Proc. of EXPRESS'06, ENTCS 175(3):87-108. © Elsevier.

Synchrony vs Asynchrony in Communication Primitives

Daniele Gorla[1]

Dip. di Informatica, Univ. di Roma "La Sapienza", Italy

Abstract

We study, from the expressiveness point of view, the impact of synchrony in the communication primitives that arise when combining together some common and useful programming features like arity of data, communication medium and possibility of pattern matching. For some primitives, we show how their synchronous version can be encoded in their asynchronous counterpart via a fully abstract encoding, thus proving that the two versions have the same expressive power. For the remaining primitives, we prove that no 'reasonable' encoding can exist, thus proving that synchrony adds expressiveness to the language.

Keywords: Expressiveness, Encodings, Communication Primitives, Process Calculi.

1 Introduction

One distinguishing feature of languages for concurrent systems is the choice of the communication primitives they use for inter-process exchange. These primitives can range from very skeletal ones [15,7] to more sophisticated and powerful programming constructs [12,17,3,8]. It is then natural to formally study and compare these primitives from the expressive power perspective. As a consequence, results in this research line show the peculiarities of every primitive and, thus, they can be exploited to choose the 'right' primitive when designing new languages and formalisms.

In [13], we studied *asynchronous* communication primitives and the impact that some very common and useful programming features (like *arity of data*, *communication medium* and possibility of *pattern-matching*) have on their expressiveness. As a result, we came out with:

- eight languages (that, for the sake of uniformity, were small variants of the π-calculus [21]), whose communication primitives were obtained by combining the above mentioned features;
- and with a hierarchy of such languages, based on their relative expressive power.

[1] Email: gorla@di.uniroma1.it

$$\begin{array}{l|l}
\left.\begin{array}{l}\mathrm{M},\ \mathrm{D},\ \mathrm{No}\\ \mathrm{M},\ \mathrm{D},\ \mathrm{Pm}\end{array}\right\}\ s \rightarrow a & \\
\mathrm{M},\ \mathrm{C},\ \mathrm{No}:\ \ s \leftrightarrow a & \left.\begin{array}{l}\mathrm{P},\ \mathrm{D},\ \mathrm{No}\\ \mathrm{P},\ \mathrm{D},\ \mathrm{Pm}\\ \mathrm{P},\ \mathrm{C},\ \mathrm{No}\\ \mathrm{P},\ \mathrm{C},\ \mathrm{Pm}\end{array}\right\}\ s \leftrightarrow a \\
\mathrm{M},\ \mathrm{C},\ \mathrm{Pm}:\ \ s \rightarrow a &
\end{array}$$

Fig. 1. The Impact of Synchrony in Communication Primitives

In this paper, we extend the results presented in [13] to assess, from the expressiveness point of view, the impact of *synchrony* on such primitives. Indeed, as we shall prove, the claim "for many purposes, synchronous message passing can be regarded as a special case of asynchronous message passing" [25] strongly relies on the accessory features that equip the communication primitives. In particular, for each of the eight languages studied in [13], we shall see whether their synchronous version has the same expressive power as their asynchronous counterpart or not. In the first case, we can freely implement the primitives asynchronously, since asynchrony usually poses fewer implementative problems; in the second case, asynchronous implementations are less innocuous.

Our results are summarised in Figure 1. There, M and P denote monadic/polyadic data exchanges; C and D denote channels/dataspaces; Pm and No denote presence/absence of pattern matching; s and a denote synchrony/asynchrony; finally, $s \rightarrow a$ means that the synchronous version of the primitive is strictly more expressive than its asynchronous counterpart, whereas $s \leftrightarrow a$ means that the two versions have the same expressive power.

To study the expressive power of a programming language, several techniques can be exploited. A first, very rough, test is to determine whether a language is Turing complete or not; however, since almost all 'useful' languages are Turing complete, this criterion is too coarse to compare different languages. A second, more informative, approach to show that a language is more expressive than another one is to find a problem that can be solved in the former under some conditions that cannot be met by any solution in the latter.

