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 πI, a fragment of the π-calculus where only exchange of private names among processes is permitted (internal mobility). The calculus πI enjoys a simpler mathematical treatment, very close to that of CCS. In particular, πI avoids the concept of substitution. We provide an encoding from the asynchronous π-calculus to πI and then prove that two processes are barbed equivalent in π-calculus if and only if their translations in πI cannot be distinguished, under barbed bisimilarity, by any translated static context. 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 [14] is generally recognized as *the* prototypical algebraic language for describing concurrent systems with dynamic communication linkage. Dynamic linkage, also called 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 faithful translations into the language for a variety of computational formalisms, including λ-calculus [13], higher-order process calculi [18] and calculi which permit reasoning on the causal or spatial structure of concurrent systems [5, 20].

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* (or 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:

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

Here $\{\widetilde{b}/\widetilde{c}\}$ denotes the instantiation of names in $\widetilde{c}$ with names in $\widetilde{b}$. Name instantiation is a central concern in the mathematical treatment of certain process equivalences. For instance, 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 [14, 16, 19], and it is not clear which one should be preferred. Name instantiation also complicates the pragmatics of the π-calculus, since

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

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. [9]).

It is therefore natural to seek for fragments of the π-calculus enjoying a simpler treatment of names instantiation, while retaining a non-trivial expressive power. In this paper, we examine the calculus πI, a sub-language of the π-calculus proposed by Sangiorgi in [22]. A prominent feature of πI is that it avoids using name instantiation (other than α-conversion). This fact makes the mathematical treatment and the pragmatics of πI much simpler than in the π-calculus: indeed, the only extra ingredient of πI over CCS is α-conversion of names. We show that the "asynchronous" variant of the π-calculus [11, 4, 21] can be translated, in a natural way, into πI. There is a precise operational correspondence between source process and translated process. A more precise account of our work follows.

The language πI is obtained from the full 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. After the interaction, the communicated names remain private:

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

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 [22].

In πI it is impossible to directly describe *external* mobility, i.e., output of public names (or *free output*), as given by $(*)$. On the other hand, in [22], it has been shown that πI is expressive enough to encode λ-calculus and certain forms of *strictly* higher-order process calculi. However, no one of these formalisms exhibits external mobility. In particular, in strictly higher-order calculi, no name-passing feature is present, since only processes (or abstractions of processes) can be passed around. It is therefore natural to wonder whether external mobility, at least in some limited form, can be expressed via the internal one.

In this paper, we consider as a source language the asynchronous π-calculus, π_{a}, introduced by Honda [11] and Boudol [4]. This is a variant of the π-calculus where: i) the continuation of an output prefix is always the null process (*asynchronous output*), and ii) the *matching* operator, used to test for equalities between names, and the *non-deterministic summation* operator [14] are ruled out. It is meaningful to restrict ourselves to asynchronous output prefix because it has been shown that the full output prefix can be, in a reasonable sense, encoded in π_{a} [4]. The same holds for a form of summation that only uses input guards [15]. On the other hand, matching seems to play a secondary rôle, as far as expressiveness is concerned (e.g. it is not needed to encode λ-calculus, higher-order calculi etc.). The way the presence of matching would affect our results is discussed at the end of Section 4.

We define an encoding, $[\![\,.\,]\!]$, from π_{a} to πI. The basic idea is that the output of a public name b at a, $\overline{a}b$, is replaced, in πI, by the output of a private name x, which acts as a reference 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, as this would require free output, but are, in turn, linked to them (the definition of link processes will indeed be recursive). Since a link $x \rightarrow b$ transforms outputs at x into outputs at b, a process owning x can trigger an output at b by interacting with $x \rightarrow b$. This approach is somewhat reminiscent of Sangiorgi's factorization theorem for higher-order processes, saying that the output of a process can be replaced by the output of a trigger with a private link to a replicator of the process [18].

Link processes can be used to naturally encode those π_{a}-processes in which each receiver, say $a(x).P$, can only use x in P as an output channel. The subset of π_{a} obeying this "inversion of polarity" syntactical condition will be called $\pi_{\mathrm{a}}^{\mathrm{i}}$. Then, the encoding $[\![\,.\,]\!]$ will be obtained as the composition of two simple translations: one ($\{\!|\,.\,|\!\}$) from π_{a} to the intermediate calculus $\pi_{\mathrm{a}}^{\mathrm{i}}$, and the other ($(\!|\,.\,|\!)$) from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to $\pi\mathrm{I}$. Each of these two encodings is proven to establish a close operational correspondence between source terms and translated terms, as explained below.

To study the operational correspondence established by the encoding, we consider a variety of semantic equivalences based on *barbed bisimilarity* [18]. The latter is a uniform notion of equivalence among processes, whose definition only relies on the existence of a reduction relation on processes ($P \stackrel{\tau}{\longrightarrow} P'$) and of an observation predicate on actions ($P \downarrow a$). Starting from barbed bisimilarity, more refined notions of equivalences can be obtained, still in a uniform fashion. In fact, given a family $\mathcal{F}$ of language contexts, we can declare two processes $\mathcal{F}$-equivalent if they are barbed bisimilar whenever plugged in any context of $\mathcal{F}$. One of the most significant relations obtained in this way is *barbed equivalence* [18], where *static* contexts, of the form $\nu\,\widetilde{b}\,(R \mid [\cdot])$, are used. This conveys an idea of "contexts as observers" (similar to that found in the testing scenario of De Nicola and Hennessy [7]). For our encoding, we prove that two processes in π_{a} are barbed equivalent if and only if their translations in $\pi\mathrm{I}$ cannot be distinguished using translations of static contexts. This result precisely says that π_{a} can be faithfully compiled in $\pi\mathrm{I}$. An important consequence of this fact is a soundness theorem, saying that whenever two translated terms are barbed equivalent in $\pi\mathrm{I}$ then the corresponding source terms are barbed equivalent in π_{a}. These results strengthen the claim of [22]: 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. Thus $\pi\mathrm{I}$ seems to be a model of computation more basic than the π-calculus.

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 b as an input channel. 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 seems to suggest that, without losing much expressive power, it should be possible to further refine the channel discipline of $\pi_{\mathrm{a}}^{\mathrm{i}}$ in such a way that 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 [8]. But to fully explore this connection is outside the scope of the present paper.

The rest of the paper is organized as follows. Section 2 contains some background material on π_{a}, $\pi\mathrm{I}$ and on the behavioural relations used throughout the paper. Section 3 presents the encoding from π_{a} to $\pi_{\mathrm{a}}^{\mathrm{i}}$. Section 4 presents the encoding from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to $\pi\mathrm{I}$ and some remarks about the matching operator and observation equivalence semantics. The paper ends with a few conclusive considerations in Section 5.

2 Background

In this section we introduce the languages π_{a}, $\pi_{\mathrm{a}}^{\mathrm{i}}$ and $\pi\mathrm{I}$[1], their operational semantics and some behavioural relations on them.

[1] Actually, we will introduce the summation-free fragment of $\pi\mathrm{I}$, which is enough for our purposes.

2.1 The languages π_{a}, $\pi_{\mathrm{a}}^{\mathrm{i}}$ and $\pi\mathrm{I}$

The name-passing languages π_{a}, $\pi_{\mathrm{a}}^{\mathrm{i}}$ and $\pi\mathrm{I}$ 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, by constraining the output constructs, we shall isolate out of it the fragments of our interest.

