HAL
archives-ouvertes.fr

Separation of synchronous and asynchronous communication via testing

Diletta Cacciagrano, Flavio Corradini, Catuscia Palamidessi

▶ **To cite this version:**

Diletta Cacciagrano, Flavio Corradini, Catuscia Palamidessi. Separation of synchronous and asynchronous communication via testing. Jos Baeten and Iain Phillips. Proceedings of the 12th International Workshop on Expressiveness in Concurrency (EXPRESS 2005), Aug 2005, San Francisco, United States. 154 (3), pp.95-108, 2005, Electronic Notes in Theoretical Computer Science. <10.1016/j.entcs.2006.05.009>. <inria-00201107>

HAL Id: inria-00201107
https://hal.inria.fr/inria-00201107

Submitted on 23 Dec 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Separation of synchronous and asynchronous communication via testing ⋆

D. Cacciagrano *

Dipartimento di Matematica e Informatica, Università degli Studi di Camerino, Camerino, Italy

F. Corradini

Dipartimento di Matematica e Informatica, Università degli Studi di Camerino, Camerino, Italy

C. Palamidessi

INRIA Futurs and LIX École Polytechnique, France

Abstract

One of the early results about the asynchronous π-calculus which significantly contributed to its popularity is the capability of encoding the output prefix of the (choiceless) π-calculus in a natural and elegant way. Encodings of this kind were proposed by Honda and Tokoro, by Nestmann and (independently) by Boudol. We investigate whether the above encodings preserve De Nicola and Hennessy's testing semantics. In this sense, it turns out that, under some general conditions, no encoding of output prefix is able to preserve the must testing. This negative result is due to (a) the non atomicity of the sequences of steps which are necessary in the asynchronous π-calculus to mimic synchronous communication, and (b) testing semantics's sensitivity to divergence.

Key words: Pi-Calculus, Communication, Synchrony, Asynchrony, Testing Semantics.

⋆ Expanded version of a talk presented at EXPRESS 2005 (San Francisco, August 2005).

* Corresponding author.
Email addresses: diletta.cacciagrano@unicam.it (D. Cacciagrano), flavio.corradini@unicam.it (F. Corradini), catuscia@lix.polytechnique.fr (C. Palamidessi).

Preprint accepted for publication in Theoretical Computer Science 16 September 2006

1 Introduction

In recent years, the asynchronous communication paradigm has become more and more popular in the process calculi community. Reasons include the facts that it is easy to implement in a distributed system and that it naturally represents the basic communication mechanism of most Internet and Web applications.

One of the most popular asynchronous calculi is probably the asynchronous π-calculus [17,4]. This is a proper subset of the π-calculus [20], the main differences being the absence of the output prefix and of the choice operator. It is in particular the (absence of) output prefix which is relevant for synchrony. In fact, this construct allows us to express directly that when a process performs an output it suspends until the partner performs the complementary input.

Naturally, the relation between the expressive power of the two calculi has attracted the attention of many researchers. Since the π-calculus contains the asynchronous π-calculus, it is obviously at least as expressive. As for the other direction, the third author has shown a separation result, based on the fact that the choice operator, in combination with synchronous communication, allows us to solve certain problems of distributed agreement that cannot be solved with the asynchronous π-calculus [24].

If we consider the choiceless π-calculus, however, things are quite different. The result in [24] does not say anything concerning the presence/absence of output prefix alone. As a matter of fact, Honda and Tokoro [17], and independently Boudol [4], have proposed (different) encodings of the output prefix in the asynchronous π-calculus, thus justifying the claim that synchronous communication can be "implemented" via asynchronous communication. In both cases the idea is to represent a synchronization via a sequence of asynchronous steps executing a "mutual inclusion" protocol, which involves an exchange of acknowledgment messages. Both encodings are compositional w.r.t. input and output prefixes and homomorphic w.r.t. all other operators. Denoting by $[\![\ ]\!]$ both the encoding proposed by Boudol and that one proposed by Honda and Tokoro, the former maps input and output prefixes according to the rules in Table 2, while the latter maps them according to the rules in Table 1. We give the intuition behind the rules in Table 2, the ones in Table 1 can be explained similarly.

Suppose that we wish to build a system behaving like $(\bar{x}z.S \mid x(y).R)$. In the asynchronous calculus the sending process would be written $(\bar{x}z \mid S)$, but we have to prevent the subprocess S from being active until the message $\bar{x}z$ has been actually received. Then an idea is to guard the process S by the reception

$$[\![x(y).P]\!] = (\nu v)(\bar{x}v \mid v(y).[\![P]\!])$$

$$[\![\bar{x}z.P]\!] = x(v).(\bar{v}z \mid [\![P]\!])$$

where v is a fresh name.

Table 1
Input-Output Rules of Honda and Tokoro's Encoding.

$$[\![x(y).P]\!] = x(u).(\nu v)(\bar{u}v \mid v(y).[\![P]\!])$$

$$[\![\bar{x}z.P]\!] = (\nu u)(\bar{x}u \mid u(v).(\bar{v}z \mid [\![P]\!]))$$

where $u, v \notin fn(P)$.

Table 2
Input-Output Rules of Boudol's Encoding.

of an acknowledgment, that is an explicit continuation, writing the sender as:

$$S' = (\bar{x}z \mid u(v).S)$$

assuming that v is not free in S.

Symmetrically, the receiver would send the acknowledgment just after having received z along x, that is:

$$R' = x(y).(\bar{u}v \mid R)$$

assuming that u is not free in R.

Unfortunately, we cannot apply this simple transformation independently from the context, since in this synchronization protocol there is no particular relation linking the communication channel x with the synchronization channel u. This last name should be known only by the sender and the receiver, while here it can be used also by the environment to interfere with the communication between S' and R' (for instance, S' may accept a message on u from the environment).

To achieve an interference-free synchronization, we have to use a more elaborate protocol, in which the sender and the receiver first exchange *private* links before performing the actual communication. The key observation is that, due to the restriction operator, in a sender like

$$(\nu u)(\bar{x}u \mid u(v).(\bar{v}z \mid S))$$

the subprocess $u(v).(\bar{v}z \mid S)$ can not proceed unless the message $\bar{x}u$ has been received by some other process and such process has sent the acknowledgment $\bar{u}v$. Moreover, the channel name u, being a private name of the sender, can only be used between the sender and the receiver.

Later, in [23] Nestmann has shown that even separate choice can be encoded in the asynchronous π-calculus. This is a stronger result than the ones by Honda-Tokoro and Boudol, as separate choice refers to a construct of guarded choice where the guards can be either input or output prefixes (but not together).

The above encodings significantly contributed to the popularity of the asynchronous π-calculus, but only some weak correctness results were provided for them: Boudol proved, for his encoding, the soundness w.r.t. the Morris' preorder [4]. Nestmann proved that his encoding was both deadlock-free and divergence-free [23].

In this paper we consider a semantics that, in our view, is rather "natural" as a basis for comparing expressiveness of languages: De Nicola and Hennessy's testing semantics [14,15,1,2,12]. Our choice is motivated by the fact that, in this semantics, two processes are considered equivalent when they give the same results under the same experiments. Experiments that, according to the concurrent framework, consist of interactions with a given test-process.

Our main result is that none of the above encodings preserves De Nicola and Hennessy's testing semantics. More precisely, if P and Q are π-calculus processes, $[\![\cdot]\!]$ is one of the mappings mentioned above, and $\mathcal{R}$ is the equivalence generated by the testing semantics, then

$$P \,\mathcal{R}\, Q \;\textit{ if and only if }\; [\![P]\!] \,\mathcal{R}\, [\![Q]\!] \tag{1}$$

does not hold in general.

In order to better explain our contribution, let us briefly recall some concepts behind De Nicola and Hennessy's testing semantics. Let us assume a set of test environments, namely processes with the ability to perform a special action to report success. A process P is embedded into a test environment o via parallel composition. Then, we say that P *may* o if there exists a successful computation of P and o, P *must* o if every computation of P and o is successful and P *fair* o (proposed in [6,22,2]) if each state of every computation of P

and o leads to success after finitely many interactions. Each criterion induces a preorder relation over processes: for any process P and Q, $P \sqsubseteq^{\mathcal{O}}_{sat} Q$ if and only if for each test $o \in \mathcal{O}$, $P\ sat\ o$ implies $Q\ sat\ o$, where sat stands for may, $must$ or $fair$.

The first two authors started to investigate the properties of Boudol's encoding w.r.t. various testing theories in [8]. They were particularly interested in establishing conditions on $[\![\cdot]\!]$ and on $\mathcal{R}$ so that (1) would hold. They realized however that the *only-if* part of (1) cannot hold for testing theories for the reason that the encoded processes are a strict subset of the asynchronous π-calculus. Thus testing a process $[\![P]\!]$ with a test which is not the coding of any process in the π-calculus means testing $[\![P]\!]$ over a set of tests which is "more powerful" than that of P. In fact, a test which is not the result of an encoding in general does not follow the "rules of the game" w.r.t. the communication protocol, and can interact with it in odd ways.

In [8] the first two authors proposed a refinement of the testing theories by considering only encoded tests on the right hand side, and proved that Boudol's encoding $[\![\cdot]\!]$ satisfies the following:

(i) $P \sqsubseteq^{\mathcal{O}}_{may} Q$ iff $[\![P]\!] \sqsubseteq^{[\![\mathcal{O}]\!]}_{may} [\![Q]\!]$;

(ii) $P \sqsubseteq^{\mathcal{O}}_{fair} Q$ iff $[\![P]\!] \sqsubseteq^{[\![\mathcal{O}]\!]}_{fair} [\![Q]\!]$.

In fact, the authors of [8] proved the following stronger result

$$P\ sat\ o \text{ iff } [\![P]\!]\ sat\ [\![o]\!]$$

where sat is either may or $fair$.

In this paper we investigate the must preorder. We focus on the condition that would imply the $must$ version of Properties (i) and (ii), that is:

$$P\ must\ o \text{ iff } [\![P]\!]\ must\ [\![o]\!]$$

We call this condition *preservation of must testing*.

We consider general encodings $[\![\cdot]\!]$ of the (choiceless) π-calculus into the asynchronous π-calculus. We prove that, under some general conditions, namely compositionality w.r.t. prefixes and existence of a diverging encoded term, $[\![\cdot]\!]$ cannot preserve must testing. Note that all the encodings mentioned above, by Boudol, by Honda and Tokoro, and by Nestmann, satisfy these conditions.

The source of the problem is that an (atomic) synchronous communication between a sender and a receiver can be simulated in the asynchronous world but there is no way to guarantee that the sender and the receiver will be

resumed (after communication) at the same time. More precisely, it could be the case that when the sender is ready to proceed the receiver is still engaged in some parts of the protocol, or vice versa. Therefore, there are unfair computations in which one partner is never resumed, and a test based on the interaction, after the communication, with that partner, would not succeed. This is of course not a problem in the synchronous world where the communication partners resume simultaneously.