Another interesting approach to compare two languages consists in encoding one in the other (where an encoding is a function that translates terms of one language in terms of the other language) and studying the properties of the encoding functions. This is the approach we shall follow in this paper and it is very appealing for at least two reasons. First, it is a natural way to show how the key features of a language can be rendered in the other one. Second, it allows us to also carry out quantitative measures on language expressiveness: we can consider aspects like the size and the complexity of the encoding of a term w.r.t. the source term and, consequently, quantitatively assess the encoding proposed.

This paper is organised as follows. In Section 2, we start by comparing the impact of synchrony in the π-calculus [17]; in this way, we gently introduce the reader to the problem and sum up the main related achievements. In Section 3, we present the sixteen concurrent languages arising from the combination of the four features studied (synchrony, arity of data, communication medium and presence of pattern-matching). In Section 4, we present some criteria that an encoding should satisfy to be a good means for language comparison. Then, in Section 5, we prove the results depicted in Figure 1; more precisely, we shall provide (*i*) a fully abstract encoding for all those languages whose synchronous and asynchronous versions have the same expressive power, and (*ii*) a formal proof of the impossibility for a ‘reasonable’ encoding for all those languages where synchrony improves expressiveness. Finally, Section 6 concludes the paper by discussing the results in Figure 1.

2 Synchrony and Asynchrony in the π-calculus

The π-calculus was originally equipped with synchronous, monadic and channel-based communication primitives [17]; a few years later, its asynchronous version appeared in literature [14,2] and became a reference point for its simplicity of distributed implementation [11,22]. Some effort has been spent to prove that the two formalisms have the same expressive power [14,2,23,5]; nowadays, it is widely believed that this is the case.

The idea underlying these encodings is that a synchronous exchange can be simulated by a sequence of asynchronous exchanges. As an example, consider the encodings from [14,2]:

	Honda and Tokoro’s	Boudol’s
$\llbracket\, \overline{a}\langle b\rangle.P \,\rrbracket$	$a(y).(\overline{y}\langle b\rangle \mid \llbracket\, P \,\rrbracket)$	$(\nu c)(\overline{a}\langle c\rangle \mid c(y).(\overline{y}\langle b\rangle \mid \llbracket\, P \,\rrbracket))$
$\llbracket\, a(x).P \,\rrbracket$	$(\nu c)(\overline{a}\langle c\rangle \mid c(x).\llbracket\, P \,\rrbracket)$	$a(z).(\nu d)(\overline{z}\langle d\rangle \mid d(x).\llbracket\, P \,\rrbracket)$

where $\overline{a}\langle b\rangle.P$ denotes the output prefix (send b along a and, after reception, behave like P), $a(x).P$ denotes the input prefix (receive something from a and use it to replace x in the continuation P), $(\nu c)P$ denotes the restriction of c to P (c is accessible only from within P) and $P \mid Q$ denotes the parallel composition of processes P and Q.

These encodings are proved sound by exploiting some ad hoc techniques; e.g., Boudol only proves that his encoding is adequate w.r.t. a Morris-like preorder. On the other hand, [23,5] aim at stronger results for such an encoding: in particular, the first paper shows that it enjoys full abstraction w.r.t. a typed version of barbed equivalence [18], whereas the second paper proves full abstraction w.r.t. to may and fair testing [10,19] restricted to the translation of synchronous contexts. In both cases, it is necessary to reduce the observational power of the contexts since a context that does not abide by the protocol put forward by the encoding can easily break full abstraction. [2] In the first case, the type system characterises the respectful contexts, whereas in the second case the encoding itself yields them. Of course, the first alternative entails a stronger full abstraction result, because in general it accepts more contexts than the translated ones; however, it is usually much more complex. Thus, for the sake of simplicity, in this paper we shall adopt the second alternative; we strongly believe that all our full abstraction results could be also formulated in terms of typed equivalences, instead of translated equivalences.