The countable set $\mathcal{N}$ of *names* is ranged over by $a, b, \ldots, x, y, \ldots$. The set of names and co-names, $\mathcal{N} \cup \{\overline{a} \,|\, a \in \mathcal{N}\}$, is ranged over by p. A countable set of *agent constants*, each having a non-negative integer arity, is ranged over by A. *Processes* are ranged over by P, Q and R. The subset of the π-calculus syntax we shall consider is built out of the operators of action prefix, restriction, parallel composition, replication and agent constant:

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

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 and are ranged over by α; 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. We abbreviate $\alpha.\,\mathbf{0}$ to α and $\nu\, a\, \nu\, b\, P$ to $(\nu\, a, b\,)P$.

Following [22], we have explicitly introduced the bound output prefix $\overline{a}(\widetilde{b}).P$: in the full π-calculus, this would only be syntactic sugar for $\nu\, \widetilde{b}\, (\overline{a}\langle\widetilde{b}\rangle.P)$, when $a \notin \widetilde{b}$.

Input prefix $a(\widetilde{b})$. and restriction $\nu\, a$ act as *binders* for names $\widetilde{b}$ and a, respectively. *Free names* and *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 applying σ to E, while $E\sigma\sigma'$ stands for $(E\sigma)\sigma'$ and $\{\widetilde{b}/\widetilde{c}\}$ stands for the substitution which maps $\widetilde{c}$ onto $\widetilde{b}$ and is the identity elsewhere. We assume the following decreasing order of precedence when writing process expressions: substitution, prefix, replication, restriction, parallel composition.

Each agent constant 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), $\nu\, \widetilde{b}'\, \overline{a}\langle\widetilde{b}\rangle$ (output) or $\overline{a}(\widetilde{b})$ (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 shall work up to α-conversion on names so to avoid tedious side conditions in transition rules and bisimulation clauses. Therefore:

- in processes, bound names are assumed to be different from each other and from the free names;
- α-equivalent processes are identified. In particular, they are assumed to have the same transitions;
- substitutions do not touch bound names.

Following Milner [12], we only admit *well-sorted processes*: the sorting is necessary to prevent arity mismatching in communications, like in $\overline{a}\langle b, c\rangle.\, P \,|\, a(x).\, Q$. 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 content of this paper.

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

$$\texttt{Act}: \ \alpha.P \xrightarrow{\alpha} P \qquad \texttt{Rep}: \ \frac{P \mid !P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P'}$$

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

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

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

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

- $\mathcal{P}$ the above defined set of π-calculus processes;
- π_{a} the subset of $\mathcal{P}$ with no bound output prefixes and no agent constants;
- $\pi_{\mathrm{a}}^{\mathrm{i}}$ 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.

Each of the subsets π_{a}, $\pi_{\mathrm{a}}^{\mathrm{i}}$ and $\pi\mathrm{I}$ is easily seen to be closed w.r.t. the transition relation $\xrightarrow{\mu}$. Note that the language $\pi\mathrm{I}$ contains both replication and agent constants: contrary to what happens in the π-calculus, these two primitives are not equivalent in $\pi\mathrm{I}$ (see [22]). Even though replication can be defined in terms of agent constants, we decided to keep it for notational convenience.

2.2 Behavioural relations on processes

Weak barbed bisimilarity and the equivalences based on it [18] are the relations we are most interested in. In some of the proofs we will, however, use 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 $\stackrel{\mu}{\Longrightarrow}$ be $\Longrightarrow\xrightarrow{\mu}\Longrightarrow$, and let $P \stackrel{\widehat{\mu}}{\Longrightarrow} Q$ be $P \stackrel{\mu}{\Longrightarrow} Q$, if $\mu \neq \tau$, and $P \Longrightarrow Q$, if $\mu = \tau$.

2.2.1 Barbed bisimilarity and equivalence

Barbed bisimulation [18] represents a uniform mechanism for defining meaningful behavioural equivalences, which relies on two concepts common to different process calculi: the *reduction relation* $\xrightarrow{\tau}$ and an *observation predicate* $\downarrow p$. In π-calculus, we say that P *commits* to a (resp. to $\overline{a}$), and write $P \downarrow a$ (resp. $P \downarrow \overline{a}$), if P contains a prefix $a(\widetilde{b})$, (resp. $\overline{a}\langle\widetilde{b}\rangle$ 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 $P \Downarrow p$ if P is capable of interacting on p possibly after a few invisible steps, i.e. if $P \Longrightarrow \downarrow p$.

Definition 2.1 (weak barbed bisimilarity) *A symmetric binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ 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. *$P \downarrow p$ implies $Q \Downarrow p$, for any p.*

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

Weak barbed bisimilarity is a rather coarse relation. For example, the processes $P \stackrel{\text{def}}{=} a$ and $Q \stackrel{\text{def}}{=} a.b$ are barbed bisimilar. A way of getting a finer relation is that of "closing" $\dot{\simeq}$ under a family of language contexts. Indeed, P and Q above are distinguished as soon as they are plugged in the context $\overline{a} \mid [\cdot]$. By closing barbed bisimilarity under all *static* contexts, one obtains barbed equivalence. Static contexts are of the form $(\nu\, \widetilde{b})(R \mid [\cdot])$, for any $\widetilde{b}$ and R.

Definition 2.2 (barbed equivalence) *For each language $\mathcal{L} \in \{ \pi_{\mathrm{a}},\ \pi_{\mathrm{a}}^{\mathrm{i}}, \pi\mathrm{I}\}$, we define* barbed equivalence on $\mathcal{L}$, $\simeq_{\mathcal{L}}$, *as follows: $P \simeq_{\mathcal{L}} Q$ if and only if for each static context $(\nu\, \widetilde{b})(R \mid [\cdot])$ of $\mathcal{L}$, it holds that $(\nu\, \widetilde{b})(R \mid P) \dot{\simeq} (\nu\, \widetilde{b})(R \mid Q)$.*

From the above definition, it is clear that barbed equivalence is preserved by restriction and parallel composition operators. Indeed, it is easy to prove that $\simeq_{\pi\mathrm{I}}$ is a congruence, and we strongly conjecture that the same holds for $\simeq_{\pi_{\mathrm{a}}}$ and $\simeq_{\pi_{\mathrm{a}}^{\mathrm{i}}}$. In the sequel, we shall omit the subscript $\mathcal{L}$ when it is clear which language we are referring to.

2.2.2 Standard bisimilarities

Several forms of bisimulation-based equivalences have been proposed for the π-calculus, notably the *late, early* and *open* bisimilarities [14, 16, 19]: the difference among these depends 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*[2] bisimilarity (see also [10, 11, 21, 6]). In the latter, no name instantiation of the input formal parameter is required when matching input actions, apart from α-conversion. We recall its definition below.

Definition 2.3 (strong bisimilarity) *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a* strong ground bisimulation *if whenever $P \mathcal{R} Q$ and $P \xrightarrow{\mu} P'$[3] then 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 ground bisimulation $\mathcal{R}$.*

Some elementary laws for strong bisimulation:

Proposition 2.4

1. *$\nu\, x\, (P \mid Q) \sim (\nu\, x\, P) \mid Q$, if $x \notin \mathrm{fn}(Q)$;*
2. *$!\, a(\widetilde{y}).P \sim a(\widetilde{y}).(P \mid\, !\, a(\widetilde{y}).P)$, if $a \notin \widetilde{y}$;*

[2] In [14], *ground* bisimilarity indicates a different equivalence.

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

3. $\nu\,\widetilde{z}\,\big(\alpha.P \mid \Pi_{i\in I} a_i(\widetilde{y}_i).Q_i\big) \sim \alpha.\nu\,\widetilde{z}\,\big(P \mid \Pi_{i\in I} a_i(\widetilde{y}_i).Q_i\big)$, *if* $\alpha = a(\widetilde{y})$ *or* $\alpha = \overline{a}\langle\widetilde{y}\rangle$ *for some* $\widetilde{y}$, *and* $a \notin \widetilde{z}$ *and* $\widetilde{y} \cap \widetilde{z} = \emptyset$ *and* $\{a_i \mid i \in I\} \subseteq \widetilde{z}$.