The fact that our result holds for a general class of encodings points out, to our opinion, an inherent shortcoming of asynchronous communication with respect to synchronous communication. One can also argue that the result points out an inherent shortcoming of the must testing. Must testing is indeed based on the observation of an action indicating the success of the test. If this action follows a communication action, when communication is asynchronous the observation of the success can be delayed forever, even though the test has been successful. This problem can be ruled out, though, by imposing some sort of fairness. We will discuss this idea in Section 8. It is worth noting that our result would not be valid under such a fairness assumption. One indication in this sense is the result in [8], which proves the correctness of Boudol's encoding wrt fair must testing. However fair must testing does not coincide exactly with must testing under a fairness assumption (see [11]) hence the correctness wrt the latter notion is not implied by the result in [8].

The rest of the paper is organized as follows. Section 2 presents the π-calculus and the asynchronous π-calculus. Section 3 formally defines the must testing. Section 4 recalls some basic definitions about encodings. Section 5 proves our main result and Section 6 investigates some consequences of it. All of the proofs omitted in the body of the paper are in the appendix.

A preliminary version of this paper appeared in the proceedings of EXPRESS 2005 [10].

2 The π-calculus and the asynchronous π-calculus

In this section we briefly recall the basic notions about the (choiceless) π-calculus and the asynchronous π-calculus.

2.1 The pi-calculus

Let $\mathcal{N}$ (ranged over by $x, y, z, \dots$) be a set of names. The set $\mathcal{P}_s$ (ranged over by $P, Q, R, \dots$) of π-calculus processes is generated by the following grammar:

$$P ::= \ 0 \ \Big| \ x(y).P \ \Big| \ \tau.P \ \Big| \ \bar{x}y.P \ \Big| \ P \,|\, P \ \Big| \ (\nu x)P \ \Big| \ !\, P$$

The input prefix $y(x).P$, and the restriction $(\nu x)P$, act as name binders for the name x in P. The free names $fn(P)$ and the bound names $bn(P)$ of P are defined as usual. The set of names of P is defined as $n(P) = fn(P) \cup bn(P)$. P is closed if $fn(P) = \emptyset$.

The operational semantics of processes is given via a labelled transition system, whose states are the process themselves. The labels (ranged over by $\mu, \gamma, \dots$) "correspond" to prefixes, input xy, output $\bar{x}y$ and tau τ, and to the bounded output $\bar{x}(y)$ (which models scope extrusion). If $\mu = xy$ or $\mu = \bar{x}y$ or $\mu = \bar{x}(y)$ we define $sub(\mu) = x$ and $obj(\mu) = y$. The functions fn, bn and n are extended to cope with labels as follows:

$$bn(xy) = \emptyset \qquad bn(\bar{x}(y)) = \{y\} \ bn(\bar{x}y) = \emptyset \qquad bn(\tau) = \emptyset$$
$$fn(xy) = \{x, y\} \ fn(\bar{x}(y)) = \{x\} \ fn(\bar{x}y) = \{x, y\} \ fn(\tau) = \emptyset$$

The transition relation is given in Table 3. The symbol $\equiv$ used in Rule Cong stands for the structural congruence. This is the smallest congruence over the set $\mathcal{P}_s$ induced by the axioms in Table 4.

Definition 2.1 (*Weak transitions*) Let P and Q be $\mathcal{P}_s$ processes. Then:

- $P \stackrel{\varepsilon}{\Longrightarrow} Q$ if and only if there exist $P_0, P_1, \dots, P_n \in \mathcal{P}_s$, $n \geq 0$, such that

$$P = P_0 \stackrel{\tau}{\longrightarrow} P_1 \stackrel{\tau}{\longrightarrow} \dots \stackrel{\tau}{\longrightarrow} P_n = Q\,;$$

- $P \stackrel{\mu}{\Longrightarrow} Q$ if and only if there exist P_1, $P_2 \in \mathcal{P}_s$ such that

$$P \stackrel{\varepsilon}{\Longrightarrow} P_1 \stackrel{\mu}{\longrightarrow} P_2 \stackrel{\varepsilon}{\Longrightarrow} Q\,.$$

Notation 2.1 Sometimes we write $P \stackrel{\mu}{\longrightarrow}$ (respectively $P \stackrel{\mu}{\Longrightarrow}$) to mean that there exists P' such that $P \stackrel{\mu}{\longrightarrow} P'$ (respectively $P \stackrel{\mu}{\Longrightarrow} P'$) and we write

$$\text{Input} \quad x(y).P \xrightarrow{xz} P\{z/y\} \quad \text{where } x,\ y,\ z \in \mathcal{N}$$

$$\text{Output/Tau} \quad \alpha.P \xrightarrow{\alpha} P \ \text{ where } \alpha = \bar{x}y \text{ or } \alpha = \tau$$

$$\text{Open} \quad \frac{P \xrightarrow{\bar{x}y} P'}{(\nu y)P \xrightarrow{\bar{x}(y)} P'}\, x \neq y \qquad \text{Res} \quad \frac{P \xrightarrow{\mu} P'}{(\nu y)P \xrightarrow{\mu} (\nu y)P'}\, y \notin n(\mu)$$

$$\text{Par} \quad \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}\, bn(\mu) \cap fn(Q) = \emptyset$$

$$\text{Com} \quad \frac{P \xrightarrow{xy} P',\ Q \xrightarrow{\bar{x}y} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \text{Close} \quad \frac{P \xrightarrow{xy} P',\ Q \xrightarrow{\bar{x}(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')}$$

$$\text{Bang} \quad \frac{P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P' \mid !P}$$

$$\text{Cong} \quad \frac{P \equiv P' \quad P' \xrightarrow{\mu} Q' \quad Q' \equiv Q}{P \xrightarrow{\mu} Q}$$

Table 3
Early operational semantics for $\mathcal{P}_s$ terms.

$a_1)$	$P \equiv Q$ iff Q can be obtained from P by α-conversion
$a_2)$	$(\mathcal{P}_s/\equiv, \mid, 0)$ is a commutative monoid
$a_3)$	$((\nu x)P \mid Q) \equiv (\nu x)(P \mid Q)$, if $x \notin fn(Q)$
$a_4)$	$(\nu x)P \equiv P$, if $x \notin fn(P)$
$a_5)$	$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$

Table 4
The structural congruence.

$P \stackrel{\varepsilon}{\Longrightarrow} \xrightarrow{\mu}$ to mean that there are P' and Q such that $P \stackrel{\varepsilon}{\Longrightarrow} P'$ and $P' \xrightarrow{\mu} Q$. We say that P *diverges*, notation $P\uparrow$, if there exists an infinite sequence of τ transitions starting from P, i.e. $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots P_i \xrightarrow{\tau} P_{i+1} \xrightarrow{\tau} \ldots$ for some $P_1, \ldots P_i, P_{i+1}, \ldots$. In the opposite case, i.e. such an infinite sequence does not exist, we say that P *converges*, notation $P\downarrow$.

The set $\mathcal{P}_a$ of the asynchronous π-calculus processes is generated by the following grammar:

$$P ::= \; 0 \;\Big|\; x(y).P \;\Big|\; \tau.P \;\Big|\; \bar{x}y \;\Big|\; P \mid P \;\Big|\; (\nu x)P \;\Big|\; !\,P$$

The operational semantics of $\mathcal{P}_a$ is given by the rules in Table 3, with the rule Output/Tau replaced by the rules Output and Tau in Table 5. The axioms defining the structural congruence are the same as the ones in Table 4.

Output $\quad \bar{x}y \xrightarrow{\bar{x}y} 0$	Tau $\quad \tau.P \xrightarrow{\tau} P$

Table 5
The rules for Output and Tau in $\mathcal{P}_a$.

The definitions and notation given in the synchronous setting are assumed in the asynchronous one as well. Note that the asynchronous π-calculus is a sub-set of the π-calculus. Indeed, the output-action process $\bar{x}y$ can be thought as the special case of output prefix $\bar{x}y.0$.

3 Must preorder

In this section we briefly summarize the basic definitions behind the testing machinery for the π-calculi. In the following, $\mathcal{P}$ will denote either $\mathcal{P}_s$ or $\mathcal{P}_a$.

Definition 3.1 (*Observers*)

- Let $\mathcal{N}' = \mathcal{N} \cup \{\omega\}$ be the set of names, where we assume that $\omega \notin \mathcal{N}$. By convention we let $fn(\omega) = \{\omega\}$, $bn(\omega) = \emptyset$ and $sub(\omega) = \omega$. The action ω is used to report success.
- The set $\mathcal{O}$ (ranged over by $o, o', o'', \ldots$) of observers is defined like $\mathcal{P}$, where the grammar is extended with the production $P ::= \omega.P$.
- The operational semantics of $\mathcal{P}$ terms is extended to $\mathcal{O}$ by adding the rule $\omega.o \xrightarrow{\omega} o$.

In the following we will use $\langle P \rangle$ to denote some restricted version of P, i.e. any process of the form $(\nu x_1)(\nu x_2)\ldots(\nu x_n)P$, for some $x_1, \ldots, x_n \in fn(P)$.

Definition 3.2 (*Maximal computations*) Given $P \in \mathcal{P}$ and $o \in \mathcal{O}$, a maximal computation from $P \mid o$ is either an infinite sequence of the form

$$P \mid o = P_0 \mid o_0 \xrightarrow{\tau} \langle P_1 \mid o_1 \rangle \xrightarrow{\tau} \langle P_2 \mid o_2 \rangle \xrightarrow{\tau} \ldots$$

or a finite sequence of the form

$$P \mid o = P_0 \mid o_0 \xrightarrow{\tau} \langle P_1 \mid o_1 \rangle \xrightarrow{\tau} \ldots \xrightarrow{\tau} \langle P_n \mid o_n \rangle \not\xrightarrow{\tau} .$$

We are now ready to present the definition of must testing semantics.

Definition 3.3 (*Must semantics*) Given a process $P \in \mathcal{P}$ and an observer $o \in \mathcal{O}$, define P *must* o if and only if *for every* maximal computation

$$P \mid o = P_0 \mid o_0 \xrightarrow{\tau} \langle P_1 \mid o_1 \rangle \xrightarrow{\tau} \ldots \langle P_n \mid o_n \rangle \, [\xrightarrow{\tau} \ldots]$$

there exists $i \geq 0$ such that $P_i \mid o_i \xrightarrow{\omega}$.

Note that P *must* $\omega.o$, for every $P \in \mathcal{P}$ and $o \in \mathcal{O}$.