Recently [6], it has been proved that there is no encoding of the synchronous π-calculus in its asynchronous version preserving *must testing* [10] and enjoying a few minimal properties. [3] This raises the problem of which equivalence should be adopted when defining the full abstraction property to assess expressiveness of two languages. As testified by the case of the π-calculus, such a choice is crucial, mainly when proving that a language $\mathcal{L}_1$ is more

[2] For example, processes $\overline{a}\langle b\rangle.\overline{a}\langle b\rangle$ and $\overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle$ are equated both by barbed equivalence and by may/fair testing; nevertheless, $\llbracket\, \overline{a}\langle b\rangle.\overline{a}\langle b\rangle \,\rrbracket$ and $\llbracket\, \overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle \,\rrbracket$ are not equivalent anymore. The problem is that $\llbracket\, \overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle \,\rrbracket$ can exhibit two top-level outputs, whereas $\llbracket\, \overline{a}\langle b\rangle.\overline{a}\langle b\rangle \,\rrbracket$ only one; if the receiving context sends no acknowledgement back, the second output of $\llbracket\, \overline{a}\langle b\rangle.\overline{a}\langle b\rangle \,\rrbracket$ (that is blocked by the encoding of the first prefix) is never unleashed. The same problem holds for Honda and Tokoro’s encoding, but with processes $a(x).a(y)$ and $a(x) \mid a(y)$.

[3] Another impossibility result is [20], but it relies on the interplay between output prefixes and non-deterministic choice.

expressive than another language $\mathcal{L}_2$: every separation result based on a fixed equivalence could be criticised by saying that it actually compares not the expressive power of the languages, but the discriminating power of the equivalences. For this reason, to prove that $\mathcal{L}_1$ is at least as expressive as $\mathcal{L}_2$, we shall fix a set of minimal properties that every encoding should satisfy and prove that no encoding of $\mathcal{L}_2$ in $\mathcal{L}_1$ satisfying such properties exists.

3 A Family of Process Languages

Syntax. We assume a countable set of *names*, $\mathcal{N}$, ranged over by $a, b, x, y, n, m, \cdots$. Notationally, when a name is used as a channel, we shall prefer letters $a, b, c, \cdots$; when a name is used as an input variable, we shall prefer letters $x, y, z, \cdots$; to denote a generic name, we shall use letters $n, m, \cdots$. The (parametric) syntax of our languages is given in the upper part of Figure 2. The different languages are obtained by plugging into this basic syntax a proper definition for input prefixes (IN) and output processes ($OutProc$). As usual, $\mathbf{0}$ and $P|Q$ denote the terminated process and the parallel composition of two processes, resp.; $(\nu n)P$ restricts to P the visibility of n; finally, **if** $n = m$ **then** P and $!P$ are the standard constructs for name matching and process replication. [4]

In this paper, we study the synchronous/asynchronous versions of the primitives arising by the possible combinations of three features: *arity* (monadic vs. polyadic data), *communication medium* (channels vs. shared dataspaces) and *pattern-matching*. As a result, we have a family of sixteen languages, denoted as $L^{\beta_1}_{\beta_2,\beta_3,\beta_4}$, where

- $\beta_1 = \text{S}$, if we have synchronous communications, and $\beta_1 = \text{A}$, otherwise;
- $\beta_2 = \text{P}$, if we have polyadic data, and $\beta_2 = \text{M}$, otherwise;
- $\beta_3 = \text{C}$, if we have channel-based communications, and $\beta_3 = \text{D}$, otherwise;
- $\beta_4 = \text{PM}$, if we have pattern-matching, and $\beta_4 = \text{NO}$, otherwise.

Now, the full syntax of every language is obtained from the productions in the lower part of Figure 2. There, $\widetilde{-}$ denotes a (possibly empty) sequence of elements of kind $-$. Whenever useful, we shall write a tuple $\widetilde{-}$ as the sequence of its elements, separated by a comma; sometimes, we shall also consider tuples simply as sets. Templates of kind x are called *formal* and can be replaced by every name upon withdrawal of a datum; templates of kind $\ulcorner n \urcorner$ are called *actual* and impose that the datum withdrawn contains exactly name n. As usual, $a(\cdots, x, \cdots).P$ and $(\nu x)P$ bind x in P; the corresponding notions of free and bound names of a process, $\text{FN}(P)$ and $\text{BN}(P)$, and of alpha-conversion, $=_\alpha$, are assumed. We let $\text{N}(P)$ denote $\text{FN}(P) \cup \text{BN}(P)$.

