Separation of Synchronous and Asynchronous Communication Via Testing

Diletta Cacciagrano\textsuperscript{1}, Flavio Corradini\textsuperscript{2}

\textit{Dipartimento di Matematica e Informatica}
\textit{Università degli Studi di Camerino, Italy}

Catuscia Palamidessi\textsuperscript{3}

\textit{INRIA Futurs Saclay and LIX}
\textit{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.

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 \cite{16,4}. This is a proper subset of the π-calculus \cite{19}, the main differences being the absence of the output prefix and of the choice operator. It is in particular

\textsuperscript{1} Email: diletta.cacciagrano@unicam.it
\textsuperscript{2} Email: flavio.corradini@unicam.it
\textsuperscript{3} Email: catuscia@lix.polytechnique.fr

1571-0661 © 2006 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2006.05.009
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 [23].

If we consider the choiceless π-calculus, however, things are quite different. The result in [23] does not say anything concerning the presence/absence of output prefix alone. As a matter of fact, Honda and Tokoro [16], 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 acknowledgement 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 1, while the latter maps them according to the rules in Table 2.

We give the intuition behind the rules in Table 1, the ones in Table 2 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 of an
acknowledgement, 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 acknowledgement 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 restriction, 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 this process has sent the acknowledgement \( \bar{u}v \). Moreover, the channel name \( u \), being a private name of the sender, can only be used between the sender and the receiver.

Later, in [22] Nestmann has shown that even separate choice can be encoded in the asynchronous \( \pi \)-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 \( \pi \)-calculus, but only some weak correctness result was 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 [22].

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 [13,14,1,2,11]. 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 \( \pi \)-calculus processes, \([\cdot] \) is one of the mappings mentioned above, and \( \mathcal{R} \) is the equivalence generated by the testing semantics, then

\[ P \mathcal{R} Q \text{ if and only if } [P] \mathcal{R} [Q] \]
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 [6,21,2]) 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 $Q$, $P \sqsubseteq^O_{\text{sat}} Q$ if and only if for each test $o \in O$, $P \text{sat} o$ implies $Q \text{sat} o$, 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 $\llbracket \cdot \rrbracket$ and on $\cal R$ 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 $\pi$-calculus. Thus testing a process $\llbracket P \rrbracket$ with a test which is not the coding of any process in the $\pi$-calculus means testing $\llbracket P \rrbracket$ 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 $\llbracket \cdot \rrbracket$ satisfies the following:

(i) $P \sqsubseteq^O_{\text{may}} Q$ iff $\llbracket P \rrbracket \sqsubseteq^{\text{may}}_{\cal O} \llbracket Q \rrbracket$;

(ii) $P \sqsubseteq^O_{\text{fair}} Q$ iff $\llbracket P \rrbracket \sqsubseteq^{\text{fair}}_{\cal O} \llbracket Q \rrbracket$.

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

$$P \text{sat} o \text{ iff } \llbracket P \rrbracket \text{sat} \llbracket o \rrbracket$$

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:

$$P \text{must} o \text{ iff } \llbracket P \rrbracket \text{must} \llbracket o \rrbracket$$

We call this condition preservation of must testing.

We consider general encodings $\llbracket \cdot \rrbracket$ of the (choiceless) $\pi$-calculus into the asynchronous $\pi$-calculus. We prove that, under some general conditions, namely compositionality w.r.t. prefixes and existence of a diverging encoded term, $\llbracket \cdot \rrbracket$ 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 be-
tween 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 the synchronous one.

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.

Because of space limitation the details of the proofs are omitted. The interested reader can find them in the full version of this paper [10].

2 The pi-calculus and the asynchronous pi-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 \( N \) (ranged over by \( x, y, z, \ldots \)) be a set of names. The set \( P_s \) (ranged over by \( P, Q, R, \ldots \)) of processes is generated by the following grammar:

\[
P :: 0 \mid x(y).P \mid \tau.P \mid \bar{x}y.P \mid P \mid P \mid (\nu x)P \mid !P
\]

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 \( n(P) = fn(P) \cup bn(P) \). Whenever \( fn(P) = \emptyset \), \( P \) is said closed.