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 2.3 the transition $Q \stackrel{\mu}{\longrightarrow} Q'$ with $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$. We use $\approx$ for weak ground bisimilarity.

Since we are only interested in the $\pi_{\rm a}$ and πI fragments of $\mathcal{P}$, where all the mentioned forms of standard bisimilarity coincide, in the sequel we shall drop the adjective 'ground' when referring to $\sim$ and $\approx$. Both $\sim$ and $\approx$ are congruences on the fragments of the π-calculus of our interest [10, 11, 21, 6].

2.2.3 Expansion preorder

The expansion relation $\lesssim$ [2, 23] is an asymmetric variant of $\approx$ which allows one to "count" the number of τ-actions performed by the processes. Thus, $P \lesssim Q$ holds if $P \approx Q$, and 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, $\pi_{\rm a}$ and πI (the proof parallels that given in [10, 11, 21, 6] for standard bisimilarities). We give below the definition of ground expansion preorder, omitting the adjective 'ground'.

Definition 2.5 (expansion preorder) *A relation* $\mathcal{R} \subseteq \mathcal{P}\times\mathcal{P}$ *is an* expansion *if* $P\,\mathcal{R}\,Q$ *implies:*

1. *Whenever* $P \stackrel{\mu}{\longrightarrow} P'$, *there exists* Q' *s.t.* $Q \stackrel{\mu}{\Longrightarrow} Q'$ *and* $P'\,\mathcal{R}\,Q'$;
2. *whenever* $Q \stackrel{\mu}{\longrightarrow} Q'$, *there exists* P' *s.t.* $P \stackrel{\widehat{\mu}}{\longrightarrow} P'$ *and* $P'\,\mathcal{R}\,Q'$.

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

For instance, $P \lesssim \tau.P$, but $\tau.P \not\lesssim P$ since $\tau.P$ has to perform more τ-actions than P in order to mimic its actions. The relation $\lesssim$ is indeed a preorder. 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 useful relationships between them:

Proposition 2.6

a) *In* $\pi_{\rm a}$ *and* πI, *the relations* $\sim$, $\approx$ *and* $\lesssim$ *are preserved by all operators and by name instantiation.*

b) *In* $\pi_{\rm a}$, $\pi_{\rm a}^{\rm i}$ *and* πI, *the following is an increasing chain of relations w.r.t. inclusion:* $\sim$, $\lesssim$, $\approx$, $\simeq$, $\dot{\simeq}$.

Proof: Part a) is standard. For b), the inclusion $\approx \subseteq \simeq$ is easily derived from the congruence properties of $\approx$ and from the definition of $\simeq$. All the remaining inclusions are straightforward. □

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

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

$$\{\!|\overline{a}b|\!\}\rho \stackrel{\text{def}}{=} \nu\, z\left(\overline{a}\langle b,z\rangle \mid z \hookrightarrow b\right) \text{ with } z \text{ fresh} \qquad \{\!|\nu\, x\, P|\!\}\rho \stackrel{\text{def}}{=} \nu\, x\, \{\!|P|\!\}\rho$$

$$\{\!|P \mid Q|\!\}\rho \stackrel{\text{def}}{=} \{\!|P|\!\}\rho \mid \{\!|Q|\!\}\rho \qquad \{\!|\,!\,P|\!\}\rho \stackrel{\text{def}}{=} \,!\,\{\!|P|\!\}\rho$$

$$\text{Define: } \{\!|P|\!\} \stackrel{\text{def}}{=} \{\!|P|\!\}\emptyset\ .$$

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

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

Let us illustrate informally the way 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 receives at z 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 output at z that activate $z \hookrightarrow b$.

For notational simplicity, below we will first present the encoding for the monadic fragment of π_{a}. Hence from now on in the section and until otherwise stated, π_{a} refers to the monadic fragment. The polyadic case will be accommodated later. 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:*

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

The encoding $\{\!|\,.\,|\!\}$ from (monadic) π_{a} to (polyadic) $\pi_{\mathrm{a}}^{\mathrm{i}}$ is defined in Table 2. The definition makes use of an auxiliary parameter, ρ, which is an injective finite partial function from $\mathcal{N}$ to $\mathcal{N}$. Function ρ 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 $\rho[y/x]$ denotes the partial function which yields y on x and behaves like ρ elsewhere. Furthermore, $\mathrm{ran}(\rho)$ denotes the set $\{y : \rho(x) = y, \text{ for some } x\}$. 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.

Example 3.2 Consider the π_{a}-process $P \stackrel{\text{def}}{=} a(x).x(y).\overline{y}v \mid \overline{a}b \mid \overline{b}c$ and its reductions:

$$P \stackrel{\tau}{\longrightarrow} b(y).\overline{y}v \mid \overline{b}c \stackrel{\tau}{\longrightarrow} \overline{c}v\ .$$

Let us see how these reductions are mimicked by $\{\!|P|\!\}$:

$$
\begin{array}{rcl}
\{\!|P|\!\} & = & \text{(definition of } \{\!|\,.\,|\!\}) \\
a(x,z).\nu\, h\left(\overline{z}h \mid h(y,y').\{\!|\overline{y}v|\!\}[y'/y]\right) \mid \nu\, z\left(\overline{a}\langle b,z\rangle \mid\ z \hookrightarrow b\right) \mid \{\!|\overline{b}c|\!\} & \xrightarrow{\tau}\sim & \text{(interaction at } a\text{, laws for } \nu\, x\,) \\
\nu\, z\left(\nu\, h\,(\overline{z}h \mid h(y,y').\{\!|\overline{y}v|\!\}[y'/y]) \mid\ z \hookrightarrow b\right) \mid \{\!|\overline{b}c|\!\} & \xrightarrow{\tau}\sim & \text{(interaction at } z\text{, laws for } \nu\, z\,) \\
\nu\, h\left(h(y,y').\{\!|\overline{y}v|\!\}[y'/y] \mid b(t,w).\overline{h}\langle t,w\rangle\right) \mid \nu\, w\left(\overline{b}\langle c,w\rangle \mid\ w \hookrightarrow c\right) & \xrightarrow{\tau} & \text{(interaction at } b\text{)} \\
(\nu\, w,h)\big(h(y,y').\{\!|\overline{y}v|\!\}[y'/y] \mid \overline{h}\langle c,w\rangle \mid\ w \hookrightarrow c\big) & \xrightarrow{\tau}\sim & \text{(interaction at } h\text{, laws for } \nu\, h\,) \\
\nu\, w\left(\{\!|\overline{y}v|\!\}[y'/y]\{c,w/y,y'\} \mid\ w \hookrightarrow c\right) & \sim & \text{(laws for } \nu\, w \text{ and def. of } \{\!|\,.\,|\!\}) \\
\{\!|\overline{c}v|\!\}\,. & &
\end{array}
$$

Before proving our main result about $\{\!|\,.\,|\!\}$ we need to fix a few basic properties of this mapping. First, a technical lemma. Part 1 and 2 state well-known distributivity laws for $!$, due to Milner [12], while part 3 and 4 are distributivity properties for input managers.