Notice that in $L^{-}_{-,-,\text{PM}}$ the **if-then** construct is redundant because it can be implemented via pattern matching; we kept it for the sake of uniformity with the other languages. Finally, notice that $L^{\text{A}}_{-,-,-}$ can be seen as the sub-language of $L^{\text{S}}_{-,-,-}$ where every output prefix is followed by a $\mathbf{0}$ continuation. Thus, the non-trivial contribution of this work is in giving a converse encoding, or in proving that this cannot exist.

Operational semantics. The operational semantics of the languages is given by means of a *labelled transition system* (LTS) describing the actions a process can perform to evolve. Judgements take the form $P \xrightarrow{\alpha} P'$, meaning that P can become P' upon exhibition of label

[4] Notice that, for the sake of simplicity, we used here replication and a **if-then** construct instead of recursion and the more powerful **if-then-else** used in [13]; this choice does not undermine all our results that still hold also with the other operators.

Basic Processes:

$$P,Q,R \quad ::= \quad \mathbf{0} \;\Big|\; \mathit{OutProc} \;\Big|\; \mathit{IN}.P \;\Big|\; (\nu n)P \;\Big|\; P|Q \;\Big|\; \mathbf{if}\ n = m\ \mathbf{then}\ P \;\Big|\; !P$$

$L^{\mathrm{A}}_{\text{-},\text{-},\text{-}}$	:	$\mathit{OutProc} ::= \mathit{OUT}$		
$L^{\mathrm{S}}_{\text{-},\text{-},\text{-}}$	:	$\mathit{OutProc} ::= \mathit{OUT}.P$		
$L^{\text{-}}_{\mathrm{M,D,NO}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= (x)$	$\mathit{OUT} ::= \langle b\rangle$
$L^{\text{-}}_{\mathrm{M,D,PM}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= (T)$	$\mathit{OUT} ::= \langle b\rangle$
$L^{\text{-}}_{\mathrm{M,C,NO}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= a(x)$	$\mathit{OUT} ::= \overline{a}\langle b\rangle$
$L^{\text{-}}_{\mathrm{M,C,PM}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= a(T)$	$\mathit{OUT} ::= \overline{a}\langle b\rangle$
$L^{\text{-}}_{\mathrm{P,D,NO}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= (\widetilde{x})$	$\mathit{OUT} ::= \langle \widetilde{b}\rangle$
$L^{\text{-}}_{\mathrm{P,D,PM}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= (\widetilde{T})$	$\mathit{OUT} ::= \langle \widetilde{b}\rangle$
$L^{\text{-}}_{\mathrm{P,C,NO}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= a(\widetilde{x})$	$\mathit{OUT} ::= \overline{a}\langle \widetilde{b}\rangle$
$L^{\text{-}}_{\mathrm{P,C,PM}}$	:	$P,Q,R ::= \ldots$	$\mathit{IN} ::= a(\widetilde{T})$	$\mathit{OUT} ::= \overline{a}\langle \widetilde{b}\rangle$

where $\quad T \;::=\; x \;\Big|\; \ulcorner n \urcorner \qquad (\mathit{Template})$

Fig. 2. Syntax of the 16 Languages

α. *Labels* take the form

$$\alpha \quad ::= \quad \tau \;\Big|\; a?\widetilde{b} \;\Big|\; (\nu\widetilde{c})a!\widetilde{b} \;\Big|\; ?\widetilde{b} \;\Big|\; (\nu\widetilde{c})!\widetilde{b}$$

Traditionally, τ denotes an internal computation; $a?\widetilde{b}$ and $(\nu\widetilde{c})a!\widetilde{b}$ denote the reception/sending of a sequence of names $\widetilde{b}$ along channel a; when channels are not present, $?\widetilde{b}$ and $(\nu\widetilde{c})!\widetilde{b}$ denote the withdrawal/emission of $\widetilde{b}$ from/in the shared dataspace. In $(\nu\widetilde{c})a!\widetilde{b}$ and $(\nu\widetilde{c})!\widetilde{b}$, some of the sent names, viz. $\widetilde{c}$ $(\subseteq \widetilde{b})$, are restricted. Notationally, $(\nu\widetilde{c})\text{-}!\widetilde{b}$ stands for either $(\nu\widetilde{c})a!\widetilde{b}$ or $(\nu\widetilde{c})!\widetilde{b}$; similarly, $\text{-}?\widetilde{b}$ stands for either $a?\widetilde{b}$ or $?\widetilde{b}$. As usual, $\mathrm{BN}((\nu\widetilde{c})\text{-}!\widetilde{b}) \triangleq \widetilde{c}$; $\mathrm{FN}(\alpha)$ and $\mathrm{N}(\alpha)$ are defined accordingly.