4 Encodings of the π-calculus into the asynchronous π-calculus

In this section we recall some notions about encodings. In general an encoding is simply a syntactic transformation between languages. We will focus on encodings of the π-calculus into the asynchronous π-calculus, and we will use the notation $[\![\cdot]\!] : \mathcal{P}_s \to \mathcal{P}_a$ to represent one such transformation. In general a "good" encoding satisfies some additional properties, but there is no agreement on a general notion of "good" encoding. Perhaps indeed there should not be a unique notion, but several, depending on the purpose. Anyway, in this paper we focus on the most common requirements, which are the compositionality w.r.t. certain operators, and the correctness w.r.t. a given semantics.

To describe compositionality we use contexts $C[\,]$, which are terms in $\mathcal{P}_a$ with one or more "holes" $[\,]$. Given $P_1, \ldots, P_n \in \mathcal{P}_a$ and a context $C[\,]$ with n holes, $C[P_1, \ldots, P_n]$ denotes the term in $\mathcal{P}_a$ obtained by replacing the occurrences of $[\,]$ by $P_1, \ldots, P_n$ respectively.

Definition 4.1 (*Compositionality w.r.t. an operator*) Let op be an n-ary operator of $\mathcal{P}_s$. We say that an encoding $[\![\cdot]\!]$ is *compositional* w.r.t. op if and only if there exists a context $C_{op}[\,]$ in $\mathcal{P}_a$ such that

$$[\![op(P_1, \ldots, P_n)]\!] = C_{op}[[\![P_1]\!], \ldots, [\![P_n]\!]].$$

Note that a particular case of compositionality is the *homomorphism*, in which an operator of the source language is mapped into an operator of the target language, i.e. $C_{op}[\,] = op'$. Usually the homomorphism is required only for certain operators (typically, in distributed languages, it is required for the parallel construct) while for the others we simply require a compositional translation. However our main result (Theorem 5.1) states the non-existence of encodings under very general conditions, namely no homomorphism is required, only compositionality w.r.t. prefixes.

Concerning semantic correctness, we consider preservation of must testing:

Definition 4.2 (*Soundness, completeness and must-preservation*) Let $[\![\cdot]\!]$ be an encoding from $\mathcal{P}_s$ to $\mathcal{P}_a$, We say that $[\![\cdot]\!]$ is:

- *sound w.r.t. must* iff $\forall\, P \in \mathcal{P}_s$, $\forall\, o \in \mathcal{O}$, $[\![P]\!]$ *must* $[\![o]\!]$ implies P *must* o;
- *complete w.r.t. must* iff $\forall\, P \in \mathcal{P}_s$, $\forall\, o \in \mathcal{O}$, P *must* o implies $[\![P]\!]$ *must* $[\![o]\!]$;
- *must-preserving* iff $[\![\cdot]\!]$ is *sound* and *complete* w.r.t. *must*.

In the following, we will take into account an extended notion of encoding, lifted on the observers. We assume that, given an encoding $[\![\cdot]\!]$ from $\mathcal{P}_s$ to $\mathcal{P}_a$, its lifted version is an encoding from $\mathcal{O}_s$ to $\mathcal{O}_a$ behaving as $[\![\cdot]\!]$, i.e. prefixes contexts do not contain ω. We will keep of using the notation $[\![\cdot]\!]$ to represent one such transformation.

5 Non existence of a must-preserving, input-output prefix compositional encoding

This section is the core of the paper. We prove a general negative result for a large class of encodings of the π-calculus into the asynchronous π-calculus, which includes the ones of Boudol, of Honda and Tokoro, and of Nestmann. Our main result states that any encoding $[\![\cdot]\!]$, that is compositional w.r.t. input and output prefixes and produces at least one divergent term, cannot be *must*-preserving. This negative result is a consequence of (a) the non atomicity of the sequences of steps which are necessary to mimic synchronous communication, and (b) testing semantics's sensitivity to divergence. We remark that we need very few hypotheses to obtain this impossibility result. In particular, we do not require homomorphism, neither w.r.t. parallel operator, nor w.r.t. any other operator.

First, we present the intuition behind this result, showing what happens when $[\![\cdot]\!]$ is Boudol's encoding, Consider the $\mathcal{P}_s$ process P defined as $P = \bar{a}z.\,!\tau.0$, and the observer $o = a(y).\omega.0$. Then the only one maximal computation that

$P \mid o$ can perform is

$$P \mid o = \bar{a}z.\, !\tau.0 \mid a(y).\omega.0 \xrightarrow{\tau} !\tau.0 \mid \omega.0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} 0 \mid 0 \mid \ldots \mid !\tau.0 \mid \omega.0 \xrightarrow{\tau} \ldots$$

Of course P *must* o. Now, consider $[\![P \mid o]\!] = [\![P]\!] \mid [\![o]\!]$ and note $[\![!\tau.0]\!] = !\tau.0$. Consider the following maximal computation:

$$[\![P \mid o]\!] = [\![\bar{a}z.\, !\tau.0]\!] \mid [\![a(y).\omega.0]\!] =$$

$$(\nu u)(\bar{a}u \mid u(v).(\bar{v}z \mid !\tau.0)) \mid a(h).(\nu k)(\bar{h}k \mid k(y).[\![\omega.0]\!]) \equiv$$

$$(\nu u)(\nu k)(\bar{a}u \mid u(v).(\bar{v}z \mid !\tau.0) \mid a(h).(\bar{h}k \mid k(y).[\![\omega.0]\!])) \xrightarrow{\tau}$$

$$(\nu u)(\nu k)(0 \mid u(v).(\bar{v}z \mid !\tau.0) \mid \bar{u}k \mid k(y).[\![\omega.0]\!]) \xrightarrow{\tau}$$

$$(\nu k)(\bar{k}z \mid !\tau.0 \mid k(y).[\![\omega.0]\!]) \xrightarrow{\tau}$$

$$(\nu k)(\bar{k}z \mid 0 \mid !\tau.0 \mid k(y).[\![\omega.0]\!]) \xrightarrow{\tau}$$

$$(\nu k)(\bar{k}z \mid 0 \mid 0 \mid !\tau.0 \mid k(y).[\![\omega.0]\!]) \xrightarrow{\tau}$$

$$\ldots\ldots \xrightarrow{\tau}$$

$$(\nu k)(\bar{k}z \mid 0 \mid 0 \mid \ldots \mid 0 \mid !\tau.0 \mid k(y).[\![\omega.0]\!]) \xrightarrow{\tau}$$

$$\ldots\ldots \xrightarrow{\tau} \ldots\ldots$$

Note that each intermediate state of the computation cannot perform any ω action. Hence, $[\![P]\!]$ *m*$\not$*ust* $[\![o]\!]$.

In the following we will generalize and formalize the idea behind the above counterexample. We first introduce a new formalism, namely the asynchronous π-calculus with focusing terms. This formalism is going to be very convenient to prove our main result.

5.1 The asynchronous π-calculus with focusing contexts

We introduce particular kinds of contexts in the asynchronous π-calculus that differ from those we have introduced in Section 4, $C[\,]$, in that brackets do not disappear once we "fill the holes" with process terms. The reason for introducing these contexts is mainly technical: in the proof of Theorem 5.1,

we need to "isolate" and "monitor" specific subprocesses along a computation. For this reason we call these contexts "focusing contexts". In order to avoid confusion, we use braces in place of square brackets. Additionally, we decorate the braces with indexes $i, j, .. \in \mathbb{N}$, and we stipulate that different occurrences of braces with the same index are to be filled with the same subprocess, while those with different indexes can be filled with different subprocesses. The base case of focusing context is a substitution context of the form $\{\ \}_i\sigma$, where σ is a (name) substitution, that is a function from $\mathcal{N}$ to $\mathcal{N}$. Substitutions will be denoted by $\sigma, \vartheta \ldots$ The domain of a substitution σ is $Dom(\sigma) = \{x |\, x\sigma \neq x\}$; the range of a substitution σ is $Ran(\sigma) = \{y |\, \exists x \in Dom(\sigma)\, x\sigma = y\}$.

Definition 5.1 A *focusing context* $C\{\ \}_i$ for a fixed natural number i is a term generated by the following grammar:

$$C\{\}_i := \{\}_i\sigma \;\Big|\; 0 \;\Big|\; \bar{x}y \;\Big|\; x(y).C\{\}_i \;\Big|\; \tau.C\{\}_i \;\Big|\; (\nu x)C\{\}_i \;\Big|\; C\{\}_i|C\{\}_i \;\Big|\; !C\{\}_i$$

where $x, y \in \mathcal{N}$.

Note that i is a parameter of the grammar and every "hole" in $C\{\ \}_i$ is indexed by i.

For a focusing context $C\{\ \}_i$ and a $\mathcal{P}_a$ process P, define $C\{P\}_i$ as the term obtained by replacing each occurrence $\{\ \}_i\sigma$ in $C\{\ \}_i$ by $\{P\}_i\sigma$.

Definition 5.2 Let P be a $\mathcal{P}_a$ process and i be a natural number. We denote by $\mathcal{L}(P, i)$ (ranged over by $B, B', \ldots$) the language

$$\{C\{P\}_i \mid P \in \mathcal{P}_a \text{ and } C\{\ \}_i \text{ is a focusing context}\}.$$

We give in Table 6 the operational semantics for the language $\mathcal{L}(P, i)$; it is defined on the basis of the one for the asynchronous π-calculus, the only difference being that terms are in $\mathcal{L}(P, i)$ instead than in $\mathcal{P}_a$. Consequently we need to define the notions of application of a substitution and of structural congruence on $\mathcal{L}(P, i)$. These are given in Definition 5.3 and in Table 7, respectively. Note that these definitions are based on the fact that $\mathcal{L}(P, i)$ can be equivalently defined by induction by replacing, in Definition 5.1, $\{\ \}_i\sigma$ by $\{P\}_i\sigma$.