Lemma 3.3 *Let P, P_1 and P_2 be processes in π_{a} such that z may occur free in P, P_1 and P_2 only in output-subject position.*

1. $\nu\, z\left(!\, z(x).P \mid P_1 \mid P_2\right) \sim \nu\, z\,(!\, z(x).P \mid P_1) \mid \nu\, z\,(!\, z(x).P \mid P_2)$.
2. $\nu\, z\left(!\, z(x).P \mid\ !P_1\right) \sim !\,\nu\, z\,(!\, z(x).P \mid P_1)$.
3. $\nu\, z\left(\ z \hookrightarrow b \mid P_1 \mid P_2\right) \sim \nu\, z\left(\ z \hookrightarrow b \mid P_1\right) \mid \nu\, z\left(\ z \hookrightarrow b \mid P_2\right)$, *where* $z \neq b$.
4. $\nu\, z\left(\ z \hookrightarrow b \mid\ !P\right) \sim !\,\nu\, z\left(\ z \hookrightarrow b \mid P\right)$, *where* $z \neq b$.

Proof: Part 1 and 2 are shown by exhibiting the appropriate bisimulation relations (see e.g. [12]). Part 2 and 3 are direct consequences of part 1 and 2, respectively. □

In the next lemma, part 1 states a simple syntactical property of the encoding, while part 2 asserts that an input manager $z \hookrightarrow b$, if z is hidden, somehow acts like a substitution of z with b, but just for those names appearing in input position (therefore, when combined with an appropriate substitution for names in output position, $z \hookrightarrow b$ has just the effect of a full substitution).

Lemma 3.4 *Let P be a π_{a}-process, let ρ be undefined on a and b, let $\mathrm{fn}(P) \cap \mathrm{ran}(\rho) = \emptyset$. Let z be fresh.*

1. *z may appear free in $\{\!|P|\!\}\rho[z/a]$ only in output-subject position.*
2. $\nu\, z\left(\ z \hookrightarrow b \mid \{\!|P|\!\}\rho[z/a]\right)\{b/a\} \gtrsim \{\!|P\{b/a\}|\!\}\rho$.

Proof: Part 1 is proven by a straightforward induction on P. Part 2 is proven by induction on P. The output case is trivial, since the parameter $\rho[z/a]$ is simply ignored. For the inductive step cases, those different from input prefix easily follow from the induction hypothesis. In particular, in the parallel composition and in the replication cases, we first distribute the input manager $z \hookrightarrow b$ over subterms (applying part 1 of this lemma and part 3 or 4 of Lemma 3.3), and then use induction hypothesis.

The most interesting case concerns input prefix, when $P = c(x).P'$, for some c and P'. We distinguish two situations, namely $c \neq a$ and $c = a$ (Below, we shall implicitly use Proposition 2.6).

$c \neq a$. According to the definition in Table 2, we distinguish two possibilities: ρ is undefined on c and $\rho(c) = w$, for some w. We only deal with the latter, which is a bit more difficult. By definition,

$\{\!|P|\!\}\rho[z/a] = \nu\, h\,(\overline{w}h \mid h(x,y).R)$, where $R \stackrel{\text{def}}{=} \{\!|P'|\!\}\rho[z/a][y/x]$. Rearranging subterms, we can write:

$$\nu\, z\,\big(\, z \hookrightarrow b \mid \{\!|P|\!\}\rho[z/a]\big)\{b/a\} \sim \nu\, h\,\big(\overline{w}h \mid \nu\, z\,(\, z \hookrightarrow b \mid h(x,y).R)\{b/a\}\big). \qquad (1)$$

By Proposition 2.4(2)-(3), we get:

$$\nu\, z\,(\, z \hookrightarrow b \mid h(x,y).R) \sim h(x,y).\nu\, z\,(\, z \hookrightarrow b \mid R)$$

which, when replaced in the RHS of (1), yields

$$\nu\, z\,\big(\, z \hookrightarrow b \mid \{\!|P|\!\}\rho[z/b]\big)\{b/a\} \sim \nu\, h\,\big(\overline{w}h \mid h(x,y).\nu\, z\,(\, z \hookrightarrow b \mid R)\{b/a\}\big).$$

This, by definition of R, the induction hypothesis and definition of encoding, easily implies the thesis for this case.

$c = a$. We have:

$$\begin{aligned}
&\nu\, z\,\big(\, z \hookrightarrow b \mid \{\!|P|\!\}\rho[z/a]\big)\{b/a\} && = \\
&\qquad \text{(definition of } \{\!|P|\!\}\rho[z/a]) \\
&\nu\, z\,\big(\, z \hookrightarrow b \mid \nu\, h\,\big(\overline{z}\langle h\rangle \mid h(x,y).\{\!|P'|\!\}\rho[z/a][y/x]\big)\big)\{b/a\} && \sim \\
&\qquad \text{(moving } \nu\, h \text{ to the outermost position, in virtue of Prop. 2.4(1))} \\
&\nu\, h\,\nu\, z\,\big(\, z \hookrightarrow b \mid \overline{z}\langle h\rangle \mid h(x,y).\{\!|P'|\!\}\rho[z/a][y/x]\big)\{b/a\} && \sim \\
&\qquad \text{(distributing } z \hookrightarrow b \text{, in virtue of part 1 and of Lemma 3.3(3))} \\
&\nu\, h\,\big(\nu\, z\,(\, z \hookrightarrow b \mid \overline{z}\langle h\rangle) \mid \nu\, z\,(\, z \hookrightarrow b \mid h(x,y).\{\!|P'|\!\}\rho[z/a][y/x])\big)\{b/a\} && \gtrsim \\
&\qquad \text{(given that } \nu\, z\,(\, z \hookrightarrow b \mid \overline{z}\langle h\rangle) \gtrsim b(x,y).\overline{h}\langle x,y\rangle) \\
&\nu\, h\,\big(b(x,y).\overline{h}\langle x,y\rangle \mid \nu\, z\,(\, z \hookrightarrow b \mid h(x,y).\{\!|P'|\!\}\rho[z/a][y/x])\big)\{b/a\} && \sim \\
&\qquad \text{(Proposition 2.4(2)-(3))} \\
&b(x,y).\nu\, h\,\big(\overline{h}\langle x,y\rangle \mid h(x,y).\nu\, z\,(\, z \hookrightarrow b \mid \{\!|P'|\!\}\rho[z/a][y/x])\{b/a\}\big) && \gtrsim \\
&\qquad \text{(induction hypothesis)} \\
&b(x,y).\nu\, h\,\big(\overline{h}\langle x,y\rangle \mid h(x,y).\{\!|P'\{b/a\}|\!\}\rho[y/x]\big) && \gtrsim \\
&\qquad \text{(a simple law for } \gtrsim\text{)} \\
&b(x,y).\{\!|P'\{b/a\}|\!\}\rho[y/x] && = \\
&\qquad \text{(by definition of } \{\!|\,.\,|\!\} \text{ and the fact that } \rho \text{ is undefined on } b). \\
&\qquad \{\!|P\{b/a\}|\!\}\rho\,.
\end{aligned}$$

□

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

Proposition 3.5 (correspondence on transitions) *Let P be a π_{a}-process.*

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

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

b) Suppose that $\{\!|P|\!\} \xrightarrow{\mu} P_1$. *Then there are* $P' \in \pi_{\mathrm{a}}$ *and* a *such that one of the followings holds:*

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

Proof: Parts 1, 2 and 3 of a) and b) are proven by straightforward transition induction. The only subtle points arise in the proof of parts a)(4) and b)(4), where also Lemma 3.4(2) is used. We show part a)(4), since b)(4) is handled similarly. The proof proceeds by transition induction on $P \xrightarrow{\tau} P'$.