The operational semantics of processes is given via a labelled transition system, whose states are the process themselves. The labels (ranged over by \( \mu, \gamma, \ldots \)) correspond to prefixes, input \( x(y) \), output \( \bar{x}y \) and tau \( \tau \), and to the bounded output \( \bar{x}(y) \) (which models scope extrusion). If \( \mu = x(y) \) 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:

\[
bn(x(y)) = \{ y \} \quad bn(\bar{x}(y)) = \{ y \} \quad bn(\bar{x}y) = \emptyset \quad bn(\tau) = \emptyset
\]

\[
fn(x(y)) = \{ x \} \quad fn(\bar{x}(y)) = \{ x \} \quad fn(\bar{x}y) = \{ x, y \} \quad fn(\tau) = \emptyset
\]
The transition relation is given in Table 3. The symbol ≡ 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.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output/Tau</th>
<th>Open</th>
<th>Res</th>
<th>Par</th>
<th>Com</th>
<th>Close</th>
<th>Bang</th>
<th>Cong</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x(y).P \xrightarrow{\beta(z)} P{z/y} ) where ( x, y \in \mathcal{N} )</td>
<td>( \alpha.P \xrightarrow{\alpha} P ) where ( \alpha = \bar{xy} ) or ( \alpha = \tau )</td>
<td>( P \xrightarrow{\bar{xy}} P' ) ( x \neq y )</td>
<td>( (\nu y)P \xrightarrow{\bar{xy}} P' ) ( y \notin n(\mu) )</td>
<td>( P \xrightarrow{\mu} P' ) ( (\nu y)P \xrightarrow{\mu} (\nu y)P' )</td>
<td>( P \xrightarrow{\bar{xy}} P', Q \xrightarrow{\bar{xy}} Q' )</td>
<td>( P</td>
<td>Q \xrightarrow{\tau} P'</td>
<td>Q' )</td>
</tr>
</tbody>
</table>

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

| a1) \( P \equiv Q \) iff \( Q \) can be obtained from \( P \) by alpha-renaming |
| a2) \( (\mathcal{P}_s/\equiv, |, 0) \) is a commutative monoid |
| a3) \( ((\nu x)P | Q) \equiv (\nu x)(P | Q) \), if \( x \notin fn(Q) \) |
| a4) \( (\nu x)P \equiv P \), if \( x \notin fn(P) \) |
| a5) \( (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P \) |

Table 4
The structural congruence.

**Definition 2.1 (Weak transitions)** Let \( P \) and \( Q \) be \( \mathcal{P}_s \) processes. Then:
- \( P \xrightarrow{\varepsilon} Q \) if and only if there exist \( P_0, P_1, \ldots, P_n \in \mathcal{P}_s \), \( n \geq 0 \), such that
  \[
P = P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n = Q ;
\]
- \( P \xrightarrow{\mu} Q \) if and only if there exist \( P_1, P_2 \in \mathcal{P}_s \) such that
  \[
P \xrightarrow{\varepsilon} P_1 \xrightarrow{\mu} P_2 \xrightarrow{\varepsilon} Q .
\]

**Notation 2.1** Sometimes we write \( P \xrightarrow{\mu} (P \xrightarrow{\mu}) \) to mean that there exists \( P' \) such that \( P \xrightarrow{\mu} P' \) \( (P \xrightarrow{\mu} P') \) and we write \( P \xrightarrow{\varepsilon} \mu \) to mean that there are \( P' \).
and $Q$ such that $P \xrightarrow{\varepsilon} P'$ and $P' \xrightarrow{\mu} Q$. We say that $P$ diverges, notation $P \uparrow$, if there exists an infinite sequence of $\tau$ transitions from $P$, i.e. $P = P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots P_i \xrightarrow{\tau} P_{i+1} \xrightarrow{\tau} \ldots$ for some $P_0, P_1, \ldots P_i, P_{i+1}, \ldots$. In the opposite case, i.e. if $P \not\uparrow$, we say that $P$ converges, notation $P \downarrow$.

