosen name instantiation strategy. Again, it is easily seen that all these forms coincide on the subsets of the π-calculus of our interest, π_{a} and $\pi\mathrm{I}$ (the proof parallels that given in [9, 10, 18] for standard bisimilarities). We give below the definition of ground expansion preorder, omitting the adjective 'ground'.

Definition 2.3 (expansion preorder) *A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ 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 $\mathcal{R}$.*

We often write $Q \gtrsim P$ in place of $P \lesssim Q$. The following proposition summarizes a few properties of the behavioural relations considered in the paper, and some relationships between them:

Proposition 2.4

a) *Over the languages π_{a} and $\pi\mathrm{I}$, the relations $\sim$, $\approx$ and $\lesssim$ are preserved by all operators and by name instantiations.*

b) *The following is a chain of strict inclusions between relations: $\sim$, $\lesssim$, $\approx$, $\simeq$.*

3 From π_{a} to $\pi_{\mathrm{a}}^{\mathrm{i}}$

Let us illustrate informally how a π_{a} process can be translated in $\pi_{\mathrm{a}}^{\mathrm{i}}$. The basic idea is that whenever a name b is passed, the receiver, say R, is also passed the (private) address z of an "input manager" process, $z \hookrightarrow b$. The latter serves all requests of using b as an input channel. In particular, whenever activated at z, $z \hookrightarrow b$ performs the requested input and then gives the control and the result of the input operation back at a private return address, h. Hence, all input actions of R at b are transformed, via the encoding, into interactions at z with $z \hookrightarrow b$.

For notational simplicity, we only present below the encoding for the monadic fragment of π_{a}. The polyadic case will be easily accommodated afterward. First, the formal definition of the input manager process:

Definition 3.1 (input manager process) *Let z and b be two names. An* input manager for *b* at *z is the $\pi_{\mathrm{a}}^{\mathrm{i}}$ process:*

$$z \hookrightarrow b \overset{\mathrm{def}}{=} !\, z(h).b(x,y).\overline{h}\langle x,y\rangle\,.$$

The encoding $\{\!|\,.\,|\!\}$ from (monadic) π_{a} to $\pi_{\mathrm{a}}^{\mathrm{i}}$ is defined in Table 2. The definition makes use of an auxiliary parameter, ρ, which is a finite partial function from $\mathcal{N}$ to $\mathcal{N}$. It is used in the input clause $(a(x).P)$ to record the transformation of input actions at x into interactions at z with links of the form $z \hookrightarrow b$. The notation $\rho[x/y]$ denotes the partial function which yields y on x and behaves like ρ elsewhere. Furthermore, $\mathrm{ran}(\rho)$ denotes the set $\{y : \rho(x) = y$, for some $x\}$. When, in some statement, we declare a name to be *fresh* we mean it is different from any name occurring in any process or in any function ρ previously mentioned in the statement. Bound names are always assumed to be fresh.

Before proving full abstraction of $\{\!|\,.\,|\!\}$ we need to fix a basic property of it. In the following lemma, part 1 is a well-known distributivity law for $!$, due to Milner [11]. Part 2 shows that, under certain conditions, an input manager process $z \hookrightarrow a$ somehow acts as a substitution of z with a, if z is hidden.

Let ρ be a finite partial function from $\mathcal{N}$ to $\mathcal{N}$. $\{\!|P|\!\}\rho$ is defined as:

$$\{\!|a(x).P|\!\}\rho \stackrel{\text{def}}{=} \begin{cases} a(x,z).\{\!|P|\!\}\rho[{}^{z}\!/\!{}_{x}] & \text{if } \rho \text{ is undefined on } a, \text{ with } z \text{ fresh} \\ \nu\, h\left(\overline{z}h \mid h(x,y).\{\!|P|\!\}\rho[{}^{y}\!/\!{}_{x}]\right) & \text{if } \rho(a)=z, \text{ with } h,y \text{ fresh} \end{cases}$$

$$\begin{array}{rclrcl}
\{\!|\overline{a}b|\!\}\rho & \stackrel{\text{def}}{=} & \nu\, z\left(\overline{a}\langle b,z\rangle \mid z \hookrightarrow b\right) \text{ with } z \text{ fresh} & \{\!|\nu\, x\, P|\!\}\rho & \stackrel{\text{def}}{=} & \nu\, x\,\{\!|P|\!\}\rho \\
\{\!|\sum_{i\in I} S_i|\!\}\rho & \stackrel{\text{def}}{=} & \sum_{i\in I}\{\!|S_i|\!\}\rho & \{\!|\,!\,P|\!\}\rho & \stackrel{\text{def}}{=} & !\,\{\!|P|\!\}\rho \\
\{\!|P \mid Q|\!\}\rho & \stackrel{\text{def}}{=} & \{\!|P|\!\}\rho \mid \{\!|Q|\!\}\rho & \text{Define: } \{\!|P|\!\} & \stackrel{\text{def}}{=} & \{\!|P|\!\}\emptyset\ .
\end{array}$$

Table 2: Definition of the encoding $\{\!|\,.\,|\!\}$ from π_{a} to $\pi_{\mathrm{a}}^{\mathrm{i}}$.