The cases different from `Com` easily follow by induction hypothesis. The only non-trivial case is when the last rule applied is `Com` (we suppose for simplicity that the communicated name is free; the case when it is restricted can be easily accommodated):

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

where we suppose, as usual, that x is fresh. By applying parts 1 and 2 of this lemma to the transitions in the premise of the rule, we have that:

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

Note now, that, for any Q, if $x \notin \mathrm{fn}(Q)$ then $\{\!|Q|\!\}[y/x] = \{\!|Q|\!\}$. This fact is exploited in the second line below. We have:

$$\begin{array}{lcll} \{\!|P_1 \mid P_2|\!\} \xrightarrow{\tau} & \gtrsim & \nu\, y\, \big(\, y \hookrightarrow b \mid \{\!|P_1'|\!\} \mid \{\!|P_2'|\!\}[y/x]\{b/x\}\big) & \text{(from (2) and } \texttt{Com}) \\ & = & \nu\, y\, \big(\, y \hookrightarrow b \mid \{\!|P_1' \mid P_2'|\!\}[y/x]\big)\{b/x\} & (x \notin \mathrm{fn}(P_1') \text{ and def. of } \{\!|\,.\,|\!\}) \\ & \gtrsim & \{\!|P_1' \mid P_2'\{b/x\}|\!\}\rho & \text{(Lemma 3.4(2) and } x \notin \mathrm{fn}(P_1')) \end{array}$$

□

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

Proposition 3.6

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

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

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 3.5, 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 $\pi_{\mathrm{a}}^{\mathrm{i}}$,

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

Now, by Proposition 3.5(b)(4), there is P' in π_{a} s.t.

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

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

Remark 3.7 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 $\{\!|P''|\!\} \xrightarrow{\widehat{\tau}} R$); as a consequence, we could not have applied Proposition 3.5(b)(4) to close up the induction. □

A simple proof technique for barbed bisimilarity:

Definition 3.8 *A symmetric binary relation* $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *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) $P \downarrow p$ *implies* $Q \Downarrow p$.

Lemma 3.9 *If* $\mathcal{R}$ *is a barbed bisimulation up to expansion then* $\mathcal{R} \subseteq \dot{\simeq}$.

Proof: Show that the relation

$$\mathcal{S} = \{(P, Q) \,:\, P \gtrsim \mathcal{R} \lesssim Q\}$$

is a barbed bisimulation. Straightforward. □

We arrive at the main theorem of the section:

Theorem 3.10 *Let* P *and* Q *be processes in* π_{a}. *Then* $P \dot{\simeq} Q$ *if and only if* $\{\!|P|\!\} \dot{\simeq} \{\!|Q|\!\}$.

PROOF: For the 'if' part, exploiting the above Proposition 3.6 it is easy to show that the relation:

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

is a barbed bisimulation up to expansion in $\pi_{\mathrm{a}}^{\mathrm{i}}$.

For the 'only if' part, again exploiting the above Proposition 3.6 it is easy to see that the relation:

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

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

As already explained in the introduction, the behavioural equivalence we consider on $\pi_{\mathrm{a}}^{\mathrm{i}}$ is the closure of barbed bisimilarity under the family of *translated* static contexts. Encoding $\{\!|\,.\,|\!\}$ is applied to static contexts in the expected way: given a static context $C[\cdot] = (\nu\,\widetilde{b})(R \mid [\cdot])$, $\{\!|C|\!\}[\cdot]$ is $(\nu\,\widetilde{b})(\{\!|R|\!\} \mid [\cdot])$.

Definition 3.11 ($\simeq_{\mathrm{tr1}}$ equivalence) *Let P and Q be in $\pi_{\mathrm{a}}^{\mathrm{i}}$. We define $P \simeq_{\mathrm{tr1}} Q$ if and only if for each static context $C[\cdot]$ in π_{a}, it holds that $\{\!|C|\!\}[P] \dot{\simeq} \{\!|C|\!\}[Q]$.*

An easy consequence of the previous theorem is the following corollary, where part 2 states a relevant soundness property for $\{\!|\,\cdot\,|\!\}$.

Corollary 3.12 *Let P and Q be two processes in π_{a}.*

1. *$P \simeq Q$ if and only if $\{\!|P|\!\} \simeq_{\mathrm{tr1}} \{\!|Q|\!\}$;*
2. *$\{\!|P|\!\} \simeq \{\!|Q|\!\}$ implies $P \simeq Q$.*

PROOF: Part 1 is an easy consequence of Theorem 3.10, of the congruence properties of $\simeq$ and of the compositionality of $\{\!|\,.\,|\!\}$ for restriction and parallel composition. Part 2 follows from part 1, given that $\simeq \subseteq \simeq_{\mathrm{tr1}}$. □

Note that Corollary 4.9(2) and the fact that in $\pi_{\mathrm{a}}^{\mathrm{i}}$ $\approx \subseteq \simeq$ (Proposition 2.6) imply that we can use $\approx$ on $\pi_{\mathrm{a}}^{\mathrm{i}}$ to reason about $\simeq$ on π_{a}: in other words, $\{\!|P|\!\} \approx \{\!|Q|\!\}$ implies $P \simeq Q$.

3.1 The polyadic case

We indicate now the modifications necessary to extend $\{\!|\,.\,|\!\}$ to the full polyadic π_{a}. We use the following notations: for $\widetilde{u} = (u_1, \ldots, u_k)$ and $\widetilde{v} = (v_1, \ldots, v_k)$, $[\widetilde{u}/\widetilde{v}]$ stands for $[u_1/v_1] \cdots [u_k/v_k]$ and $\widetilde{u} \hookrightarrow \widetilde{v}$ stands for $u_1 \hookrightarrow v_1 \mid \cdots \mid u_k \hookrightarrow v_k$. Recall that each name has an associated arity, which is the number of parameters it carries. Definition 3.1 of input manager process and the clauses for input and output prefixes of Table 2 are replaced by the following:

$$z \hookrightarrow b \quad \stackrel{\mathrm{def}}{=} \quad !\,z(h).b(\widetilde{x}, \widetilde{y}).\overline{h}\langle \widetilde{x}, \widetilde{y}\rangle \qquad \text{with } \widetilde{x},\ \widetilde{y} \text{ fresh}$$

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

$$\{\!|\overline{a}\langle \widetilde{b}\rangle|\!\}\rho \quad \stackrel{\mathrm{def}}{=} \quad (\nu\,\widetilde{z})\big(\overline{a}\langle \widetilde{b}, \widetilde{z}\rangle \mid \widetilde{z} \hookrightarrow \widetilde{b}\,\big) \qquad \widetilde{z} \text{ fresh.}$$

The requirements on the number of the (distinct) components of $\widetilde{x}$, $\widetilde{y}$ and $\widetilde{z}$ are obvious. In particular, in the definition of input manager, the number of components of $\widetilde{x}$ and of $\widetilde{y}$ must equal the arity of name b in the original sorting. The proofs carry over with some straightforward notational changes. We omit the details.

$(\!|P|\!)$ is defined as:

$$\begin{array}{rclrcl}
(\!|a(x).P|\!) & \stackrel{\text{def}}{=} & a(x).(\!|P|\!) & (\!|\overline{a}b|\!) & \stackrel{\text{def}}{=} & \overline{a}(x).\, x \to b\,,\ x \text{ fresh} \\
(\!|P \mid Q|\!) & \stackrel{\text{def}}{=} & (\!|P|\!) \mid (\!|Q|\!) & (\!|\nu\, x\, P|\!) & \stackrel{\text{def}}{=} & \nu\, x\, (\!|P|\!) \\
\multicolumn{6}{c}{(\!|\,!\,P|\!) \stackrel{\text{def}}{=} !\,(\!|P|\!)\,.}
\end{array}$$

Table 3: Definition of the encoding $(\!|\,.\,|\!)$, from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI.

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 that points to 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 [22], where they have been used to encode the lazy λ-calculus into πI.