Informally, a $\mathcal{L}(P, i)$ term behaves as a $\mathcal{P}$ term, assuming as a deadlock term every $\{P\}_i$ occurrence that is out of the scope of an input or a τ prefix. In fact, note that in Table 6 there are not any rule for dealing with $\{P\}_i$ terms and, consequently, these terms cannot perform any action. This should not be a concern, because for the proof of Theorem 5.1, for every σ each occurrence of $\{P\}_i\sigma$ is prefixed, i.e. in the scope of an input or a τ prefix.

$$\text{Input} \quad x(y).B \xrightarrow{xz} B\{z/y\} \text{ where } x,\ y,\ z \in \mathcal{N}$$

$$\text{Output} \quad \bar{x}y \xrightarrow{\bar{x}y} 0 \qquad\qquad \text{Tau} \quad \tau.B \xrightarrow{\tau} B$$

$$\text{Open} \quad \frac{B \xrightarrow{\bar{x}y} B'}{(\nu y)B \xrightarrow{\bar{x}(y)} B'}\, x \neq y \qquad\qquad \text{Res} \quad \frac{B \xrightarrow{\mu} B'}{(\nu y)B \xrightarrow{\mu} (\nu y)B'}\, y \notin n(\mu)$$

$$\text{Par} \quad \frac{B \xrightarrow{\mu} B'}{B \mid B_2 \xrightarrow{\mu} B' \mid B_2}\, bn(\mu) \cap fn(B_2) = \emptyset$$

$$\text{Com} \quad \frac{B_1 \xrightarrow{xy} B_1',\ B_2 \xrightarrow{\bar{x}y} B_2'}{B_1 \mid B_2 \xrightarrow{\tau} B_1' \mid B_2'} \qquad\qquad \text{Close} \quad \frac{B_1 \xrightarrow{xy} B_1',\ B_2 \xrightarrow{\bar{x}(y)} B_2'}{B_1 \mid B_2 \xrightarrow{\tau} (\nu y)(B_1' \mid B_2')}$$

$$\text{Bang} \quad \frac{B \xrightarrow{\mu} B'}{!B \xrightarrow{\mu} B' \mid !B}$$

$$\text{Cong} \quad \frac{B \equiv B' \quad B' \xrightarrow{\mu} B'' \quad B'' \equiv B'}{B \xrightarrow{\mu} B'}$$

Table 6
Early operational semantics for $\mathcal{L}(P, i)$ terms.

$a_1)$	$B \equiv B'$ iff B' can be obtained from B by α-conversion
$a_2)$	$(\mathcal{L}(P,i)/\equiv, \mid, 0)$ is a commutative monoid
$a_3)$	$((\nu x)B \mid B') \equiv (\nu x)(B \mid B'),$ if $x \notin fn(B')$
$a_4)$	$(\nu x)B \equiv B,$ if $x \notin fn(B)$
$a_5)$	$(\nu x)(\nu y)B \equiv (\nu y)(\nu x)B$

Table 7
The structural congruence.

In the following we assume that we always use α-conversion before applying a substitution, to avoid collision of names. We also stipulate that the application of a substitution has priority w.r.t. all the other operators of the language.

Notation 5.1 $\sigma\vartheta$ represents the substitution obtained by composing the substitutions σ and ϑ.

Definition 5.3 shows that $\mathcal{L}(P,i)$ is closed under substitution.[1] It follows that the Input rule in Table 6 is well-defined.

Definition 5.3 Given a substitution σ and a term $B \in \mathcal{L}(P,i)$, where all the bound variables are different from the ones in the domain and range of σ, we define $B\sigma$ by induction as follows:

- $(\{P\}_i\vartheta)\sigma = \{P\}_i\vartheta\sigma$;
- $(0)\sigma = 0$;
- $(\bar{x}y)\sigma = \bar{x}\sigma\ y\sigma$;
- $(x(y).B)\sigma = x\sigma\ y.B\sigma$;
- $(\tau.B)\sigma = \tau.B\sigma$;
- $((\nu x)B)\sigma = (\nu x)B\sigma$;
- $(B \mid B')\sigma = B\sigma \mid B'\sigma$;
- $(!B)\sigma = !B\sigma$.

Definition 5.4 Let $B \in \mathcal{L}(P,i)$. An occurrence of $\{P\}_i$ in B is *prefixed* if it is in the scope of an input or a τ prefix. We write *Pref*(B) if each occurrence of $\{P\}_i$ in B is prefixed.

Next we introduce the sort of terms that we are using in the proof.

Definition 5.5 Let P and Q be $\mathcal{P}_a$ terms and i, j be natural numbers such that $i \neq j$. We denote by $\mathcal{L}(P,i,Q,j)$ (ranged over by $D, D', \dots$) the language

$$\{\langle B \mid B' \rangle \mid B \in \mathcal{L}(P,i) \text{ and } B' \in \mathcal{L}(Q,j)\}$$

where $\langle B \mid B' \rangle$ denotes some restricted version of $B \mid B'$, i.e. any term of the form $(\nu x_1)(\nu x_2)\dots(\nu x_n)(B \mid B')$, for some $x_1, \dots, x_n \in fn(B \mid B')$.

Given $D \in \mathcal{L}(P,i,Q,j)$, we write *Pref*$(D)$ if each occurrence of $\{P\}_i$ and of $\{Q\}_j$ in D are prefixed.

We now introduce the concept of *swapping*: given a term $D \in \mathcal{L}(P,i,Q,j)$, $Swap(D,P,Q,i,j)$ is obtained by replacing each occurrence of $\{P\}_i\sigma$ with $\{Q\}_i\sigma$, and each occurrence of $\{Q\}_j\vartheta$ with $\{P\}_j\vartheta$. For simplicity, when P, i, Q, j are clear from the context, we'll write $Swap(D)$ instead of $Swap(D,P,Q,i,j)$.

Definition 5.6 For a term $D \in \mathcal{L}(P,i,Q,j)$ we define $Swap(D)$ by induction

[1] A substitution σ behaves homomorphically w.r.t. all operators, except in the case of substitution composition.

as follows:

$$Swap((\nu x)D) = (\nu x)Swap(D) \qquad Swap(B|B') = Swap(B)|Swap(B')$$
$$Swap(0) = 0 \qquad Swap(\bar{x}y) = \bar{x}y$$
$$Swap(\{P\}_i\sigma) = \{Q\}_i\sigma \qquad Swap(\{Q\}_j\vartheta) = \{P\}_j\vartheta$$
$$Swap(x(y).B) = x(y).Swap(B) \qquad Swap(\tau.B) = \tau.Swap(B)$$
$$Swap((\nu x)B) = (\nu x)Swap(B) \qquad Swap(!B) = !Swap(B)$$

where B, B' denote terms in $\mathcal{L}(P,i) \cup \mathcal{L}(Q,j)$

Given $D \in \mathcal{L}(P,i,Q,j)$, it is easy to see that $Swap(D) \in \mathcal{L}(Q,i,P,j)$.

We denote by $Unbrace(G)$ the $\mathcal{P}_a$ process obtained by removing all the braces from G (both for $G \in \mathcal{L}(P,i)$ and for $G \in \mathcal{L}(P,i,Q,j)$) and by applying the substitutions: for example, $Unbrace(\{P\}_i\sigma \mid \{Q\}_j\vartheta) = P\sigma \mid Q\vartheta$ (where $P\sigma$, $Q\vartheta$ represent the result of the application of the substitutions σ, ϑ to P, Q respectively).

We are interested in terms where all occurrences of braces are prefixed. We have the following property (proven in the appendix).

Lemma 5.1 For every $D \in \mathcal{L}(P,i,Q,j)$,

i) every occurrence of $\{P\}_i$ is prefixed in D iff every occurrence of $\{Q\}_i$ is prefixed in $Swap(D)$;
ii) every occurrence of $\{Q\}_j$ is prefixed in D iff every occurrence of $\{P\}_j$ is prefixed in $Swap(D)$.

Next proposition, whose proof is in the appendix, states some useful operational relations between the asynchronous π-calculus with focusing contexts and the asynchronous π-calculus.

Proposition 5.1 Let $D \in \mathcal{L}(P,i,Q,j)$. Then:

i) $D \xrightarrow{\mu} D'$ implies $D' \in \mathcal{L}(P,i,Q,j)$ and $Unbrace(D) \xrightarrow{\mu} Unbrace(D')$;
ii) $Pref(D)$ and $Unbrace(D) \xrightarrow{\mu} R$ imply that $\exists D' \in \mathcal{L}(P,i,Q,j)$ such that $D \xrightarrow{\mu} D'$ and $R \equiv Unbrace(D')$;
iii) $D \xrightarrow{\tau} D'$ imply $Swap(D) \xrightarrow{\tau} Swap(D')$.

The following lemma shows an interesting property of the asynchronous π-calculus with focusing contexts. It states that two prefixed occurrences of parallel subprocesses P and Q of a process R cannot occur both unprefixed

after one τ transition step from R (even if the transition is an handshake synchronization).

Lemma 5.2 Let $D \in \mathcal{L}(P, i, Q, j)$ such that *Pref*(D) and D contains at least one occurrence of $\{P\}_i$ and one occurrence of $\{Q\}_j$. Assume $D \xrightarrow{\tau} D'$. Then either all occurrences of $\{P\}_i$ are prefixed or all occurrences of $\{Q\}_j$ are prefixed in D'.

Proof: It is sufficient to remark we cannot consume two prefixes with one single transition, because the only rules that allow two processes to make a step at the same time are the communication rules (Com and Close), but in the asynchronous π-calculus only one of these processes can be a prefix. □

5.2 Proof of the main result

Some preliminary lemmas are necessary before giving our main result. We recall that $P\uparrow$ means that there exists an infinite sequence of τ transitions from P, i.e. $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots P_i \xrightarrow{\tau} P_{i+1} \xrightarrow{\tau} \dots$ for some $P_1, \dots P_i, P_{i+1}, \dots$. In the opposite case, i.e. such an infinite sequence does not exist, we say that P *converges*, notation $P\downarrow$.

Lemma 5.3 Let P be a $\mathcal{P}_a$ process. Then:

i) $P\uparrow$ implies $P\sigma\uparrow$, and
ii) $P \xrightarrow{\omega}$ implies $P\sigma \xrightarrow{\omega}$.

Proof: Statement (i) follows from the fact that σ does not rename τ. Statement (ii) follows from the fact that σ is defined on $\mathcal{N}$ and $\omega \notin \mathcal{N}$. □

Lemma 5.4 Let $[\![\cdot]\!]$ be a *must*-preserving encoding. If there exists $P \in \mathcal{P}_s$ such that $[\![P]\!]\uparrow$, then $[\![\omega.0]\!] \xrightarrow{\omega}$.

Proof: Let $P \in \mathcal{P}_s$ such that $[\![P]\!]\uparrow$. Since P *must* $\omega.0$ and the encoding $[\![\cdot]\!]$ is *must*-preserving, then $[\![P]\!]$ *must* $[\![\omega.0]\!]$. Since $[\![P]\!]\uparrow$, we have $[\![\omega.0]\!] \xrightarrow{\omega}$. □

Lemma 5.5 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. input and output prefixes,
2. *must*-preservation,
3. $\exists P \in \mathcal{P}_s$ such that $[\![P]\!]\uparrow$.

Then each $[\,]$ in $C_{x(y)}[\,]$ and $C_{\bar{x}y}[\,]$ is prefixed, i.e. it occurs after an input or a τ prefix.