2.2 The asynchronous pi-calculus

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

$$P ::= 0 \mid x(y).P \mid \tau.P \mid \bar{xy} \mid P \mid P \mid (\nu x)P \mid !P$$

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.

<table>
<thead>
<tr>
<th>Output</th>
<th>$\bar{xy}$ $\xrightarrow{\tau} 0$</th>
<th>Tau</th>
<th>$\tau.P \xrightarrow{\tau} P$</th>
</tr>
</thead>
</table>

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 $\pi$-calculus is a sub-set of the $\pi$-calculus. Indeed, the output-action process $\bar{xy}$ can be thought as the special case of output prefix $\bar{xy}.0$.

3 Must preorder

In this section we briefly summarize the basic definitions behind the testing machinery for the $\pi$-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. By convention we let $fn(\omega) = \{\omega\}$, $bn(\omega) = \emptyset$ and $sub(\omega) = \omega$. The action $\omega$ 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 $\mathcal{P}$ 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

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

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

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 \( P \text{ must } o \) if and only if for every maximal computation \( P \mid o = \langle P_0 \mid o_0 \rangle \xrightarrow{\tau} \langle P_1 \mid o_1 \rangle \xrightarrow{\tau} \ldots \langle P_n \mid o_n \rangle \xrightarrow{\tau} \ldots \) there exists \( i \geq 0 \) such that \( \langle P_i \mid o_i \rangle \xrightarrow{\omega} \).

Note that \( P \text{ must } \omega \cdot o \), for every \( P \in \mathcal{P} \) and \( o \in \mathcal{O} \).

### 4 Encodings of the pi-calculus into the asynchronous pi-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 \( \pi \)-calculus into the asynchronous \( \pi \)-calculus, and we will use the notation \([ \cdot ] : \mathcal{P}_s \rightarrow \mathcal{P}_a\) 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 \( C[\cdot] \), which are terms in \( \mathcal{P}_a \) with one or more “holes” \([\cdot] \). Given \( P_1, \ldots, P_n \in \mathcal{P}_a \) and a context \( C[\cdot] \) with \( n \) holes, \( C[P_1, \ldots, P_n] \) denotes the term in \( \mathcal{P}_a \) obtained by replacing the occurrences of \([\cdot] \) 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 \( C_{op}[\cdot] \) 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 homorphism, in which an operator on the source language is mapped into an operator on the target one, i.e. \( C_{op}[\cdot] = op' \). 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.

Concerning semantic correctness, we consider preservation of must testing:

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

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

5 Non existence of a must-preserving and 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 \(\pi\)-calculus into the asynchronous \(\pi\)-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 in the asynchronous \(\pi\)-calculus 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.

**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.

To clarify the intuition behind this result, we can show what happens when \([\cdot]\) is Boudol’s encoding. Consider the \(\mathcal{P}_s\) process \(P\) defined as \(P = \bar{a}. !\tau.0\), and the observer \(o = a.\omega.0\). Then the only one maximal computation that \(P | o\) can perform is

\[
P | o = \bar{a}. !\tau.0 | a.\omega.0 \xrightarrow{\tau} !\tau.0 | \omega.0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} 0 | 0 | \ldots | !\tau.0 | \omega.0 \xrightarrow{\tau} \ldots
\]