Definition 4.1 (link processes, [22]) *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 $(\!|\,.\,|\!)$ from $\pi_{\mathrm{a}}^{\mathrm{i}}$ to πI is defined in Table 3. Again, we present the encoding for the monadic fragment of $\pi_{\mathrm{a}}^{\mathrm{i}}$. From now on, until otherwise stated, we will work in monadic $\pi_{\mathrm{a}}^{\mathrm{i}}$. The polyadic calculus will be accommodated later.

Example 4.2 Consider the process in $\pi_{\mathrm{a}}^{\mathrm{i}}$ $P \stackrel{\text{def}}{=} a(x).\overline{x}c \mid \overline{a}b \mid b(y).\overline{y}v$ and its reductions:

$$P \ \stackrel{\tau}{\longrightarrow}\ \overline{b}c \mid b(y).\overline{y}v \ \stackrel{\tau}{\longrightarrow}\ \overline{c}v\,.$$

Let us see how these reductions are mimicked by $(\!|P|\!)$, using properties of link processes which are explained below and established in Lemma 4.3:

$$\begin{array}{rcl}
(\!|P|\!) & = & \text{(definition of } (\!|\,.\,|\!)) \\
a(x).(\!|\overline{x}c|\!) \mid \overline{a}(x).\, x \to b \mid b(y).(\!|\overline{y}v|\!) & \stackrel{\tau}{\longrightarrow} & \text{(interaction at } a \text{ and def. of } (\!|\overline{x}c|\!)) \\
\nu\, x\, \big(\overline{x}(w).\, w \to c \mid x \to b\big) \mid b(y).(\!|\overline{y}v|\!) & \stackrel{\tau}{\longrightarrow} \sim & \text{(def. of } x \to b\text{, interaction at } x\text{, laws for } \nu\, x\,) \\
\nu\, w\, \big(w \to c \mid \overline{b}(y).\, y \to w\big) \mid b(y).(\!|\overline{y}v|\!) & \stackrel{\tau}{\longrightarrow} \sim & \text{(interaction at } b) \\
(\nu\, yw)\big(w \to c \mid y \to w \mid (\!|\overline{y}v|\!)\big) & \sim & \text{(laws for } \nu\, w \text{ and } \mid) \\
\nu\, y\, \big(\nu\, w\, (y \to w \mid w \to c) \mid (\!|\overline{y}v|\!)\big) & \gtrsim & \\
\nu\, y\, \big(y \to c \mid (\!|\overline{y}v|\!)\big) & \gtrsim & \\
(\!|\overline{c}v|\!)\,. & &
\end{array}$$

In order to prove our main theorem about $(\!|\,.\,|\!)$, we need to fix a few properties of link processes. In the next lemma, part 1 is taken from [22] and 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 that, under certain conditions, a link acts as a substitution.

Lemma 4.3

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

Proof: Part 1 is proven by showing that the relation:

$$\mathcal{R} = \{\big(\ x \to z\ ,\ \nu\, y\,(\, x \to y \mid y \to z\,)\big)\ :\ \ x \text{ and } z \text{ are different from } y\ \}$$

is an *expansion up to context and expansion*. Details can be found in [22].

Part 2 is proven by induction on P and exploiting part 1. All cases of the inductive step are simple. In particular, in the parallel composition and replication cases, we exploit the fact that, given that y does not appear in input-subject position in P, y can only appear in output-subject position in $\langle\!|P|\!\rangle$, which is in πI; exploiting this fact, we can apply Lemma 3.3 (1) and (2), (respectively for parallel composition and replication) to distribute the link process $y \to a$ over the subterms, and then use the induction hypothesis.

The most interesting case is when $P = \overline{y}c$, for some c. Then we have (below we will implicitly use Proposition 2.6):

$$\begin{array}{lcl}
\nu\, y\,(\, y \to a \mid \langle\!|P|\!\rangle) & = & \nu\, y\,\big(\, y \to a \mid \overline{y}(x).\, x \to c\,\big) \\
 & & \text{(definition of } \langle\!|\,.\,|\!\rangle) \\
 & \sim & \nu\, y\,\big(y(x).(\overline{a}(w).\, w \to x \mid y \to a\,) \mid \overline{y}(x).\, x \to c\,\big) \\
 & & \text{(since } y \to a \sim y(x).(\overline{a}(w).\, w \to x \mid y \to a\,)\text{, by Prop. 2.4(2))} \\
 & \gtrsim & (\nu\, y, x)\big(\overline{a}(w).\, w \to x \mid y \to a \mid x \to c\,\big) \\
 & & \text{(a simple law for } \gtrsim\ ) \\
 & \sim & \overline{a}(w).(\nu\, y, x)\big(\, w \to x \mid x \to c \mid y \to a\,\big) \\
 & & \text{(Proposition 2.4(2)-(3))} \\
 & \sim & \overline{a}(w).\nu\, x\,\big(\, w \to x \mid \nu\, y\,(\, x \to c \mid y \to a\,)\big) \\
 & & \text{(laws for } \nu\, y\,) \\
 & \stackrel{\text{def}}{=} & P_1\,.
\end{array}$$

Now, we have to distinguish whether $c = y$ or $c \neq y$. If $c = y$ applying part 1 of the lemma we get $\nu\, y\,(\, x \to c \mid y \to a\,) \gtrsim x \to a$, by which we have:

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

If $c \neq y$, by simple laws for restriction we get that $\nu\, y\,(\, x \to c \mid y \to a\,) \sim x \to c$, by which we have:

$$\begin{array}{lcll}
P_1 & \sim & \overline{a}(w).\nu\, x\,(\, w \to x \mid x \to c\,) & \\
 & \gtrsim & \overline{a}(w).\, w \to c & \text{(applying part 1)} \\
 & = & \langle\!|\overline{x}c\{^{a}\!/_{x}\}|\!\rangle & \text{(def. of } \langle\!|\,.\,|\!\rangle).
\end{array}$$

□

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

Proposition 4.4 (correspondence on transitions) *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 $(\!|P|\!) \stackrel{a(x)}{\longrightarrow} \gtrsim (\!|P'|\!)$;*
2. *$\mu = \overline{a}b$ implies $(\!|P|\!) \stackrel{\overline{a}(x)}{\longrightarrow} \gtrsim x \to b \mid (\!|P'|\!)$, with $x \notin \mathrm{fn}(P')$;*
3. *$\mu = \overline{a}(b)$ implies $(\!|P|\!) \stackrel{\overline{a}(x)}{\longrightarrow} \gtrsim \nu\, b\,(x \to b \mid (\!|P'|\!))$, with $x \notin \mathrm{fn}(P')$;*
4. *$\mu = \tau$ implies $(\!|P|\!) \stackrel{\tau}{\longrightarrow} \gtrsim (\!|P'|\!)$.*

b) Suppose that $(\!|P|\!) \stackrel{\mu}{\longrightarrow} P_1$. Then there are $P' \in \pi^{\mathrm{i}}_{\mathrm{a}}$ and a such that one of the followings holds:

1. *$\mu = a(x)$, for some x, and $P \stackrel{a(x)}{\longrightarrow} P'$, with $P_1 \gtrsim (\!|P'|\!)$;*
2. *$\mu = \overline{a}(x)$, for some x, and either:*
   - *2.a) $P \stackrel{\overline{a}b}{\longrightarrow} P'$, for some b, with $x \notin \mathrm{fn}(P')$ and $P_1 \gtrsim x \to b \mid (\!|P'|\!)$, or*
   - *2.b) $P \stackrel{\overline{a}(b)}{\longrightarrow} P'$, for some b, with $x \notin \mathrm{fn}(P')$ and $P_1 \gtrsim \nu\, b\,(x \to b \mid (\!|P'|\!))$;*
3. *$\mu = \tau$ and $P \stackrel{\tau}{\longrightarrow} P'$, with $P_1 \gtrsim (\!|P'|\!)$.*