Proof: By definition we have 0 $m\not{u}st$ $x(y).\omega.0$. Since $[\![\cdot]\!]$ is *must*-preserving, we have $[\![0]\!]$ $m\not{u}st$ $[\![x(y).\omega.0]\!]$. Hence, $[\![0]\!]$ $m\not{u}st$ $C_{x(y)}[[\![\omega.0]\!]]$. By Lemma 5.4 $[\![\omega.0]\!] \xrightarrow{\omega}$. Hence $[\![\omega.0]\!]$ is prefixed in $C_{x(y)}[\ ]$. A similar proof holds for $C_{\bar{x}y}[\ ]$. □

We are now ready to prove our main result that states the non-existence of a *must*-preserving encoding from the π-calculus to the asynchronous π-calculus.

Theorem 5.1 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. input and output prefixes,
2. $\exists P \in \mathcal{P}_s$ such that $[\![P]\!]\uparrow$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof: Assume, by contradiction, that $[\![\cdot]\!]$ is *must*-preserving. Let $P \in \mathcal{P}_s$ s.t. $[\![P]\!]\uparrow$. Since $x(y).P$ *must* $\bar{x}y.\omega.0$ and $[\![\cdot]\!]$ is *must*-preserving, we also have $[\![x(y).P]\!]$ *must* $[\![\bar{x}y.\omega.0]\!]$. By definition $[\![x(y).P]\!] = C_{x(y)}[[\![P]\!]]$ and $[\![\bar{x}y.\omega.0]\!] = C_{\bar{x}y}[[\![\omega.0]\!]]$, for contexts $C_{x(y)}[\ ]$ and $C_{\bar{x}y}[\ ]$ in $\mathcal{P}_a$. By Lemma 5.5, each $[\ ]$ in $C_{x(y)}[\ ]$ and $C_{\bar{x}y}[\ ]$ is prefixed.

In particular, also $[\![\omega.0]\!]$ in $C_{\bar{x}y}[[\![\omega.0]\!]]$ is prefixed (ω cannot appear in $C_{\bar{x}y}[]$ and $C_{x(y)}[]$ because they are contexts in $\mathcal{P}_a$, see Definition 4.1). Furthermore $[\![P]\!]$ occurs prefixed in $C_{x(y)}[[\![P]\!]]$ [2] because otherwise we would have $C_{x(y)}[[\![P]\!]]\uparrow$ while $C_{\bar{x}y}[[\![\omega.0]\!]] \not\xrightarrow{\omega}$, in contradiction with the fact that $C_{x(y)}[[\![P]\!]]$ *must* $C_{\bar{x}y}[[\![\omega.0]\!]]$.

Consider $C_{x(y)}[[\![P]\!]]$ and $C_{\bar{x}y}[[\![\omega.0]\!]]$. Since $C_{x(y)}[[\![P]\!]]$ *must* $C_{\bar{x}y}[[\![\omega.0]\!]]$, for every computation

$$C_{x(y)}[[\![P]\!]] \mid C_{\bar{x}y}[[\![\omega.0]\!]] = A_0 \xrightarrow{\tau} A_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} A_k \xrightarrow{\tau} \ldots \qquad (2)$$

there exists $h \geq 0$ such that $A_h \xrightarrow{\omega}$. Note that we also have that there exists at least one such computation, because k cannot be 0. We consider the first such h, i.e. $\forall k \in [0..(h-1)]$, $A_k \not\xrightarrow{\omega}$. Then, $\forall k \in [0..(h-1)]$ the terms of the form $[\![P]\!]\sigma$ and those of the form $[\![\omega.0]\!]\vartheta$ are prefixed in A_k [3] (where σ, ϑ are substitutions which come from communication of names during the computation).

[2] Note that $[\![P]\!]$ in $C_{x(y)}[[\![P]\!]]$ might appear also in $C_{x(y)}[\ ]$, that is, not necessarily only within the context hole $[\ ]$.

[3] Note that any terms $[\![P]\!]\sigma$, being divergent, must occur prefixed by an input prefix in A_k for every $k \geq 0$ until ω becomes enabled. Otherwise, we would contradict the hypothesis.

Now consider $D_0 = C_{x(y)}\{[\![P]\!]\}_i \mid C_{\bar{x}y}\{[\![\omega.0]\!]\}_j$. By Proposition 5.1-ii, for each computation in (2) there is one of the form

$$C_{x(y)}\{[\![P]\!]\}_i \mid C_{\bar{x}y}\{[\![\omega.0]\!]\}_j = D_0 \xrightarrow{\tau} D_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} D_k \xrightarrow{\tau} \ldots \qquad (3)$$

with $A_k = \mathit{Unbrace}(D_k)$.

Note that, since $[\![P]\!]\sigma$ and $[\![\omega.0]\!]\vartheta$ are prefixed in A_k, the occurrences of $\{\}_i$ and $\{\}_j$ in D_k are prefixed. This is also the reason why we can iterate Proposition 5.1-ii.

Now consider D_h. Since $\{\}_i$ and $\{\}_j$ in D_{h-1} are prefixed and at least one occurrence of $\{\}_j$ in D_h is not prefixed (this is because $A_h = \mathit{Unbrace}(D_h)$ and $A_h \xrightarrow{\omega}$), by Lemma 5.2, it must be the case that each occurrence of $\{\}_i$ (containing occurrences of $[\![P]\!]$) is prefixed in D_h.

Consider now the reverse situation, obtained by switching the prefixes of P and of the test. We still have $\bar{x}y.P$ *must* $x(y).\omega.0$. However, we will show that $[\![\bar{x}y.P]\!]$ *must* $[\![x(y).\omega.0]\!]$, thus contradicting the *must*-preservation hypothesis.

Let us consider the initial term $[\![x(y).\omega.0]\!] \mid [\![\bar{x}y.P]\!] = C_{x(y)}[[\![\omega.0]\!]] \mid C_{\bar{x}y}[[\![P]\!]]$ and the corresponding term in $\mathcal{L}([\![\omega.0]\!], i, [\![P]\!], j)$, $C_{x(y)}\{[\![\omega.0]\!]\}_i \mid C_{\bar{x}y}\{[\![P]\!]\}_j = \mathit{Swap}(D_0)$. By Proposition 5.1-iii, for each computation in (3) there is one of the form

$$C_{x(y)}\{[\![\omega.0]\!]\}_i \mid C_{\bar{x}y}\{[\![P]\!]\}_j = D'_0 \xrightarrow{\tau} D'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} D'_k \xrightarrow{\tau} \ldots \qquad (4)$$

such that $\forall k \in [0..h]$, $D'_k = \mathit{Swap}(D_k)$. Now, observe that in D'_h there is at least one non-prefixed occurrence of $\{\}_j$ while each occurrence of $\{\}_i$ is prefixed. This is a consequence of Lemma 5.1 and of the fact that D_h has the reverse property (reverse in the sense that the role of i and j are reversed).

Hence, $\forall k \in [0..h]$, $\mathit{Unbrace}(D'_k)$ cannot perform ω (because $[\![\omega.0]\!]$ appears in D'_k within prefixed contexts $\{\}_i$) while $\mathit{Unbrace}(D'_h)$ can perform an infinite sequence of τ actions (because the unprefixed occurrence of $\{\ \}_i$ is filled by $[\![P]\!]$). By Proposition 5.1-i, for each computation in (4) there is one of the form

$$C_{x(y)}[[\![\omega.0]\!]] \mid C_{\bar{x}y}[[\![P]\!]] = A'_0 \xrightarrow{\tau} A'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} A'_k \xrightarrow{\tau} \ldots$$

such that $\forall k \in [0..h]$, $A'_k = \mathit{Unbrace}(D'_k)$, $A'_k \not\xrightarrow{\omega}$ and $A'_h \uparrow$. □

6 Other impossibility results

The existence of a divergent process in the target language of the encodings, which is one of the hypotheses of Theorem 5.1, can be guaranteed by suitable

assumptions on the encoding itself and the preservation of the must testing. This section investigates conditions as weak as possible on the encodings which, under the hypothesis of *must*-preservation, ensure the existence of such divergent terms and therefore, together with the compositionality w.r.t. the input and output prefixes, imply the non existence of a *must*-preserving encoding.

The first result (Theorem 6.1) states that the existence of a divergent and a convergent term in the source language whose encodings do not interact with the context is a sufficient condition.

We first need the following lemma.

Lemma 6.1 Let $[\![\cdot]\!]$ be a *must*-preserving encoding and assume that $\exists R \in \mathcal{P}_s$ such that $R\downarrow$ and $fn([\![R]\!]) = \emptyset$. Then every maximal sequence of τ transitions from $[\![\tau.\omega.0]\!]$ is successful, i.e. it reaches a state where the action ω is enabled.

Proof: Suppose, by contradiction, that there exists a maximal sequence of τ transitions from $[\![\tau.\omega.0]\!]$. Since $fn([\![R]\!]) = \emptyset$, it follows that $[\![R]\!]$ $\not\!must$ $[\![\tau.\omega.0]\!]$. Since $[\![\cdot]\!]$ is *must*-preserving, we have R $\not\!must$ $\tau.\omega.0$. But from $R\downarrow$ we derive R *must* $\tau.\omega.0$. Contradiction. □

We prove now Theorem 6.1.

Theorem 6.1 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. input and output prefixes,
2. $\exists Q \in \mathcal{P}_s$ such that $Q\uparrow$ and $fn([\![Q]\!]) = \emptyset$,
3. $\exists R \in \mathcal{P}_s$ such that $R\downarrow$ and $fn([\![R]\!]) = \emptyset$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof: Let $Q \in \mathcal{P}_s$ such that $Q\uparrow$ and $fn([\![Q]\!]) = \emptyset$. Then $Q\, \not\!must\, \tau.\omega.0$ and, by *must*-preservation, $[\![Q]\!]$ $\not\!must$ $[\![\tau.\omega.0]\!]$. By hypothesis (3) and Lemma 6.1 every maximal sequence of τ transitions from $[\![\tau.\omega.0]\!]$ is successful. Therefore, since $fn([\![Q]\!]) = \emptyset$, we necessarily have $[\![Q]\!]\uparrow$ and we can apply Theorem 5.1. □

The following result (Theorem 6.2) states that for the impossibility result it is also sufficient to have homomorphism w.r.t. τ prefix. Note that we do not require homomorphism w.r.t. bang operator. The homomorphism w.r.t. both τ prefix and bang operator would imply immediately the existence of a divergent process in the target language.

We first need the following lemma.

Lemma 6.2 Let $A \in \mathcal{P}$, $o \in O$, such that $A\downarrow$. Then $A\, must\, o$ implies $A\, must\, \tau.o$.