Lemma 3.2 *Let P, P_1 and P_2 be processes in π_{a}.*

1. $\nu\, y\left(\,!\,y(x).P \mid P_1 \mid P_2\right) \sim \nu\, y\,(\,!\,y(x).P \mid P_1) \mid \nu\, y\,(\,!\,y(x).P \mid P_2)$*, provided that y may occur free in P, P_1 and P_2 only in output-subject position.*

2. $\nu\, z\left(\, z \hookrightarrow a \mid \{\!|P|\!\}\rho[{}^{z}\!/\!{}_{a}]\right) \gtrsim [\![P]\!]\rho$*, where ρ is undefined in a, and z is fresh.*

Proof: Part 1 is shown by exhibiting the appropriate bisimulation (see e.g. [11]). Part 2 is proven by induction on P, exploiting part 1. The most interesting case is when $P = a(x).P'$. Then we have:

$$\begin{array}{rl}
\nu\, z\left(\, z \hookrightarrow a \mid \{\!|P|\!\}\rho[{}^{z}\!/\!{}_{a}]\right) & = \\
\text{(definition of } \{\!|\,.\,|\!\}) & \\
\nu\, z\left(\, z \hookrightarrow a \mid \nu\, h\left(\overline{z}\langle h\rangle \mid h(x,y).\{\!|P'|\!\}\rho[{}^{z}\!/\!{}_{a}][{}^{y}\!/\!{}_{x}]\right)\right) & \sim \\
\text{(laws for } \nu\, h \text{ and part 1)} & \\
\nu\, h\left(\nu\, z\left(\, z \hookrightarrow a \mid \overline{z}\langle h\rangle\right) \mid \nu\, z\left(\, z \hookrightarrow a \mid h(x,y).\{\!|P'|\!\}\rho[{}^{z}\!/\!{}_{a}][{}^{y}\!/\!{}_{x}]\right)\right) & \gtrsim \\
\text{(laws for } \gtrsim\,,\ \mid \text{ and } \nu\, z\,) & \\
\nu\, h\left(a(x,y).\overline{h}\langle x,y\rangle \mid h(x,y).\nu\, z\left(\, z \hookrightarrow a \mid \{\!|P'|\!\}\rho[{}^{z}\!/\!{}_{a}][{}^{y}\!/\!{}_{x}]\right)\right) & \gtrsim \\
\text{(induction hyp., laws for } \mid,\ \nu\, h\,) & \\
a(x,y).\nu\, h\left(\overline{h}\langle x,y\rangle \mid h(x,y).\{\!|P'|\!\}\rho[{}^{y}\!/\!{}_{x}]\right) & \gtrsim \\
\text{(a simple law for } \gtrsim\,) & \\
a(x,y).\{\!|P'|\!\}\rho[{}^{y}\!/\!{}_{x}] & = \\
\text{(definition of } \{\!|\,.\,|\!\}). & \\
\{\!|P|\!\}\rho\,. &
\end{array}$$

□

The following proposition shows the tight correspondence between transitions of P and transitions of $\{\!|P|\!\}\rho$.

Proposition 3.3 (strong operational correspondences) *Let ρ and P be s.t.* $\mathrm{fn}(P)\cap \mathrm{ran}(\rho)=\emptyset$.

a) Suppose that $P \xrightarrow{\mu} P'$. Then we have:

1. $\mu = a(x)$ *implies* $\{\!|P|\!\}\rho \xrightarrow{a(x,y)}\ \gtrsim \{\!|P'|\!\}\rho[{}^{y}\!/\!{}_{x}]$;

2. *$\mu = \overline{a}b$ implies $\{\!|P|\!\}\rho \xrightarrow{\nu\, y\, \overline{a}\langle b,y\rangle} \;\gtrsim\; y \hookrightarrow b \mid \{\!|P'|\!\}\rho$, with $y \notin \mathrm{fn}(P')$;*
3. *$\mu = \nu\, b\overline{a}b$ implies $\{\!|P|\!\}\rho \xrightarrow{(\nu\, b,y)\overline{a}\langle b,y\rangle} \;\gtrsim\; y \hookrightarrow b \mid \{\!|P'|\!\}\rho$, with $y \notin \mathrm{fn}(P')$;*
4. *$\mu = \tau$ implies $\{\!|P|\!\}\rho \xrightarrow{\tau} \;\gtrsim\; \{\!|P'|\!\}\rho$.*

b) The converse of part a), i.e.: Suppose that $\{\!|P|\!\}\rho \xrightarrow{\mu} P_1$. Then there is $P' \in \pi_{\mathrm{a}}$ s.t.:

1. *$\mu = a(x,y)$ implies $P \xrightarrow{a(x)} P'$ with $P_1 \gtrsim \{\!|P'|\!\}\rho[{}^{y}\!/\!{}_{x}]$;*
2. *$\mu = \nu\, y\, \overline{a}\langle b,y\rangle$ implies $P \xrightarrow{\overline{a}b} P'$ with $y \notin \mathrm{fn}(P')$ and $P_1 \gtrsim y \hookrightarrow b \mid \{\!|P'|\!\}\rho$;*
3. *$\mu = (\nu\, b,y)\overline{a}\langle b,y\rangle$ implies $P \xrightarrow{\nu\, b\, \overline{a}b} P'$ with $y \notin \mathrm{fn}(P')$ and $P_1 \gtrsim y \hookrightarrow b \mid \{\!|P'|\!\}\rho$;*
4. *$\mu = \tau$ implies $P \xrightarrow{\tau} P'$, with $P_1 \gtrsim \{\!|P'|\!\}\rho$.*

PROOF: Each part is proven by transition induction. The only subtle points arise in the proof of parts a)(4) and b)(4), where also Lemma 3.2(2) is used. As an example, we show part a)(4). The only non-trivial case is when the last rule applied for deriving $P \xrightarrow{\tau} P'$ is a communication rule (we suppose for simplicity that the communicated name is free; the case when it is restricted can be easily accommodated):

$$\mathtt{Com} : \frac{P_1 \xrightarrow{\overline{a}b} P_1',\, P_2 \xrightarrow{a(x)} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1 \mid P_2\{{}^{b}\!/\!{}_{x}\}}\,.$$

where we suppose that $x \notin \mathrm{fn}(P_1')$. By induction hypothesis, we have that:

$$\{\!|P_1|\!\}\rho \xrightarrow{\nu\, y\, \overline{a}\langle b,y\rangle} \;\gtrsim\; y \hookrightarrow b \,|\, \{\!|P_1'|\!\}\rho \text{ with } y \notin \mathrm{fn}(P_1') \text{ and } \{\!|P_2|\!\}\rho \xrightarrow{a(x,y)} \;\gtrsim\; \{\!|P_2'|\!\}\rho[{}^{y}\!/\!{}_{x}]\,. \quad (1)$$

Then we have:

$$\begin{array}{lll}
\{\!|P_1 \mid P_2|\!\}\rho \;\xrightarrow{\tau} & \gtrsim & \nu\, y\Big(\, y \hookrightarrow b \mid \{\!|P_1'|\!\}\rho \mid \{\!|P_2'|\!\}\rho[{}^{y}\!/\!{}_{x}]\{{}^{b}\!/\!{}_{x}\}\Big) \\
& & \text{(from (1) and interaction)} \\
& = & \nu\, y\Big(\, y \hookrightarrow b \mid \{\!|P_1' \mid P_2'|\!\}\rho[{}^{y}\!/\!{}_{x}]\Big)\{{}^{b}\!/\!{}_{x}\} \\
& & \text{(since } x,y \notin \mathrm{fn}(P_1') \text{ and by def. of } \{\!|\,.\,|\!\}) \\
& \gtrsim & \{\!|P_1' \mid P_2'|\!\}\rho\{{}^{b}\!/\!{}_{x}\} \quad \text{(Lemma 3.2(2))} \\
& = & \{\!|P_1' \mid P_2'\{{}^{b}\!/\!{}_{x}\}|\!\}\rho \\
& & \text{(by a simple property of the encoding and the fact} \\
& & \text{that } x \notin \mathrm{fn}(P_1')).
\end{array}$$

$\square$

As a consequence of the previous proposition, we get the following correspondence on commitments and weak invisible transitions:

Proposition 3.4

a) $P \downarrow a$ if and only if $\{\!|P|\!\} \downarrow a$.

b) $P \Longrightarrow P'$ implies $\{\!|P|\!\} \Longrightarrow \;\gtrsim\; \{\!|P'|\!\}$;

c) $\{\!|P|\!\} \Longrightarrow P_1$ *implies that there is* P' *s.t.* $P \Longrightarrow P'$ *and* $P_1 \gtrsim \{\!|P'|\!\}$.

d) $P \Downarrow a$ *if and only if* $\{\!|P|\!\} \Downarrow a$.

PROOF: Part a) is a trivial consequence of the previous lemma. Part d) is a consequence of parts a), b) and c). Part b) and c) are shown by exploiting Proposition 3.3, parts a(4) and b(4), and the properties of $\lesssim$. As an example, we show part c).