The LTS provides some rules shared by all the languages; the different semantics are obtained from the axioms for input/output actions. The LTS relies on π-calculus structural equivalence, $\equiv$, that rearranges a process to let it evolve according to the rules of the LTS and that is defined by the following standard axioms [21]:

$$\begin{array}{ccc} P \mid \mathbf{0} \equiv P & P \mid Q \equiv Q \mid P & P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\ !P \equiv P \mid !P & \mathbf{if}\ n = n\ \mathbf{then}\ P \equiv P & P \equiv P' \ \text{ if } P =_{\alpha} P' \\ (\nu n)\mathbf{0} \equiv \mathbf{0} & (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P & P \mid (\nu n)Q \equiv (\nu n)(P \mid Q) \ \text{ if } n \notin \mathrm{FN}(P) \end{array}$$

The common rules of the LTS are reported below (since they are an easy adaptation of an early-style LTS for the π-calculus, we do not comment on them and refer the interested reader to [21]):

$$\frac{P \xrightarrow{?\widetilde{b}} P' \qquad Q \xrightarrow{!\widetilde{b}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad\qquad \frac{P \xrightarrow{a?\widetilde{b}} P' \qquad Q \xrightarrow{a!\widetilde{b}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{\alpha} P' \qquad n \notin \mathrm{N}(\alpha)}{(\nu n)P \xrightarrow{\alpha} (\nu n)P'} \qquad\qquad \frac{P \xrightarrow{(\nu\widetilde{c})\text{-}!\widetilde{b}} P' \qquad n \in \widetilde{b} \setminus \{\text{-}, \widetilde{c}\}}{(\nu n)P \xrightarrow{(\nu n,\widetilde{c})\text{-}!\widetilde{b}} P'}$$

$$\frac{P \xrightarrow{\alpha} P' \qquad \textsc{Bn}(\alpha) \cap \textsc{Fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad\qquad \frac{P \equiv P_1 \xrightarrow{\alpha} P_2 \equiv P'}{P \xrightarrow{\alpha} P'}$$

The rules for output actions in languages $L^{\mathrm{A}}_{\text{-},\mathrm{D},\text{-}}$, $L^{\mathrm{A}}_{\text{-},\mathrm{C},\text{-}}$, $L^{\mathrm{S}}_{\text{-},\mathrm{D},\text{-}}$ and $L^{\mathrm{S}}_{\text{-},\mathrm{C},\text{-}}$ are, respectively,

$$\langle\widetilde{b}\rangle \xrightarrow{!\widetilde{b}} \mathbf{0} \qquad \overline{a}\langle\widetilde{b}\rangle \xrightarrow{a!\widetilde{b}} \mathbf{0} \qquad \langle\widetilde{b}\rangle.P \xrightarrow{!\widetilde{b}} P \qquad \overline{a}\langle\widetilde{b}\rangle.P \xrightarrow{a!\widetilde{b}} P$$

On the other hand, to define the semantics for the input actions, we must specify when a template matches a datum. Intuitively, this happens whenever both have the same length and corresponding fields match (i.e., $\ulcorner n \urcorner$ matches n and x matches every name). This can be formalised via a partial function, called *pattern-matching* and written $\textsc{Match}$, that also returns a substitution σ; the latter will be applied to the process that performed the input to replace formal templates with the corresponding names of the datum retrieved. These intuitions are formalised by the following rules:

$$\textsc{Match}(\,;\,) = \epsilon \qquad \textsc{Match}(\ulcorner n \urcorner; n) = \epsilon \qquad \textsc{Match}(x; n) = \{^{n}/_{x}\}$$

$$\frac{\textsc{Match}(T; b) = \sigma_1 \qquad \textsc{Match}(\widetilde{T}; \widetilde{b}) = \sigma_2}{\textsc{Match}(T, \widetilde{T}\,;\, b, \widetilde{b}) = \sigma_1 \circ \sigma_2}$$

where 'ϵ' denotes the empty substitution and '$\circ$' denotes substitution composition. Now, the operational rules for input actions in languages $L^{\text{-}}_{\text{-},\mathrm{D},\text{-}}$ and $L^{\text{-}}_{\text{-},\mathrm{C},\text{-}}$ are

$$(\widetilde{T}).P \xrightarrow{?\widetilde{b}} P\sigma \qquad a(\widetilde{T}).P \xrightarrow{a?\widetilde{b}} P\sigma$$

whenever $\textsc{Match}(\widetilde{T}\,;\,\widetilde{b}) = \sigma$.