Proof: Assume A *m⁄ust* $\tau.o$. Then there exists a computation

$$A \mid \tau.o = T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} T_n \,[\xrightarrow{\tau} \ldots]$$

such that $\forall i \geq 0, T_i \not\xrightarrow{\omega}$. Since $A\downarrow$, the component $\tau.o$ cannot remain always idle. Let $k \geq 0$ be the index for which the transition $T_k \xrightarrow{\tau} T_{k+1}$ is due to the transition $\tau.o \xrightarrow{\tau} o$. Then there exist $A_0, A_1, \ldots, A_k \in \mathcal{P}$ such that $A = A_0 \xrightarrow{\tau} A_1 \xrightarrow{\tau} ... \xrightarrow{\tau} A_k$ and $\forall i \in [0..k]\ T_i = A_i \mid \tau.o$, while $T_{k+1} = A_k \mid o$. Hence there exists a computation

$$A \mid o = A_0 \mid o \xrightarrow{\tau} A_1 \mid o \xrightarrow{\tau} \ldots \xrightarrow{\tau} A_k \mid o = T_{k+1} \xrightarrow{\tau} \ldots \xrightarrow{\tau} T_n \,[\xrightarrow{\tau} \ldots]$$

Now observe that $T_{k+1} \not\xrightarrow{\omega}$ implies $o \not\xrightarrow{\omega}$ and $\forall i \in [0..k]$, $A_i \not\xrightarrow{\omega}$ because $A_i \in \mathcal{P}$. Hence the above is an unsuccessful computation for $A|o$, and therefore A *m⁄ust* o. □

We prove now Theorem 6.2.

Theorem 6.2 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. input and output prefixes,
2. homomorphism w.r.t. τ prefix,

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof: Suppose, by contradiction, that $[\![\cdot]\!]$ is *must*-preserving. Then, since $!\tau.0$ *m⁄ust* $\tau.\omega.0$, we have $[\![!\tau.0]\!]$ *m⁄ust* $[\![\tau.\omega.0]\!]$. By homomorphism w.r.t. τ prefix we also have $[\![\tau.\omega.0]\!] = \tau.[\![\omega.0]\!]$, and therefore $[\![!\tau.0]\!]$ *m⁄ust* $\tau.[\![\omega.0]\!]$. By Theorem 5.1 we must have $[\![!\tau.0]\!]\downarrow$, hence by Lemma 6.2 we get $[\![!\tau.0]\!]$ *m⁄ust* $[\![\omega.0]\!]$. Therefore, since $!\tau.0$ *must* $\omega.0$, we have that $[\![\cdot]\!]$ cannot be *must*-preserving. □

The next result is, to our opinion, the most surprising. It states that a prefix-compositional encoding cannot be *must*-preserving if the encodings of $\tau.[\,]$ and 0 do not interact with the environment. We first need the following lemma.

Lemma 6.3 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. prefixes,
2. $fn([\![\tau.[\,]]\!]) = fn([\![0]\!]) = \emptyset$,
3. *must*-preservation.

Then $\forall P \in \mathcal{P}_s$, $[\![P]\!]$ *must* $[\![\tau.\omega.0]\!]$.

Proof: Since 0 *must* $\tau.\omega.0$, by *must*-preservation we have $[\![0]\!]$ *must* $[\![\tau.\omega.0]\!]$. Since $fn([\![0]\!]) = \emptyset$, we have that every maximal sequence of τ transitions from $[\![\tau.\omega.0]\!]$ is successful. By compositionality w.r.t. τ prefix we have that

$[\![\tau.\omega.0]\!] = C_\tau[[\![\omega.0]\!]]$, and since $fn([\![\tau.[\,]]\!]) = fn(C_\tau[\,]) = \emptyset$, we have that $C_\tau[\,]$ does not interact with any $[\![P]\!]$. Furthermore, from Theorem 5.1, we know that $[\![P]\!]\downarrow$ for every $P \in \mathcal{P}_s$. We can therefore conclude that every computation of $[\![P]\!] \mid C_\tau[[\![\omega.0]\!]]$ is successful. □

We can now prove Theorem 6.3.

Theorem 6.3 Let $[\![\cdot]\!]$ be an encoding that satisfies:

1. compositionality w.r.t. input, output, and τ prefixes,
2. $fn([\![\tau.[\,]]\!]) = fn([\![0]\!]) = \emptyset$.

Then $[\![\cdot]\!]$ is not *must*-preserving.

Proof: Suppose, by contradiction, that $[\![\cdot]\!]$ is *must*-preserving. Consider a $P \in \mathcal{P}_s$ such that $P\uparrow$ (for instance $P = !\tau.0$). Then P $\not\!must$ $\tau.\omega.0$ while, by Lemma 6.3, $[\![P]\!]$ *must* $[\![\tau.\omega.0]\!]$. □

7 Related work

The expressiveness of several communication mechanisms has been studied in many papers. The standard way in the literature is to define an encoding between the languages equipped with the two communication mechanisms, and to verify the existence of full abstraction results w.r.t. the intended semantics. If we consider in particular synchronous and asynchronous communication, several languages and calculi offer operators to implement either the first or the second mechanism. The most popular calculi are the π-calculus and its variants, for the synchronous communication, and the asynchronous π-calculus and its variants, for the asynchronous communication.

The π-calculus with mixed choice and the asynchronous π-calculus have been compared in [24]. The paper shows that it is not possible to map the π-calculus into the asynchronous π-calculus with a *uniform* encoding while preserving a *reasonable* semantics. We remark that Boudol's encoding is uniform and that may and fair semantics are not *reasonable*, while must is. However, our negative result w.r.t. must is not a consequence of the result in [24]. Indeed, the latter one is relative to the presence of mixed choices, while we do not consider choice in our source language. The separation result in [24] does not hold for the two languages that we consider here.

The attempt to prove a full abstraction result for an encoding that introduces a communication protocol (like the ones of Boudol, Honda and Tokoro, and Nestmann) involves a general difficulty: the presence, in the target language, of

terms which do not follow the rules of the protocol. Thus, for instance, those encodings cannot be fully abstract w.r.t. barbed congruence. The following example, provided by Honda and Yoshida [18], explains why. Consider the processes $P = \bar{x}y.\bar{x}y.0$ and $Q = \bar{x}y.0 | \bar{x}y.0$. They are clearly barbed congruent. However their encodings $[\![P]\!]$ and $[\![Q]\!]$ (where $[\![\cdot]\!]$ is, for instance, the encoding of Boudol, see Table 2) are not congruent because, if we consider $R = x(y).0$, $R \mid [\![P]\!]$ reduces to a process that does not have a $\bar{x}$ barb, while this is not the case for $R \mid [\![Q]\!]$. Note that R is a process that does not "follow the rules" of the protocol, because it does not send the acknowledgment on u to $[\![P]\!]$ (see Table 2), and this is why $[\![P]\!]$ gets stuck.

In literature we find various approaches to the above problem. Typically, one can restrict the contexts of the target language, or impose certain restrictions on its semantics.

One of the papers which uses the restriction on contexts is [25]. The authors consider the polyadic π-calculus and the asynchronous version of the monadic π-calculus as source and target languages respectively, a Boudol-like encoding, and asynchronous barbed congruence as the semantics to be preserved. They consider a type-system that allows them to eliminate the contexts which do not respect "the synchronization protocol" of the encoding, and prove a full abstraction result w.r.t. arbitrary contexts in the source and typeble contexts in the target. The first two authors explore in [8] similar issues w.r.t. testing semantics. The main difference w.r.t. [25] is that in [8] the restriction on contexts is more drastic: in fact, because of the definition of testing semantics, the only relevant contexts are parallel test processes. Moreover, [8] considers only the tests that result from encoding processes of the source language. In [8] it is proved that Boudol's encoding is fully abstract w.r.t. may and fair testing, but not w.r.t. must testing. It is worth noting that the restriction to encoded contexts is sufficient to prove the full abstraction of Boudol's encoding w.r.t. Morris' preorder [7], and it would be sufficient also to prove it w.r.t. asynchronous barbed congruence (this can be easily checked by looking at the proof of Lemma 17 in [25]). On the other hand, with the contexts of [25], the *completeness* result, i.e. the "if part" of the full abstraction, is stronger because it implies the congruence for a larger set of contexts.

Another work with similar issues is [16]. This paper focuses on the ν-calculus, a subset of the asynchronous π-calculus, where only input prefixed terms can be in the scope of the bang operator. Notice that this is not a real restriction, since this kind of replicator is as expressive as the full bang operator [21]. Two operational semantics are considered: the first one, called "synchronous", is essentially the standard reduction semantics of the asynchronous π-calculus. The second one, called "asynchronous", relies on a new input-prefix rule, which allows any process to perform an input action, also when not present syntactically, and make available the corresponding message again, afterwards.

The paper considers two encodings, one for each direction, of the ν-calculus equipped with the synchronous and asynchronous semantics. Then it proves that the first encoding is fully abstract w.r.t. weak bisimulation under some restrictions on the asynchronous semantics. More precisely, it consider only those traces of encoded processes that result from "encoding" traces of the original process. The second encoding is fully abstract w.r.t. weak bisimulation thanks to the fact that the encoding weakens the terms by putting them in parallel with special processes called *identity receptors*.

In [17] the authors consider the two operational semantics of [16] for a variant of the ν-calculus, obtained by replacing bang with recursion. In addition to the results of [16], [17] proves also that weak bisimulation in the asynchronous calculus is strictly weaker than weak bisimulation in the synchronous one, and that it is possible to erase this gap by weakening the synchronous calculus, as proposed in [16].

There are several other calculi which implement specific mechanism of communication. For instance, logical and physical localities, remote communication, higher order communication, and so on. As an example we mention Klaim, an asynchronous language with programming primitives for global computing, obtained by combining features from process algebras and coordination languages. In [13] the authors study the expressive power of Klaim and some of its sublanguages. As usual, this is done by defining encodings from one language to another and by studying fully abstraction results of each encoding w.r.t. barbed bisimilarity and barbed congruence. In particular, it is worth noting that there exists an encoding of the asynchronous π-calculus into a variant of Klaim. The latter is obtained by removing from Klaim the basic action of readiness, the distinction between logical and physical localities and the possibility of higher order and polyadic communication. The full abstraction result for this encoding w.r.t. barbed equivalence is again obtained thanks to the restriction to encoded contexts in the target language.

8 Conclusion and future work

In this paper we have investigated the encodability of output prefix in the asynchronous version of the π-calculus w.r.t. must testing semantics. Our main result is that, if the encoding meets some general requirements, namely compositionality w.r.t. prefixes and existence of a diverging encoded term, then it cannot preserve the must testing. This negative result is a consequence of (a) the non atomicity of the sequences of steps which are necessary in the asynchronous π-calculus to mimic synchronous communication, and (b) testing semantics's sensitivity to divergence.