For some $n \geq 0$ it holds that $\{\!|P|\!\} \xrightarrow{\tau^n} P_1$. We proceed by induction on n. The case $n = 0$ is trivial. If $n > 0$, the for some Q, we have $\{\!|P|\!\} \xrightarrow{\tau^{n-1}} Q \xrightarrow{\tau} P_1$. By the induction hypothesis, we have, for some P'' in π_{a},

$$P \Longrightarrow P'' \quad \text{with } Q \gtrsim \{\!|P''|\!\}\,.$$

From this and $Q \xrightarrow{\tau} P_1$, we deduce that, for some R in $\pi_{\mathrm{a}}^{\mathrm{i}}$,

$$\{\!|P''|\!\} \xrightarrow{\hat{\tau}} R \quad \text{with } P_1 \gtrsim R\,.$$

Now, by Proposition 3.3(b)(4), there is P' in π_{a} s.t.

$$P'' \xrightarrow{\hat{\tau}} P' \quad \text{with } R \gtrsim \{\!|P'|\!\}.$$

Thus, we have found P' s.t. $P \Longrightarrow P'$ and $P_1 \gtrsim \{\!|P'|\!\}$, and proved the thesis. □

Remark 3.5 In the proof of item (c) of the above proposition the use of the expansion relation turns out to be necessary to close up the induction. Had we used weak bisimilarity $\approx$ in place of $\gtrsim$ in the above proof, from $Q \approx \{\!|P''|\!\}$ and $Q \xrightarrow{\tau} P_1$, we could have only inferred $\{\!|P''|\!\} \Longrightarrow R$ (in place of the stronger $\{\!|P''|\!\} \xrightarrow{\hat{\tau}} R$); as a consequence, we could not have applied Proposition 3.3(b)(4) to close up the induction. □

A simple proof technique for barbed bisimilarity:

Definition 3.6 *A symmetric binary relation* $\mathcal{R}$ *over* π_{a} *is a* barbed bisimulation up to expansion *if, whenever* $P \,\mathcal{R}\, Q$ *it holds:*

a) $P \xrightarrow{\tau} P'$ *implies that there exist* P_1, Q' *and* Q_1 *s.t.:* $P' \gtrsim P_1$ *and* $Q \Longrightarrow Q' \gtrsim Q_1$ *and* $P_1 \,\mathcal{R}\, Q_1$.

b) $P \downarrow a$ *implies* $Q \Downarrow a$.

Lemma 3.7 *If* $\mathcal{R}$ *is a barbed bisimulation up to expansion then* $\mathcal{R} \subseteq \simeq$.

We arrive at the main theorem of the section:

Theorem 3.8 (full abstraction of $\{\!|\,.\,|\!\}$ w.r.t. barbed bisimilarity) *Let* P *and* Q *be processes in* π_{a}. *Then* $P \simeq Q$ *if and only if* $\{\!|P|\!\} \simeq \{\!|Q|\!\}$.

PROOF: Exploiting the above Proposition 3.4 and Proposition 3.3, it is easy to show that the relation:

$$\mathcal{R} = \{(\{\!|P|\!\}, \{\!|Q|\!\}) : P \simeq Q\}$$

is a barbed bisimulation up to expansion in $\pi_{\mathrm{a}}^{\mathrm{i}}$: this establishes the 'if' part.

For the 'only if' part, again exploiting the above Proposition 3.4 and Proposition 3.3, it is easy to see that the relation:

$$\mathcal{R} = \{(P, Q) : \{\!|P|\!\} \simeq \{\!|Q|\!\}\}$$

is a barbed bisimulation up to expansion in π_{a}. □

We indicate now the modifications necessary to extend $\{\!| \,.\, |\!\}$ to the full polyadic π_{a}. We use the following notations: for $\widetilde{u} = (u_1, \dots, u_k)$ and $\widetilde{v} = (v_1, \dots, v_k)$, $[\widetilde{v}/\widetilde{u}]$ stands for $[v_1/u_1] \cdots [v_k/u_k]$ and $\widetilde{u} \hookrightarrow \widetilde{v}$ stands for $u_1 \hookrightarrow v_1 \mid \cdots \mid u_k \hookrightarrow v_k$. The clauses for input and output prefixes of Table 2 are replaced by the two clauses:

$$\begin{array}{rcl}\{\!|a(\widetilde{x}).P|\!\}\rho & \stackrel{\mathrm{def}}{=} & \left\{\begin{array}{ll} a(\widetilde{x}, \widetilde{z}).\{\!|P|\!\}\rho[\widetilde{z}/\widetilde{x}] & \text{if } \rho \text{ is undefined on } a, \text{ with } \widetilde{z} \text{ fresh} \\ \nu\, h\left(\overline{z}h \mid h(\widetilde{x}, \widetilde{y}).\{\!|P|\!\}\rho[\widetilde{y}/\widetilde{x}]\right) & \text{if } \rho(a) = z, \text{ with } h \text{ and } \widetilde{y} \text{ fresh}\end{array}\right. \\ \{\!|\overline{a}\langle\widetilde{b}\rangle|\!\}\rho & \stackrel{\mathrm{def}}{=} & (\nu\, \widetilde{z})\left(\overline{a}\langle\widetilde{b}, \widetilde{z}\rangle \mid \widetilde{z} \hookrightarrow \widetilde{b}\right) \quad \widetilde{z} \text{ fresh}\end{array}$$

with the obvious requirements on the number of components of $\widetilde{z}$ and $\widetilde{y}$, which must furthermore be all distinct. The proofs carry over with some straightforward (mostly notational) changes. We omit the details.