Notation: A substitution σ is a finite partial mapping of names for names; $P\sigma$ denotes the (capture avoiding) application of σ to P. As usual, we let $\Rightarrow$ stand for the reflexive and transitive closure of $\xrightarrow{\tau}$, $\xRightarrow{\alpha}$ stand for $\Rightarrow\xrightarrow{\alpha}\Rightarrow$ and $\xrightarrow{\tau}^{k}$ denote a sequence of k τ-steps. We shall write $P \xrightarrow{\alpha}$ to mean that there exists a process P' such that $P \xrightarrow{\alpha} P'$; a similar notation is adopted for $P \Rightarrow$ and $P \xRightarrow{\alpha}$. Moreover, we let ϕ range over visible actions (i.e. labels different from τ) and ρ to range over (possibly empty) sequences of visible actions. Formally, $\rho \ ::= \ \varepsilon \ \mid \ \phi \cdot \rho$, where '$\varepsilon$' denotes the empty sequence of actions and '$\cdot$' represents concatenation; then, $N \xRightarrow{\varepsilon}$ is defined as $N \Rightarrow$ and $N \xRightarrow{\phi\cdot\rho}$ is defined as $N \xRightarrow{\phi}\xRightarrow{\rho}$.

We conclude this part with a proposition collecting together some properties of the LTSs we have just defined, that will be useful in the sequel; the proof of these results easily follows from the definition of the LTSs.

Proposition 3.1 *The following facts hold:*

(i) *if* $P \in L^{-}_{-,-,\mathrm{NO}}$ *and* $P \xrightarrow{-?\widetilde{b}}$, *then* $P \xrightarrow{-?\widetilde{c}}$ *for every* $\widetilde{c}$ *of the same length as* $\widetilde{b}$*;*

(ii) *if* $P \xrightarrow{\tau} P'$ *then* $P \equiv (\nu\widetilde{c})(P_1 \mid P_2)$ *and* $P' \equiv (\nu\widetilde{c})(P'_1 \mid P'_2)$, *where either* $P_1 \xrightarrow{?\widetilde{b}} P'_1$ *and* $P_2 \xrightarrow{!\widetilde{b}} P'_2$, *or* $P_1 \xrightarrow{a?\widetilde{b}} P'_1$ *and* $P_2 \xrightarrow{a!\widetilde{b}} P'_2$*;*

(iii) *if* $P \in L^{\mathrm{A}}_{-,-,-}$ *and* $P \xrightarrow{(\nu\widetilde{c})-!\widetilde{b}} \xrightarrow{\alpha} P'$, *for* $\widetilde{c} \cap \mathrm{N}(\alpha) = \emptyset$, *then* $P \xrightarrow{\alpha} \xrightarrow{(\nu\widetilde{c})-!\widetilde{b}} P'$*; moreover, if* $\alpha = -?\widetilde{b}$, *then* $P \xrightarrow{\tau} (\nu\widetilde{c})P'$.

4 Quality of an Encoding

We now compare the synchronous and the asynchronous version of the communication primitives just presented by trying to encode every synchronous language in its asynchronous version. Formally, an *encoding* $[\![\cdot]\!]$ is a function mapping terms of the source language into terms of the target language. As already said, the relative expressive power of our languages can be established by defining some criteria to evaluate the quality of the encodings or to prove impossibility results.

Roughly speaking, the encoding must not change the semantics of a source term, i.e. it must preserve the observable behaviour of the term without introducing new behaviours. This means that the encoded term and the source one should be engageable in the same kinds of interactions and that aspects like deadlock and divergence are either present in both terms or in neither of them. We now discuss two possible ways of formalising this requirement. The first one, called *full abstraction*, is usually exploited for encodability results; the second one, called *reasonableness*, is usually exploited in the impossibility results.