Of course \(P \text{ must } o\). Now, consider \([P | o] = [P] | [o]\) and note that \([P] \uparrow\). Consider the following maximal computation:

\[
[P | o] = [\bar{a}. !\tau.0] | [a.\omega.0] = (\nu u)(\bar{a}u | u(v).(\bar{v} | [ !\tau.0])) | a(h).(\nu k)(\bar{h}k | k.\omega.0)) \equiv \\
(\nu u)(\nu k)(\bar{a}u | u(v).(\bar{v} | [ !\tau.0]) | a(h).\bar{h}k | k.\omega.0)) \xrightarrow{\tau} \\
(\nu u)(\nu k)(0 | u(v).(\bar{v} | [ !\tau.0]) | \bar{v}k | k.\omega.0) \xrightarrow{\tau} \\
(\nu k)(\bar{k} | [ !\tau.0] | k.\omega.0)) = (\nu k)(\bar{k} | [ !\tau.0] | k.\omega.0)) \xrightarrow{\tau} \\
(\nu k)(\bar{k} | 0 | [ !\tau.0] | k.\omega.0)) \xrightarrow{\tau} \\
(\nu k)(\bar{k} | 0 | 0 | [ !\tau.0] | k.\omega.0)) \xrightarrow{\tau} \\
\ldots \xrightarrow{\tau} \\
(\nu k)(\bar{k} | 0 | 0 | \ldots | 0 | [ !\tau.0] | k.\omega.0)) \xrightarrow{\tau}
\]
Note that each intermediate state of the computation cannot perform any ω action. Hence, $[P]$ must $[o]$.

### 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 nonexistence of a must-preserving encoding.

The following theorem 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.

**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.

The following theorem states that for the impossibility result it is also sufficient to have homomorphism w.r.t. τ-prefixes. Note that we don’t require homomorphism w.r.t. bang operator. The homomorphism w.r.t. both τ-prefixes and bang would imply immediately the existence of a divergent process in the target language.

**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.

The next result is, to our opinion, the most surprising. It states that a compositional encoding cannot be must-preserving if the encodings of τ.$\cdot$ and 0 do not interact with the environment.

**Theorem 6.3** Let $[\cdot]$ be an encoding that satisfies:
1. compositionality w.r.t. input, output, and τ prefixes,
2. $fn([\tau.\cdot]) = fn([0]) = \emptyset$.

Then $[\cdot]$ is not must-preserving.
7 Related work

The expressiveness of several communication mechanisms has been studied in many papers. The standard way in literature is to define an encoding between the languages equipped with the two communication mechanisms, and to verify on 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 $\pi$-calculus and its variants, for the synchronous communication, and the asynchronous $\pi$-calculus and its variants, for the asynchronous communication.

The $\pi$-calculus with mixed choice and the asynchronous $\pi$-calculus have been compared in [23]. The paper shows that it is not possible to map the $\pi$-calculus into the asynchronous $\pi$-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 [23]. 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 [23] 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 [17], 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 $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ (where $\llbracket \cdot \rrbracket$ is, for instance, the encoding of Boudol, see Table 1) are not congruent because, if we consider $R = x(y).0$, $R | \llbracket P \rrbracket$ reduces to a process that does not have a $\bar{x}$ barb, while this is not the case for $R | \llbracket Q \rrbracket$. Note that $R$ is a process that does not “follow the rules” of the protocol, because it does not send the acknowledgement on $u$ to $\llbracket P \rrbracket$ (see Table 1), and this is why $\llbracket P \rrbracket$ 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 [24]. The authors consider the polyadic $\pi$-calculus and the asynchronous version of the monadic $\pi$-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. [24] 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. Furthermore, [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 [24]). On the other hand, with the contexts of [24], 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 [15]. This paper focuses on the $\nu$-calculus, a subset of the asynchronous $\pi$-calculus, where only input guarded 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 [20]. Two operational semantics are considered: the first one, called “synchronous”, is essentially the standard reduction semantics of the asynchronous $\pi$-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 $\nu$-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 [16] the authors consider the two operational semantics of [15] for a variant of the $\nu$-calculus, obtained by replacing bang with recursion. In addition to the results of [15], [16] 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 [15].

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 [12] 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 $\pi$-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 communica-
tion. 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 pi-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.

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 result in this paper apply to broadcasting vs point to point communication.

References