4 From $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI

Let us explain informally the second step of our translation, from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI. The basic idea is that the output of a free name b is replaced by the output of a bound name x plus a *link* from x to b, $x \to b$. The latter transforms outputs at x into outputs at b. Intuitively, $x \to b$ behaves like a buffer with entrance at x and exit at b: however, the name transmitted at b is not the same as the one received at x, but just, recursively, *linked* to it. Link processes have been introduced in [19], where they have been used to encode the lazy λ-calculus into πI.

Definition 4.1 (link processes, [19]) *Let a and b be two names. A* link from a to b *is the recursively defined* πI *process:*

$$a \to b \;\Leftarrow\; !\, a(x).\overline{b}(y).\, y \to x \;.$$

The encoding $\langle\!| \,.\, |\!\rangle$ from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI is defined in Table 3. Again, we present the encoding for the monadic fragment of $\pi_{\mathrm{a}}^{\mathrm{i}}$. The polyadic calculus will be accommodated afterward.

In order to prove full abstraction of $\langle\!| \,.\, |\!\rangle$, we need to fix a few properties of link processes. In the next lemma, part 1 says that whenever the exit point of one link coincides with the entrance point of another one, and this common point is hidden, then the two links are, so to speak, connected. This means that they behave as a single link. Part 2 of the lemma states then, under certain conditions, a link acts as a substitution.

$\langle\!| P |\!\rangle$ is defined as:

$$\begin{array}{rcl@{\qquad}rcl}
\langle\!| a(x).P |\!\rangle & \stackrel{\mathrm{def}}{=} & a(x).\langle\!| P |\!\rangle & \langle\!| \overline{a}b |\!\rangle & \stackrel{\mathrm{def}}{=} & \overline{a}(x).\, x \rightarrow b \;\; x \text{ fresh} \\
\langle\!| \sum_{i \in I} S_i |\!\rangle & \stackrel{\mathrm{def}}{=} & \sum_{i \in I} \langle\!| S_i |\!\rangle & \langle\!| \nu\, x\, P |\!\rangle & \stackrel{\mathrm{def}}{=} & \nu\, x\, \langle\!| P |\!\rangle \\
\langle\!| P \mid Q |\!\rangle & \stackrel{\mathrm{def}}{=} & \langle\!| P |\!\rangle \mid \langle\!| Q |\!\rangle & \langle\!| \,!\, P |\!\rangle & \stackrel{\mathrm{def}}{=} & !\, \langle\!| P |\!\rangle\,.
\end{array}$$

Table 3: Definition of the encoding $\langle\!| \,.\, |\!\rangle$, from $\pi^{\mathrm{i}}_{\mathrm{a}}$ to $\pi\mathrm{I}$.

Lemma 4.2

1. *Let x and y be different from z. Then $\nu\, y\, (\, x \rightarrow y \mid y \rightarrow z\,) \gtrsim x \rightarrow z\,.$*
2. *Let P be a process in π_{a} and suppose that y does not occur free in P in input-subject position. Then $\nu\, y \big(\, y \rightarrow a \mid \langle\!| P |\!\rangle \big) \gtrsim \langle\!| P\{a/y\} |\!\rangle\,.$*

PROOF: Part 1 is proven by exhibiting the appropriate expansion relation.

Part 2 is proven by induction on P and exploiting part 1. The most interesting case is when $P = \overline{y}c$, for some c. Then we have:

$$\begin{array}{rcl}
\nu\, y\, (\, y \rightarrow a \mid \langle\!| P |\!\rangle) & = & \nu\, y \big(\, y \rightarrow a \mid \overline{y}(x).\, x \rightarrow c\, \big) \\
& & \text{(def. of } \langle\!| \,.\, |\!\rangle) \\
& \sim & \nu\, y \big(y(x).(a(w).\, w \rightarrow x \mid y \rightarrow a\,) \mid \overline{y}(x).\, x \rightarrow c\, \big) \\
& & \text{(def. of } y \rightarrow a \text{ and laws for } !\,) \\
& \gtrsim & (\nu\, y, x) \big(a(w).\, w \rightarrow x \mid y \rightarrow a \mid x \rightarrow c\, \big) \\
& & \text{(a simple law for } \gtrsim\,) \\
& \sim & a(w).(\nu\, y, x) \big(\, w \rightarrow x \mid x \rightarrow c \mid y \rightarrow a\, \big) \\
& & \text{(laws for } \mid \text{ and } (\nu\, y, x)) \\
& \sim & a(w).\nu\, x \big(\, w \rightarrow x \mid \nu\, y\, (\, x \rightarrow c \mid y \rightarrow a\,)\big) \\
& & \text{(laws for } \nu\, y\,) \\
& \stackrel{\mathrm{def}}{=} & P_1\,.
\end{array}$$