It is worth noting that the condition of preservation of the must semantics can be interpreted also as implying a restricted form of homomorphism on the parallel operator. In fact, the property P *must* o could be defined as some property $\mathcal{M}$ of $P \,|\, o$. Hence the condition P iff $[\![P]\!]$ *must* $[\![o]\!]$ could be rewritten as $\mathcal{M}(\ |\ o)$ iff $\mathcal{M}([\![P]\!] \mid [\![o]\!])$. However, we do not need full homomorphism for the parallel operator, in the sense that it is not necessary for the occurrences of the operator internal to P and to o.

As a future work, we plan to investigate the possibility of positive results under some "fair" scheduling assumption. The idea of trying the fairness assumption comes from the observation that the negative result for the must testing is essentially due to divergent components and unfair scheduling strategies. Of course, if we imposed fairness on all parts of the computations, then we would have to impose it both on the source and on the target languages in order for the encoding to preserve the semantics. This would weaken the intended result. To avoid this problem, we plan to impose fairness only on asynchronous computations and, more specifically, only on those actions which belong to simulations.

We are also planning to investigate whether the results in this paper apply to broadcasting vs point to point communication.

Acknowledgments

The authors would like to strongly thank anonymous referees for their valuable feedback which helped to improve the work considerably.

References

[1] Boreale, M. & De Nicola, R., Testing Equivalence for Mobile Processes, Information and Computation, **120**, pp. 279-303 (1995).

[2] Boreale, M., De Nicola. R. & Pugliese, R., Basic Observables for Processes, LNCS, **1256**, pp. 482-492 (1997).

[3] Boreale, M., De Nicola, R. & Pugliese, R., Trace and Testing Equivalence in Asynchronous Processes, Information and Computation, **172**, pp. 139-164, Academic Press (2002).

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

[5] De Boer, F.S., Klop, J.W. & Palamidessi, C., Asynchronous Communication in Process Algebra, Proc. of LICS'92, pp. 137-147 (1992).

[6] Brinksma, E., Rensink, A. & Vogler, W., Fair Testing, Proc. of CONCUR'95, LNCS, **962**, pp. 313-327 (1995).

[7] Cacciagrano, D., Comunicazione Sincrona ed Asincrona in Sistemi Concorrenti e Distribuiti, Master Thesys, University of L'Aquila (1999).

[8] Cacciagrano, D. & Corradini, F., On Synchronous and Asynchronous Communication Paradigms, Proc. of ICTCS '01, LNCS, **2202**, pp. 256-268 (2001).

[9] Cacciagrano, D., On Synchronous and Asynchronous Communication: Some Expressiveness Results, PhD Thesys, University La Sapienza of Rome (2004).

[10] Cacciagrano, D., Corradini, F. & Palamidessi, C., Separation of Synchronous and Asynchronous Communication Via Testing. Proc. of EXPRESS'05, ENTCS, **154**(3), pp. 95-108 (2006).

[11] Cacciagrano, D., Corradini, F. & Palamidessi, C., Fair Pi, Proc. of EXPRESS'06, To appear (2006).

[12] Castellani, I. & Hennessy, M., Testing Theories for Asynchronous Languages, Proc. of FSTTCS '98, LNCS, **1530**, pp. 90-101 (1998).

[13] De Nicola, R., Gorla, D. & Pugliese, R., On the expressive power of KLAIM-based calculi, Proc. of EXPRESS '04, ENTCS (2004).

[14] De Nicola, R. & Hennessy, M., Testing Equivalence for Processes, ENTCS, **34**, pp. 83-133 (1984).

[15] Hennessy, M., An Algebraic Theory of Processes, MIT Press, Cambridge (1988).

[16] Honda, H., Two Bisimilarities in ν-calculus, Keio CS report 92-002, Department of Computer Science, Keio University (1992).

[17] Honda, K. & Tokoro, M., An Object calculus for Asynchronous Communication, Proc. of ECOOP '91, LNCS, **512**, pp. 133-147 (1991).

[18] Honda, H. & Yoshida, N., Personal communication (2005).

[19] Milner, R., Communication and Concurrency, Prentice-Hall International (1989).

[20] Milner, R., Parrow, J. & Walker, D., A Calculus of Mobile Processes, Part I and II, Information and Computation, **100**, pp. 1-78 (1992).

[21] Merro, M. & Sangiorgi, D., On asynchrony in name-passing calculi, Proc. of ICALP '98, LNCS, **1443** (1998).

[22] Natarajan, V. & Cleaveland, R., Divergence and Fair Testing, Proc. of ICALP '95, LNCS, **944**, pp. 648-659 (1995).

[23] Nestmann, U., What is a 'Good' Encoding of Guarded Choice?, Information and Computation, **156**, pp. 287-319 (2000).

[24] Palamidessi, C., Comparing the Expressive Power of the Synchronous and Asynchronous π-calculus, Mathematical Structures in Computer Science, **13**(5), pp. 685-719 (2003).

[25] Quaglia, P, & Walker, D., On Synchronous and Asynchronous Mobile Processes, Proc. of FOSSACS 2000, LNCS, **1784**, pp. 283-296 (2000).

Appendix A

In this appendix we give the proofs omitted in Section 5. We start with the proof of Lemma 5.1. We first introduce the following notation.

Notation 8.1 Given $B \in \mathcal{L}(P,i)$ and $Q \in \mathcal{P}_a$, we denote by $B\{\{Q\}_i\sigma/\{P\}_i\sigma\}$ the $\mathcal{L}(Q,i)$ term obtained by replacing every occurrence of $\{P\}_i\sigma$ in B with $\{Q\}_i\sigma$.

Note: we may need to apply α-conversion to B in order to avoid variable-capture.

Lemma 8.1 $\forall B \in \mathcal{L}(P,i)$, *Pref*$(B)$ iff *Pref*$(B\{\{Q\}_i\sigma/\{P\}_i\sigma\})$.

Proof: We first prove the *only if* implication. We proceed by induction on the structure of B.

- $B = 0$ and $B = \bar{x}y$: these cases are trivial, since B does not contain any occurrences of $\{\ \}_i$.
- $B = \{P\}_i\sigma$: this case is trivial, since *Pref*(B) does not hold.
- $B = x(y).B'$, where $B' \in \mathcal{L}(P,i)$: B is such that *Pref*(B); $B\{\{Q\}_i\sigma/\{P\}_i\sigma\}$ $= x(y).B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$ and, by definition, *Pref*$(x(y).B'\{\{Q\}_i\sigma/\{P\}_i\sigma\})$.
- $B = \tau.B'$, where $B' \in \mathcal{L}(P,i)$: this case can be proven similarly to the previous one.
- $B = (\nu x)B_1$, where $B_1 \in \mathcal{L}(P,i)$: *Pref*(B) implies *Pref*(B_1) and, by induction, *Pref*$(B_1\{\{Q\}_i\sigma/\{P\}_i\sigma\})$, which implies *Pref*$(B\{\{Q\}_i\sigma/\{P\}_i\sigma\})$.
- Cases $B = B_1 \mid B_2$ and $B = !B'$, where $B_1, B_2, B' \in \mathcal{L}(P,i)$, can be proven similarly.

To prove the *if* implication notice that $(B\{\{Q\}_i\sigma/\{P\}_i\sigma\})\{\{P\}_i\sigma/\{Q\}_i\sigma\} = B$. □

Lemma 5.1 For every $D \in \mathcal{L}(P,i,Q,j)$,

i) every occurrence of $\{P\}_i$ is prefixed in D iff every occurrence of $\{Q\}_i$ is prefixed in $Swap(D)$;
ii) every occurrence of $\{Q\}_j$ is prefixed in D iff every occurrence of $\{P\}_j$ is prefixed in $Swap(D)$.

Proof: By Definition 5.5, $D = \langle B \mid B' \rangle$, where $B \in \mathcal{L}(P,i)$ and $B' \in \mathcal{L}(Q,j)$. By Definition 5.6, $Swap(D) = \langle Swap(B) \mid Swap(B') \rangle$. We can prove that $\langle Swap(B) \mid Swap(B') \rangle = \langle B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \mid B'\{\{P\}_j\vartheta/\{Q\}_j\vartheta\} \rangle$. Now it suffices to apply Lemma 8.1. □

We now prove Proposition 5.1. We first need the following lemma.

Lemma 8.2 Let $B \in \mathcal{L}(P,i)$. Then:

i) $B \xrightarrow{\mu} B'$ implies $B' \in \mathcal{L}(P,i)$ and $\mathit{Unbrace}(B) \xrightarrow{\mu} \mathit{Unbrace}(B')$;
ii) $\mathit{Pref}(B)$ and $\mathit{Unbrace}(B) \xrightarrow{\mu} R$ imply that there exists $B' \in \mathcal{L}(P,i)$ such that $B \xrightarrow{\mu} B'$ and $R \equiv \mathit{Unbrace}(B')$.
iii) $B \xrightarrow{\mu} B'$ implies $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\mu} B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$.

Proof: This proposition can be proven by induction on the depth of the proof of transitions. Note that $\mathcal{L}(P,i)$ is closed under substitution, and that the structural congruence is preserved by *Unbrace* and the "process substitution" $\{\{Q\}_i\sigma/\{P\}_i\sigma\}$. First we prove item (i).