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 4.3(2) is used. As an example, we show part a)(4), since b)(3) is handled similarly. The only non-trivial case is when the last rule applied for deriving $P \stackrel{\tau}{\longrightarrow} P'$ is `Com` (we suppose for simplicity that the communicated name is free; the case when it is restricted can be easily accommodated):

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

By applying parts (a)1 and(a) 2 of this lemma to the transitions in the premises of the rule, we have that:

$$(\!|P_1|\!) \stackrel{\overline{a}(x)}{\longrightarrow} \gtrsim x \to b \mid (\!|P_1'|\!) \text{ with } x \notin \mathrm{fn}(P_1'), \text{ and } (\!|P_2|\!) \stackrel{a(x)}{\longrightarrow} \gtrsim (\!|P_2'|\!)\,. \tag{3}$$

Then we have that:

$$\begin{array}{lcll} (\!|P_1 \mid P_2|\!) & \stackrel{\tau}{\longrightarrow} \gtrsim & \nu\, x\,\big(x \to b \mid (\!|P_1' \mid P_2'|\!)\big) & \text{(from (3) and } \texttt{Com}) \\ & \gtrsim & (\!|(P_1' \mid P_2')\{{}^{b}\!/\!_{x}\}|\!) & \text{(Lemma 4.3(2)).} \\ & = & (\!|P_1' \mid P_2'\{{}^{b}\!/\!_{x}\}|\!) & \text{(since } x \notin \mathrm{fn}(P_1')). \end{array}$$

□

Remark 4.5 Note in the above proof that, since P_2 is a $\pi^{\mathrm{i}}_{\mathrm{a}}$-process, the name x does not appear in P_2' in input-subject position: this fact permits applying Lemma 4.3(2). This is the point in the technical development where the "inversion of polarity" property of $\pi^{\mathrm{i}}_{\mathrm{a}}$ turns out to be crucial.

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

Proposition 4.6

a) $P \downarrow p$ *if and only if* $(\!|P|\!) \downarrow p$.

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) $P \Downarrow p$ *if and only if* $(\!|P|\!) \Downarrow p$.

Theorem 4.7 *Let* P *and* Q *be processes in* $\pi^{\mathrm{i}}_{\mathrm{a}}$. *Then* $P \dot{\simeq} Q$ *if and only if* $(\!|P|\!) \dot{\simeq} (\!|Q|\!)$.

The closure of barbed bisimilarity under translation of static contexts of $\pi^{\mathrm{i}}_{\mathrm{a}}$, $\simeq_{\mathrm{tr2}}$, is defined over πI in analogy with $\simeq_{\mathrm{tr1}}$.

Corollary 4.8 *Let* P *and* Q *be processes in* $\pi^{\mathrm{i}}_{\mathrm{a}}$.

1. $P \simeq Q$ *if and only if* $(\!|P|\!) \simeq_{\mathrm{tr2}} (\!|Q|\!)$;
2. $(\!|P|\!) \simeq (\!|Q|\!)$ *implies* $P \simeq Q$.

4.1 The polyadic case

In order to extend the encoding $(\!|\,.\,|\!)$ to polyadic $\pi^{\mathrm{i}}_{\mathrm{a}}$, it is enough to replace Definition 4.1 of link processes and input prefix and output prefix clauses of Table 3 with the following:

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

$$(\!|a(\widetilde{x}).P|\!) \stackrel{\text{def}}{=} a(\widetilde{x}).(\!|P|\!), \quad (\!|\overline{a}\langle\widetilde{b}\rangle|\!) \stackrel{\text{def}}{=} \overline{a}(\widetilde{x}).\,\widetilde{x} \to \widetilde{b}\,,\ \widetilde{x} \text{ fresh}$$

where, for $\widetilde{x} = (x_1, \ldots, x_k)$ and $\widetilde{b} = (b_1, \ldots, b_k)$, $\widetilde{x} \to \widetilde{b}$ stands for $x_1 \to b_1 \mid \cdots \mid x_k \to b_k$. In the definition of link process, the number of components of $\widetilde{x}$ and $\widetilde{y}$ must equal the arity of a and b, which must be the same. Again, the proofs are easily extended. We omit the details.

Let us define now the encoding $[\![\,.\,]\!]$ from polyadic π_{a} to polyadic πI as the composition of $\{\!|\,.\,|\!\}$ and $(\!|\,.\,|\!)$, thus:

$$[\![P]\!] \stackrel{\text{def}}{=} (\!|\{\!|P|\!\}|\!)\,.$$

The closure of barbed bisimilarity under translated (via $[\![\,.\,]\!]$) static contexts of π_{a}, $\simeq_{\mathrm{tr}}$, is defined over πI in analogy with $\simeq_{\mathrm{tr1}}$ and $\simeq_{\mathrm{tr2}}$. As an easy consequence of (the extension to the polyadic case of) Theorems 3.10 and 4.7, of the congruence properties of $\simeq$ over π_{a} and πI and of the compositionality of $[\![\,.\,]\!]$ for parallel composition and restriction, we get:

Corollary 4.9 *Let* P *and* Q *be processes in* π_{a}.

1. $P \simeq Q$ *if and only if* $[\![P]\!] \simeq_{\mathrm{tr}} [\![Q]\!]$;
2. $[\![P]\!] \simeq [\![Q]\!]$ *implies* $P \simeq Q$.

Again, note that Corollary 4.9(2) and the fact that in πI $\approx \subseteq \simeq$ (Proposition 2.6) allow us to use $\approx$ on πI to reason about $\simeq$ on π_{a}: in other words, $[\![P]\!] \approx [\![Q]\!]$ implies $P \simeq Q$.

Remark 4.10 (on the matching operator) In this paper we have considered a language without the matching operator, written $[a = b]$, which allows one to test for the identity of two names. There are indeed many situations in which it is convenient to assume that the observer cannot compare the received names (this is typically the case when passing references to objects or considering names with I/O typing, like in [17]). For this reason it is meaningful to consider a notion of barbed equivalence in which matching is excluded, as we did in the paper. Now, we want to discuss how the presence of matching would affect the results presented so far.

The operational semantics for matching is given by the rule:

$$\texttt{Match}: \ \frac{P \stackrel{\mu}{\longrightarrow} P'}{[a = a]P \stackrel{\mu}{\longrightarrow} P'}\,.$$

The extension of our translation to accommodate matching seems to be problematic. In the first place, in πI it is meaningless to consider matching (which is in fact ruled out in [22]). The reason is that any two distinct names x and y never get instantiated to the same name in πI. This is a consequence of the fact that only bound (hence 'new') names are passed around and received. Thus, for example, given $P \stackrel{\text{def}}{=} \overline{a}b \mid \overline{c}b \mid a(x).c(y).[x = y]p$ in $\pi^{\text{i}}_{\text{a}}$, we have that $P \Downarrow p$, while the (naive) tentative translation of P in πI

$$\overline{a}(x).\, x \to b \mid \overline{c}(y).\, y \to b \mid a(x).c(y).[x = y]p$$

never commits to p, because the bound names x and y are deemed to remain different. The problem here is that, in the translated process, we are trying to compare the *references* (x and y) to the received names, instead of the names themselves. We seem to need here some kind of "dereferencing" mechanism for link processes, which would allow us to access the data pointed to by x and y and compare them. But how to achieve this in the context of our translation is far from obvious. Maybe πI is simply too weak to express matching.