Now, we have to distinguish whether $c = y$ or $c \neq y$. Suppose that $c = y$ (the case $c \neq y$ can be easily accommodated). Then applying part 1 of the lemma we get $\nu\, y\, (\, x \rightarrow c \mid y \rightarrow a\,) \gtrsim x \rightarrow a$, by which we have:

$$\begin{array}{rcll}
P_1 & \gtrsim & a(w).\nu\, x\, (\, w \rightarrow x \mid x \rightarrow a\,) & \\
& \gtrsim & a(w).\, w \rightarrow a & \text{(applying part 1 again)} \\
& = & \langle\!| \overline{x}x\{a/x\} |\!\rangle & \text{(def. of } \langle\!| \,.\, |\!\rangle).
\end{array}$$

$\square$

The following proposition shows the tight correspondence between transitions of P and transitions of $\langle\!| P |\!\rangle$.

Proposition 4.3 (strong operational correspondences) *Let P be a process in $\pi^{\mathrm{i}}_{\mathrm{a}}$.*

a) Suppose that $P \stackrel{\mu}{\longrightarrow} P'$. Then we have:

1. $\mu = a(x)$ *implies* $\langle\!| P |\!\rangle \xrightarrow{a(x)} \gtrsim \langle\!| P' |\!\rangle$;
2. $\mu = \overline{a}b$ *implies* $\langle\!| P |\!\rangle \xrightarrow{\overline{a}(x)} \gtrsim x \to b \mid \langle\!| P' |\!\rangle$, *with* $x \notin \mathrm{fn}(P')$;
3. $\mu = \overline{a}(b)$ *implies* $\langle\!| P |\!\rangle \xrightarrow{\overline{a}(x)} \gtrsim \nu\, b(\, x \to b \mid \langle\!| P' |\!\rangle)$, *with* $x \notin \mathrm{fn}(P')$;
4. $\mu = \tau$ *implies* $\langle\!| P |\!\rangle \xrightarrow{\tau} \gtrsim \langle\!| P' |\!\rangle$.

b) *The converse of part a), i.e.: Suppose that* $\langle\!| P |\!\rangle \xrightarrow{\mu} P_1$. *Then there is* $P' \in \pi_{\mathrm{a}}^{\mathrm{i}}$ *s.t.:*

1. $\mu = a(x)$ *implies* $P \xrightarrow{a(x)} P'$ *with* $P_1 \gtrsim \langle\!| P' |\!\rangle$;
2. $\mu = \overline{a}(x)$ *implies either:*

   2.a) $P \xrightarrow{\overline{a}b} P'$, *with* $x \notin \mathrm{fn}(P')$ *and* $P_1 \gtrsim x \to b \mid \langle\!| P' |\!\rangle$, *or*

   2.b) $P \xrightarrow{\overline{a}(b)} P'$, *with* $x \notin \mathrm{fn}(P')$ *and* $P_1 \gtrsim \nu\, b(\, x \to b \mid \langle\!| P' |\!\rangle)$;
3. $\mu = \tau$ *implies* $P \xrightarrow{\tau} P'$, *with* $P_1 \gtrsim \langle\!| P' |\!\rangle$.

PROOF: Each part of the lemma is proven by transition induction. The only subtle points arise in the proof of parts a)(4) and b)(3), where also Lemma 4.2(2) is used. As an example, we show part a)(4). The only non-trivial case is when the last rule applied for deriving $P \xrightarrow{\tau} P'$ is a communication rule (we suppose for simplicity that the communicated name is free; the case when it is restricted can be easily accommodated):

$$\mathtt{Com}: \frac{P_1 \xrightarrow{\overline{a}b} P_1', P_2 \xrightarrow{a(x)} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1 \mid P_2\{b/x\}} .$$

By induction hypothesis, we have that:

$$\langle\!| P_1 |\!\rangle \xrightarrow{\overline{a}(x)} \gtrsim x \to b \mid \langle\!| P_1' |\!\rangle \text{ with } x \notin \mathrm{fn}(P_1'), \text{ and } \langle\!| P_2 |\!\rangle \xrightarrow{a(x)} \gtrsim \langle\!| P_2' |\!\rangle . \qquad (2)$$

Then we have that:

$$\begin{array}{llll} \langle\!| P_1 \mid P_2 |\!\rangle & \xrightarrow{\tau} \gtrsim & \nu\, x\big(\, x \to b \mid \langle\!| P_1' \mid P_2' |\!\rangle\big) & \text{(from (2) and interaction)} \\ & \gtrsim & \langle\!| (P_1' \mid P_2')\{b/x\} |\!\rangle & \text{(Lemma 4.2(2)).} \\ & = & \langle\!| P_1' \mid P_2'\{b/x\} |\!\rangle & \text{(since } x \notin \mathrm{fn}(P_1')\text{).} \end{array}$$

$\square$

Remark 4.4 Note in the above proof that, since P_2 is a $\pi_{\mathrm{a}}^{\mathrm{i}}$ process, name x does not appear in P_2' in input-subject position: this fact permits applying Lemma 4.2(2). This is the point in the technical development where the "inversion of polarity" property of $\pi_{\mathrm{a}}^{\mathrm{i}}$ turns out to be essential.