Full abstraction. When a language can be encoded in another one, we shall prove that the encoding function enjoys *full abstraction* w.r.t. barbed equivalence restricted to translated contexts. This is a satisfying result since (weak) barbed equivalence is often considered to be the 'touchstone' semantic theory for several process languages. *Barbed equivalence* is obtained by closing under name restriction and parallel composition a relation called *barbed bisimilarity*, that equates two terms that offer the same observable behaviour along all possible computations.

In our framework, a *context* $C[\cdot]$ is a process built up from a hole $[\cdot]$ (to be filled with any process) by using parallel composition and restriction. Formally,

$$C[\cdot] \ ::= \ [\cdot] \ \Big| \ P \mid C[\cdot] \ \Big| \ (\nu n)C[\cdot]$$

Definition 4.1 [Barbs] [5]

- $P \downarrow_{OUT^k}$ holds true iff $P \xrightarrow{(\nu\widetilde{c})!\widetilde{b}}$ and $|\widetilde{b}| = k$; $P \downarrow_{OUT_a}$ holds true iff $P \xrightarrow{(\nu\widetilde{c})a!\widetilde{b}}$.
- $P \downarrow_{IN^k}$ holds true iff $P \xrightarrow{?\widetilde{b}}$ and $|\widetilde{b}| = k$; $P \downarrow_{IN_a}$ holds true iff $P \xrightarrow{a?\widetilde{b}}$.
- Let o range over $\{OUT^k, OUT_a, IN^k, IN_a\}$; then, $P \Downarrow_o$ stands for $\exists P'.P \Rightarrow P' \downarrow_o$.

Definition 4.2 [Barbed Bisimilarity and Equivalence] A symmetric relation $\mathfrak{R}$ between processes is a *barbed bisimulation* if, for every $(P, Q) \in \mathfrak{R}$, it holds that

(i) $P \downarrow_o$ implies $Q \Downarrow_o$, and
(ii) $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow Q'$, for some Q' such that $(P', Q') \in \mathfrak{R}$.

Barbed bisimilarity, $\dot{\cong}$, is the largest barbed bisimulation. P and Q are *barbed equivalent*, written $P \cong Q$, if and only if $C[P] \mathrel{\dot{\cong}} C[Q]$, for every context $C[\cdot]$.

As already said in Section 2, a good form of full abstraction for a given encoding $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ is w.r.t. *translated observers*, i.e. observers that abide by the schema imposed by the encoding function. Thus, we now restrict the equivalences introduced so far to keep this choice into account: first, not all the barbs from Definition 4 can be observed by a translated observer; second, we only need to consider translated contexts when defining barbed equivalence. The following definition formalises these ideas; there, we say that an action α performed by a $\mathcal{L}_2$-process can be consumed by the translation of a $\mathcal{L}_1$-process R if $[\![R]\!] \overset{\rho}{\Longrightarrow} \xrightarrow{\alpha'}$, with α' *synchronisable* with α (i.e., with $\alpha' = \text{-}?\widetilde{b}$ and $\alpha = (\nu\widetilde{c})\text{-}!\widetilde{b}$, or vice versa) and $\textsc{Bn}(\rho) \cap \textsc{N}(\alpha) = \emptyset$.

Definition 4.3 [Translated Barbed Bisimilarity and Equivalence] Fix an encoding $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$.

- Let P be a $\mathcal{L}_2$-process; $P \downarrow_o^{tr}$ holds true iff $P \downarrow_o$ with an action that can be consumed by the translation of some $\mathcal{L}_1$-process; $P \Downarrow_o^{tr}$ is defined accordingly.
- A symmetric relation $\mathfrak{R}$ between $\mathcal{L}_2$-processes is a *translated barbed bisimulation* if, for every $(P, Q) \in \mathfrak{R}$, it holds that
  (i) $P \downarrow_o^{tr}$ implies $Q \Downarrow_o^{tr}$, and
  (ii) $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow Q'$, for some Q' such that $(P', Q') \in \mathfrak{R}$.
  Translated barbed bisimilarity, $\dot{\cong}^{tr}$, is the largest translated barbed bisimulation.
- P and Q are *translated barbed equivalent*, written $P \cong^{tr} Q$, if and only if $C[P] \mathrel{\dot{\cong}^{tr}} C[Q]$, for every context $C[\cdot]$ resulting from the translation of a $\mathcal{L}_1$-context via $[\![\cdot]\!]$ extended with $[\![[\cdot]]\!] \triangleq [\cdot]$.