Remark 4.11 (non-soundness of $\langle\!|\, . \,|\!\rangle$ for $\approx$) The following counter-example shows that $\langle\!|\, . \,|\!\rangle$ is not sound for $\approx$. Consider the processes in $\pi^{\text{i}}_{\text{a}}$:

$$P \stackrel{\text{def}}{=} Eq(a,b) \mid \overline{c}a \text{ and } Q \stackrel{\text{def}}{=} Eq(a,b) \mid \overline{c}b$$

where

$$Eq(a,b) \stackrel{\text{def}}{=} !\, a(x).\overline{b}x \mid !\, b(x).\overline{a}x$$

is Honda's *equalizer* [11]. Of course, $P \not\approx Q$, as P can do an output $\overline{c}a$ which Q cannot reply to. However, $\langle\!|P|\!\rangle \approx \langle\!|Q|\!\rangle$, as we prove below.

First, we show that, for any z different from a and b, the relation

$$\mathcal{R} = \{\, (\langle\!|Eq(a,b)|\!\rangle \mid z \to b\,, \langle\!|Eq(a,b)|\!\rangle \mid z \to a\,)\,,\ (\langle\!|Eq(a,b)|\!\rangle \mid z \to a\,, \langle\!|Eq(a,b)|\!\rangle \mid z \to b\,)\,\}$$

is a weak bisimulation *up to expansion and up to context* (defined below) and hence is contained in $\approx$ [5, 22]. From the latter 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 (see [5]), and hence is again included in $\approx$. Thus $\langle\!|P|\!\rangle \approx \langle\!|Q|\!\rangle$.

Recall that $\mathcal{R}$ is a weak bisimulation up to expansion and up to context iff, whenever $A\,\mathcal{R}\,B$ and $A \stackrel{\mu}{\longrightarrow} A'$, then there are a static context $C[\,.\,] = (\nu\, \widetilde{d})(R \mid [\,.\,])$, and processes A_1 and B_1 s.t.

$$A' \sim C[A_1],\ B \stackrel{\widehat{\mu}}{\Longrightarrow} \gtrsim C[B_1] \text{ and } A_1\,\mathcal{R}\,B_1. \tag{4}$$

Suppose now that $A = \langle\!|Eq(a,b)|\!\rangle \mid z \to b$ and $B = \langle\!|Eq(a,b)|\!\rangle \mid z \to a$ (the other case is symmetrical). The only non-trivial case arises when $\mu = z(w)$. Then we have $A \stackrel{z(w)}{\longrightarrow} A \mid \overline{b}(x).\, x \to w \stackrel{\text{def}}{=} A'$. The matching transitions for B are:

$$\begin{array}{lll} B & \stackrel{z(w)}{\longrightarrow} & B \mid \overline{a}(y).\, y \to w \\ & \stackrel{\tau}{\longrightarrow} \sim & \nu\, y \left(B \mid y \to w \mid \overline{b}(x).\, x \to y \right) \quad \text{(interaction at } a\text{)} \\ & \sim & B \mid \overline{b}(x).\nu\, y\, (\, x \to y \mid y \to w\,) \quad \text{(laws for } \mid \text{ and } \nu\, x\,\text{)} \\ & \gtrsim & B \mid \overline{b}(x).\, x \to w \quad \text{(Lemma 4.3(a)).} \end{array}$$

Defining $A_1 \stackrel{\text{def}}{=} A$, $B_1 \stackrel{\text{def}}{=} B$ and $C[\,.\,] \stackrel{\text{def}}{=} [\,.\,] \mid \overline{b}(x).\, x \to w$, we see that (4) is fulfilled.

Incidentally, our counter-example shows that $\approx$ and $\simeq$ are different on $\pi_{\mathrm{a}}^{\mathrm{i}}$. Indeed, $\langle\!|P|\!\rangle \approx \langle\!|Q|\!\rangle$ implies that $\langle\!|P|\!\rangle \simeq_{\mathrm{tr}} \langle\!|Q|\!\rangle$, which in turn implies $P \simeq Q$ in $\pi_{\mathrm{a}}^{\mathrm{i}}$, in virtue of Corollary 4.8(2). From an observational point of view, in the absence of matching it is perfectly reasonable to regard the processes P and Q as equivalent: $Eq(a,b)$ can transform any use of a into a use of b, and vice-versa. There is no means of distinguishing a and b for an external observer without matching.

5 Conclusions

In this paper, we have provided an encoding from the asynchronous π-calculus to πI and proved that two π_{a}-processes are barbed equivalent if and only if their translations are in the closure of barbed bisimilarity under translated static contexts. This shows that external mobility can be programmed via internal mobility.

A problem left open by the present paper is full abstraction of our encoding for barbed equivalence in π_{a} and πI: i.e. whether the translations of two barbed equivalent processes in π_{a} cannot be distinguished by *any* static context in πI (not just by the translated ones). We suspect that, at least for $\pi_{\mathrm{a}}^{\mathrm{i}}$, this is true.

Another interesting problem is that of finding a tractable, bisimulation-like characterization of $\simeq$ on π_{a}, which would allow us to reason about this equivalence in a more direct fashion. In [1], such a characterization is obtained for a version of π_{a} which includes matching and input-guarded summation. As already noted, in absence of matching processes such as P and Q in Remark 4.11 are equated by $\simeq$.

Relationships between π_{a} and the join-calculus should be investigated. The simplicity of our second encoding (which is fully compositional) indicates that the complications due to external mobility mainly arise because some terms of π_{a} do not follow the type discipline of $\pi_{\mathrm{a}}^{\mathrm{i}}$. Now, the join-calculus naturally enjoys an "inversion of polarity" discipline similar to that of $\pi_{\mathrm{a}}^{\mathrm{i}}$. 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. Moreover, 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 simply imposing to π_{a} some natural type discipline. Some work in this direction has been made by R. Amadio [3].

Acknowledgments

Davide Sangiorgi has helped me a lot, both pointing out to me the research direction pursued in the paper and taking part in its initial technical development. Thanks to Paola Quaglia and David Walker for spotting an error in a previous version of the work. Three anonymous referees provided valuable comments and suggestions.

References

[1] R. Amadio and I. Castellani and D. Sangiorgi. On bisimulations for the aynchronous π-calculus. Technical Report RR-2913, INRIA-Sophia Antipolis, 1996. Extended abstract in *Proc. of Concur'96, LNCS 1119*.

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

[3] R. Amadio. An aynchronous model of locality, failure and process mobility. Available at `http://protis.univ-mrs.fr/amadio/locfaIII`.

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

[5] 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. To appear in *Acta Informatica*. An extract appeared in *Proc. of STACS'95*, LNCS 900, Springer Verlag.

[6] M. Boreale and D. Sangiorgi. Some congruence properties for π-calculus bisimilarities. Technical Report RR-2870, INRIA-Sophia Antipolis, 1996. To appear in *Theoretical Computer Science*.

[7] R. De Nicola and M. Hennessy. Testing Equivalence for Processes. *Theoretical Computer Science*, 34:83–133, 1984.

[8] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proc. of 23rd Symposium on Principles of Programming Languages (POPL'96)*.

[9] G. Ferrari, U. Montanari, and P. Quaglia. A π-calculus with explicit substitutions. *Theoretical Computer Science*, 168(2):53–103, 1996.

[10] 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.

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

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

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

[14] 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.

[15] U. Nestmann, B.C. Pierce. Decoding Choiche Encodings. In *Proc. 7th Conference on Concurrency Theory (CONCUR'96)*, LNCS 1119:179–194, Springer Verlag, 1996.

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

[17] B. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *Math. Struct. in Comp. Science*, 11, 1995.

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

[19] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996.

[20] 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.

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

[22] D. Sangiorgi. π-calculus, internal mobility, and agent-passing calculi. *Theoretical Computer Science*, 167(2):235-274, 1996.

[23] 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.