The proof of the next tworesults is similar to the corresponding proofs for $\{\!| \,.\, |\!\}$.

Proposition 4.5

a) $P \downarrow a$ *if and only if* $\langle\!| P |\!\rangle \downarrow a$.

b) $P \Longrightarrow P'$ *implies* $\langle\!| P |\!\rangle \Longrightarrow \gtrsim \langle\!| P' |\!\rangle$;

c) $\langle\!| P |\!\rangle \Longrightarrow P_1$ *implies that there is* P' *s.t.* $P \Longrightarrow P'$ *and* $P_1 \gtrsim \langle\!| P' |\!\rangle$.

d) $P \Downarrow a$ *if and only if* $\langle\!| P |\!\rangle \Downarrow a$.

Theorem 4.6 (full abstraction of $\langle\!| \,.\, |\!\rangle$ w.r.t. barbed bisimilarity) *Let* P *and* Q *be any two processes in* $\pi_{\mathrm{a}}^{\mathrm{i}}$. *Then* $P \simeq Q$ *if and only if* $\langle\!| P |\!\rangle \simeq \langle\!| Q |\!\rangle$.

In order to extend the encoding $\langle\!| \,.\, |\!\rangle$ to polyadic $\pi_{\mathrm{a}}^{\mathrm{i}}$, it is enough to replace the input prefix and output prefix clauses of Table 3 with the following two:

$$\langle\!| a(\tilde{x}).P |\!\rangle \stackrel{\mathrm{def}}{=} a(\tilde{x}).\langle\!| P |\!\rangle, \quad \langle\!| \overline{a}\langle\tilde{b}\rangle |\!\rangle \stackrel{\mathrm{def}}{=} \overline{a}(\tilde{x}).\,\tilde{x} \rightarrow \tilde{b}$$

where, for $\tilde{x} = (x_1, \ldots, x_k)$ and $\tilde{b} = (b_1, \ldots, b_k)$, $\tilde{x} \rightarrow \tilde{b}$ stands for $x_1 \rightarrow b_1 \mid \cdots \mid x_k \rightarrow b_k$. Again, the proofs are easily extended. We omit the details.

One might wonder whether our encoding is fully abstract w.r.t. weak bisimilarity. The answer is negative, as shown by the following counter-example.

Counterexample 4.7 (non-full abstraction of $\langle\!| \,.\, |\!\rangle$ w.r.t. weak bisimilarity) Consider the processes in $\pi_{\mathrm{a}}^{\mathrm{i}}$: $P \stackrel{\mathrm{def}}{=} Eq(a,b) \mid \overline{c}a$ and $Q \stackrel{\mathrm{def}}{=} Eq(a,b) \mid \overline{c}b$, where $Eq(a,b) \stackrel{\mathrm{def}}{=} !\, a(x).\overline{b}x \mid !\, b(x).\overline{a}x$ is Honda's *equalizer* [10]. Of course, $P \not\approx Q$, but $\langle\!| P |\!\rangle \approx \langle\!| Q |\!\rangle$. The proof proceeds by first showing that for any z different from a and b, the relation $\mathcal{R}$ defined as:
$\{\, (\langle\!| Eq(a,b) |\!\rangle \mid z \rightarrow b\,, \langle\!| Eq(a,b) |\!\rangle \mid z \rightarrow a\,)\,, (\langle\!| Eq(a,b) |\!\rangle \mid z \rightarrow a\,, \langle\!| Eq(a,b) |\!\rangle \mid z \rightarrow b\,)\,\}$
is a weak bisimulation *up to expansion and up to context* and hence is contained in $\approx$ [4, 19]. From this fact, it easily follows that the relation $\{(\langle\!| P |\!\rangle, \langle\!| Q |\!\rangle), (\langle\!| Q |\!\rangle, \langle\!| P |\!\rangle)\} \cup \approx$ is a weak bisimulation up to context [4], and hence $\langle\!| P |\!\rangle \approx \langle\!| Q |\!\rangle$.

Note that, from an observational point of view, in the absence of matching it is perfectly reasonable to regard the processes P and Q as equivalent, because the equalizer $Eq(a,b)$ makes a and b indistinguishable under any context. Indeed, P and Q are barbed congruent, i.e. they are barbed bisimilar under any context. This leaves open the possibility that $\langle\!| \,.\, |\!\rangle$ be fully abstract for barbed congruence as well: but the proof or disproof of this fact seems to be quite difficult.