- $B = \bar{x}y$: trivial, since in this case $\mathit{Unbrace}(B) = B$.
- $B = x(y).B' \xrightarrow{xz} B'\{z/y\}$: since $B \in \mathcal{L}(P,i)$, then $B' \in \mathcal{L}(P,i)$. It follows that $B'\{z/y\} \in \mathcal{L}(P,i)$. And $\mathit{Unbrace}(B) = x(y).\mathit{Unbrace}(B') \xrightarrow{xz} \mathit{Unbrace}(B'\{z/y\})$.
- $B = \tau.B' \xrightarrow{\tau} B'$: since $B \in \mathcal{L}(P,i)$, then $B' \in \mathcal{L}(P,i)$. And $\mathit{Unbrace}(B) = \tau.\mathit{Unbrace}(B') \xrightarrow{\tau} \mathit{Unbrace}(B')$.
- $B = (\nu y)B_1 \xrightarrow{\bar{x}(y)} B'$, where $B_1 \xrightarrow{\bar{x}y} B'$ and $x \neq y$: since $B \in \mathcal{L}(P,i)$, then $B_1 \in \mathcal{L}(P,i)$. By induction $B' \in \mathcal{L}(P,i)$ and $\mathit{Unbrace}(B_1) \xrightarrow{\bar{x}y} \mathit{Unbrace}(B')$. Hence $\mathit{Unbrace}(B) \xrightarrow{\bar{x}(y)} \mathit{Unbrace}(B')$.
- $B = (\nu y)B_1 \xrightarrow{\mu} (\nu y)B'$, where $B_1 \xrightarrow{\mu} B'$ and $y \notin n(\mu)$: this case can be proven similarly to the previous one.
- $B = B_1 \mid B_2 \xrightarrow{\mu} B_1' \mid B_2$, where $B_1 \xrightarrow{\mu} B_1'$ and $bn(\mu) \cap fn(B_2) = \emptyset$: since $B \in \mathcal{L}(P,i)$, then $B_1, B_2 \in \mathcal{L}(P,i)$. By induction hypothesis $B_1' \in \mathcal{L}(P,i)$ and $\mathit{Unbrace}(B_1) \xrightarrow{\mu} \mathit{Unbrace}(B_1')$. We can deduce $\mathit{Unbrace}(B) \xrightarrow{\mu} \mathit{Unbrace}(B_1') \mid \mathit{Unbrace}(B_2) = \mathit{Unbrace}(B_1' \mid B_2)$.
- The cases $B = B_1 \mid B_2 \xrightarrow{\tau} B_1' \mid B_2'$, where $B_1 \xrightarrow{xy} B_1'$ and $B_2 \xrightarrow{\bar{x}y} B_2'$, $B = B_1 \mid B_2 \xrightarrow{\tau} (\nu y)(B_1' \mid B_2')$, where $B_1 \xrightarrow{xy} B_1'$ and $B_2 \xrightarrow{\bar{x}(y)} B_2'$, and $B = !B_1 \xrightarrow{\mu} B_1' \mid !B_1$, can be proven similarly.
- $B \xrightarrow{\mu} B'$, where $B \equiv B_1$, $B_1 \xrightarrow{\mu} B_2$ and $B_2 \equiv B'$: $B \equiv B_1$ implies $B_1 \in \mathcal{L}(P,i)$. By induction hypothesis, $B_1 \xrightarrow{\mu} B_2$ implies $B_2 \in \mathcal{L}(P,i)$ and $\mathit{Unbrace}(B_1) \xrightarrow{\mu} \mathit{Unbrace}(B_2)$. Since $B_2 \equiv B'$ and $B_2 \in \mathcal{L}(P,i)$, then $B' \in \mathcal{L}(P,i)$. Since $\mathit{Unbrace}(B) \equiv \mathit{Unbrace}(B_1)$ and $\mathit{Unbrace}(B') \equiv \mathit{Unbrace}(B_2)$, it follows that $\mathit{Unbrace}(B) \xrightarrow{\mu} \mathit{Unbrace}(B')$.

Now we prove item (ii). By induction on the depth of the proof of transitions. By convenience, we split up the cases following the structure of B.

- $B = 0$: this case is not possible, since $\mathit{Unbrace}(B) = 0$ can not perform any action.
- $B = \bar{x}y$: this case is not possible, since B does not contain any $\{P\}_i\sigma$.

- $B = \{P\}_i\sigma$: this case is not possible, since $\mathit{Pref}(B)$ does not hold.
- $B = x(y).B'$: $\mathit{Unbrace}(B) \xrightarrow{xz} \mathit{Unbrace}(B')\{z/y\} = \mathit{Unbrace}(B'\{z/y\})$ and $B \xrightarrow{xz} B'\{z/y\}$.
- $B = \tau.B'$: this case can be proven similarly to the previous one.
- $B = (\nu y)B_1$: Since $\mathit{Pref}(B)$, then $\mathit{Pref}(B_1)$. We have two cases to consider
  a. $(\nu y)\,\mathit{Unbrace}(B_1) \xrightarrow{\bar{x}(y)} A'$, where $\mathit{Unbrace}(B_1) \xrightarrow{\bar{x}y} A'$ and $x \neq y$: By induction there exists $B' \in \mathcal{L}(P, i)$ such that $B_1 \xrightarrow{\bar{x}y} B'$ and $A' \equiv \mathit{Unbrace}(B')$. Hence $(\nu y)B_1 \xrightarrow{\bar{x}(y)} B'$ and $A' \equiv \mathit{Unbrace}(B')$.
  b. $(\nu y)\,\mathit{Unbrace}(B_1) \xrightarrow{\mu} (\nu y)A'$, where $\mathit{Unbrace}(B_1) \xrightarrow{\mu} A'$ and $y \notin n(\mu)$: this case can be proven similarly to the previous one.
- $B = B_1 \mid B_2$: Since $\mathit{Pref}(B, i)$, then $\mathit{Pref}(B_1)$ and $\mathit{Pref}(B_2)$. We have three cases two consider.
  a. $\mathit{Unbrace}(B_1 \mid B_2) \xrightarrow{\mu} A_1' \mid \mathit{Unbrace}(B_2)$, where $\mathit{Unbrace}(B_1) \xrightarrow{\mu} A_1'$ and $bn(\mu) \cap fn(\mathit{Unbrace}(B_2)) = \emptyset$: By induction there exists $B_1' \in \mathcal{L}(P, i)$ such that $B_1 \xrightarrow{\mu} B_1'$ and $A_1' \equiv \mathit{Unbrace}(B_1')$. Hence $B \xrightarrow{\mu} B_1' \mid B_2$ and $\mathit{Unbrace}(B_1' \mid B_2) \equiv A_1' \mid \mathit{Unbrace}(B_2)$.
  b. $\mathit{Unbrace}(B_1 \mid B_2) \xrightarrow{\tau} A_1' \mid A_2'$, where we have $\mathit{Unbrace}(B_1) \xrightarrow{xy} A_1'$ and $\mathit{Unbrace}(B_2) \xrightarrow{\bar{x}y} A_2'$: By induction there exists $B_1', B_2' \in \mathcal{L}(P, i)$ such that $B_1 \xrightarrow{xy} B_1'$, $B_2 \xrightarrow{\bar{x}y} B_2'$, $A_1' \equiv \mathit{Unbrace}(B_1')$ and $A_2' \equiv \mathit{Unbrace}(B_2')$. Hence $B_1 \mid B_2 \xrightarrow{\tau} B_1' \mid B_2'$ and $\mathit{Unbrace}(B_1' \mid B_2') \equiv A_1' \mid A_2'$.
  c. $\mathit{Unbrace}(B_1 \mid B_2) \xrightarrow{\tau} (\nu y)(A_1' \mid A_2')$, such that $\mathit{Unbrace}(B_1) \xrightarrow{xy} A_1'$ and $\mathit{Unbrace}(B_2) \xrightarrow{\bar{x}(y)} A_2'$: this case can be proven similarly to the previous one.
- $B = !B_1$: this case can be proven similarly.
- $\mathit{Unbrace}(B) \xrightarrow{\mu} A$, where $\mathit{Unbrace}(B) \equiv \mathit{Unbrace}(B')$ (hence $B \equiv B'$), $\mathit{Unbrace}(B') \xrightarrow{\mu} A'$, $A' \equiv A$ and $\mathit{Pref}(B)$: It easy to prove that $\mathit{Pref}(B)$ iff $\mathit{Pref}(B')$ and, by induction, there exists $B'' \in \mathcal{L}(P, i)$ such that $B' \xrightarrow{\mu} B''$ and $A' \equiv \mathit{Unbrace}(B'')$. Hence $A \equiv \mathit{Unbrace}(B'')$. Then $B \xrightarrow{\mu} B''$ and $A \equiv \mathit{Unbrace}(B'')$.

Now we prove item (iii). By induction on the depth of the proof of transitions.

- $B = \bar{x}y$: trivial, since B does not contain any $\{P\}_i\sigma$.
- $B = x(y).B' \xrightarrow{xz} B'\{z/y\}$: then we have the transition $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} = x(y).B'\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{xz} B'\{z/y\}\{\{Q\}_i\sigma\{z/y\}/\{P\}_i\sigma\{z/y\}\}$;
- $B = \tau.B' \xrightarrow{\tau} B'$: this case can be proven similarly;
- $B = (\nu y)B_1 \xrightarrow{\bar{x}(y)} B'$, where $B_1 \xrightarrow{\bar{x}y} B'$ and $x \neq y$: because $B_1 \xrightarrow{\bar{x}y} B'$ and, by induction, $B_1\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\bar{x}y} B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$, it follows that $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\bar{x}(y)} B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$;
- $B = (\nu y)B_1 \xrightarrow{\mu} (\nu y)B'$, where $B_1 \xrightarrow{\mu} B'$ and $y \notin n(\mu)$: this case can be proven similarly to the previous one.
- $B = B_1|B_2 \xrightarrow{\mu} B_1'|B_2$, where $B_1 \xrightarrow{\mu} B_1'$ and $bn(\mu) \cap fn(B_2) = \emptyset$: suppose to

apply α-conversion s.t. $bn(\mu) \cap fn(B_2) = bn(\mu) \cap fn(B_2\{\{Q\}_i\sigma/\{P\}_i\sigma\}) = \emptyset$. By induction, we have that $B_1\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\mu} B_1'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$. Hence $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\mu} (B_1' \mid B_2)\{\{Q\}_i\sigma/\{P\}_i\sigma\}$;

- The cases $B = B_1 \mid B_2 \xrightarrow{\tau} B_1' \mid B_2'$, where $B_1 \xrightarrow{xy} B_1'$ and $B_2 \xrightarrow{\bar{x}y} B_2'$, $B = B_1 \mid B_2 \xrightarrow{\tau} (\nu y)(B_1' \mid B_2')$, where $B_1 \xrightarrow{xy} B_1'$ and $B_2 \xrightarrow{\bar{x}(y)} B_2'$, and $B = !B_1 \xrightarrow{\mu} B_1' \mid !B_1$, can be proven similarly.
- $B \xrightarrow{\mu} B'$, where $B' \equiv B_1$, $B_1 \xrightarrow{\mu} B_2$ and $B_2 \equiv B'$: $B \equiv B_1$ implies $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \equiv B_1\{\{Q\}_i\sigma/\{P\}_i\sigma\}$. By induction $B_1\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\mu} B_2\{\{Q\}_i\sigma/\{P\}_i\sigma\}$. Since $B_2\{\{Q\}_i\sigma/\{P\}_i\sigma\} \equiv B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$ then $B\{\{Q\}_i\sigma/\{P\}_i\sigma\} \xrightarrow{\mu} B'\{\{Q\}_i\sigma/\{P\}_i\sigma\}$.

□

Proposition 5.1 Let $D \in \mathcal{L}(P, i, Q, j)$. Then:

i) $D \xrightarrow{\mu} D'$ implies $D' \in \mathcal{L}(P, i, Q, j)$ and $Unbrace(D) \xrightarrow{\mu} Unbrace(D')$;
ii) $Pref(D)$ and $Unbrace(D) \xrightarrow{\mu} R$ imply that there exists $D' \in \mathcal{L}(P, i, Q, j)$ such that $D \xrightarrow{\mu} D'$ and $R \equiv Unbrace(D')$;
iii) $D \xrightarrow{\tau} D'$ imply $Swap(D) \xrightarrow{\tau} Swap(D')$.

Proof: All the statements (i), (ii) and (iii) are obvious consequences of Lemma 8.2 and Definition 5.5. □