Let us define now the encoding $[\![\,.\,]\!]$ from π_{a} to $\pi\mathrm{I}$ as the composition of $\{\!| \,.\, |\!\}$ and $\langle\!| \,.\, |\!\rangle$, thus: $[\![P]\!] \stackrel{\mathrm{def}}{=} \langle\!| \{\!| P |\!\} |\!\rangle$. As an easy consequence of Theorems 3.8 and 4.6, we get the result we were most interested in:

Corollary 4.8 (full abstraction of $[\![\,.\,]\!]$ for barbed bisimilarity) *Let* P *and* Q *be two processes in* π_{a}. *Then* $P \simeq Q$ *if and only if* $[\![P]\!] \simeq [\![Q]\!]$.

5 Conclusions

In this paper, we have provided an encoding from asynchronous π-calculus to $\pi\mathrm{I}$ which is fully abstract on the reductions relations of the two calculi, thus proving that external mobility can be programmed via internal mobility.

For future work, it would be interesting to investigate full abstraction of (variations of) our encoding w.r.t. relations finer than barbed bisimilarity, such as barbed congruence or weak bisimilarity. The existence of these encodings would give evidence that it is possible to *reason* (not only program) on the π-calculus using the simpler theory of πI.

Relationships between πI and the join-calculus should be investigated. Indeed, the join-calculus naturally enjoys the "inversion of polarity" condition on input actions that makes it possible to define a simple encoding from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI. This suggests that a translation of the join-calculus into πI might be even simpler than the translation of π_{a} presented in this work. Also, the first of our encodings suggests that it might be possible to recast, in a traditional name-passing setting, some features of the join-calculus (like the unique-receiver property), by imposing to $\pi_{\mathrm{a}}^{\mathrm{i}}$ natural syntactic limitations and/or typing disciplines, without loosing (much) expressive power. Similar work in this direction is independently being made by R. Amadio [2].

Acknowledgments

I am grateful to Davide Sangiorgi for having pointed to me the research direction pursued in the paper. Three anonymous referees provided valuable comments.

References

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

[2] R. Amadio. A note on objects and localities. Technical report, INRIA-Sophia Antipolis, 1996.

[3] G. Boudol. Asynchrony and the π-calculus (note). Technical Report RR-1702, INRIA-Sophia Antipolis, 1992.

[4] M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. Technical Report ECS-LFCS-94-297, Dept. of Comp. Sci., Edinburgh University, 1994. An extract appeared in *Proc. of STACS'95*, LNCS 900, Springer Verlag.

[5] M. Boreale and D. Sangiorgi. Some congruence properties for π-calculus bisimilarities. Technical Report RR-2870, INRIA-Sophia Antipolis, 1996.

[6] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 5(34):381–392, 1972.

[7] C. Fournet and G. Gonthier. The reflexive CHAM and the join-calculus. 1996. To appear in the *Proc. of POPL'96*.

[8] G. Ferrari, U. Montanari, and P. Quaglia. π-calculus with explicit substitutions. Technical report, Università di Pisa, 1994. Extended abstract appeared in *Proc. of MFCS'94*.

[9] M. Hansen, J. Kleist, and H. Hüttel. Bisimulations for asynchronous mobile processes. In *Proceedings of the Tbilisi Symposium on Language, Logic, and Computation.*, 1995. Research paper HCRC/RP-72, Human Communication Research Centre, University of Edinburgh.

[10] K. Honda. Two bisimilarities for the ν-calculus. Technical Report 92-002, Departement of Computer Science, Keio University, 1992.

[11] R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, LFCS, Dept. of Computer Science, Edinburgh Univ., 1991.

[12] R. Milner. Functions as processes. *Mathematical Structure in Computer Science*, 2(2):119–141, 1992.

[13] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I and II. *Information and Computation*, 100:1 –41 and 42–78, 1992.

[14] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120(2):174–197, 1995.

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

[16] D. Sangiorgi. A theory of bisimulation for the π-calculus. In E. Best, editor, *Proceedings of CONCUR '93, LNCS 715.* Springer-Verlag, Berlin, 1993. To appear in *Acta Informatica.*

[17] D. Sangiorgi. Locality and non-interleaving semantics in calculi for mobile processes. Technical Report ECS–LFCS–94–282, LFCS, Dept. of Computer Science, Edinburgh University, 1994. An extract appeared in *Proc. TACS '94, LNCS*, Springer-Verlag.

[18] D. Sangiorgi. Lazy functions and mobile processes. Technical Report RR-2515, INRIA, 1995.

[19] D. Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. Technical Report RR-2539, INRIA, 1995. Extended Abstract in *Proc. of TAPSOFT'95.*

[20] D. Sangiorgi and R. Milner. The problem of "Weak Bisimulation up-to". In W.R. Cleveland, editor, *Proceedings of CONCUR '92*, volume 630, pages 32–46. Springer Verlag, 1992.