Reasonable Encoding. To prove that two languages have different expressive power, we shall leave full abstraction out (since it requires to fix an equivalence relation): instead, we shall collect together some 'reasonable' requirements and prove that no encoding function satisfying them exists. The main requirement is *faithfulness*: the encoding must preserve and reflect the barbs (i.e., the encoding should maintain all the original barbs without introducing new ones); moreover, it should also preserve and reflect divergence. However, these

[5] In order to obtain meaningful equivalences, barbs in $L_{\text{M,D,-}}^{\text{-}}$ should be defined by also specifying the argument of the action. However, since we shall not give full abstraction results for such languages, we ignore this aspect. By the way, notice that for languages $L_{\text{P,D,-}}^{\text{-}}$ the arguments of the action are not strictly necessary, since the barbed equivalence arising from this different kind of barbs would coincide with $\cong$ defined here.

two requirements alone are not enough to control deadlock. Thus, we shall also require that the computations of a process correspond to the computations of its encoding, and vice versa; this property is usually known as *operational correspondence*. Furthermore, a good encoding cannot depend on the particular names involved in the source process, since we are dealing with a family of name-passing languages; we call this property *name invariance*. Finally, the encoding should not decrease the degree of parallelism in favour of centralised entities that control the behaviour of the encoded term; we express this last property as *homomorphism w.r.t.* '|'.

Definition 4.4 [Reasonable Encoding] An encoding $[\![\cdot]\!]$ is *reasonable* if it enjoys the following properties:

(i) *(homomorphism w.r.t. '|'):* $[\![P_1|P_2]\!] \triangleq [\![P_1]\!] \mid [\![P_2]\!]$.

(ii) *(name invariance):* $[\![P\sigma]\!] \triangleq [\![P]\!]\sigma$, for every permutation of source language names σ.

(iii) *(faithfulness):* $P \Downarrow_o$ iff $[\![P]\!] \Downarrow_{o'}$; P diverges iff $[\![P]\!]$ diverges.

(iv) *(operational correspondence):*
   (a) if $P \Rightarrow P'$ then $[\![P]\!] \Rightarrow [\![P']\!]$;
   (b) if $[\![P]\!] \Rightarrow Q$ then there exists a P' such that $P \Rightarrow P'$ and $Q \Rightarrow [\![P']\!]$.

Evaluation criteria. To sum up, for our encodability results we aim at proving that the encoding function does not introduce divergence and that it enjoys full abstraction w.r.t. translated barbed equivalence; on the other hand, we shall establish our impossibility results by proving that no reasonable encoding exists. Usually, the latter proofs are by contradiction: we assume that a reasonable encoding exists and show that it cannot be reasonable. This can require a lot of work. However, in this paper, we shall exploit the simple proof-technique developed in [13]: exhibit a process that cannot reduce but whose encoding reduces. This fact, together with operational correspondence, implies that the encoding introduces divergence.

Proposition 4.5 *Let P be a process such that* $P \not\xrightarrow{\tau}$ *but* $[\![P]\!] \xrightarrow{\tau}$ *; then,* $[\![\cdot]\!]$ *is not reasonable.*

5 The Impact of Synchrony in Communication Primitives

In this section, we first consider those languages in which synchrony does not play a crucial rôle, i.e. those primitives whose synchronous and asynchronous versions have the same expressive power. We then analyse those primitives in which the presence of synchrony matters, i.e. those primitives whose asynchronous version is less expressive than the synchronous one.

$L^{\mathrm{S}}_{\mathrm{M,C,NO}}$ and $L^{\mathrm{A}}_{\mathrm{M,C,NO}}$ have the same expressive power. Easily, Boudol's encoding [2] can be used to prove that $L^{\mathrm{S}}_{\mathrm{M,C,NO}}$ is encodable in $L^{\mathrm{A}}_{\mathrm{M,C,NO}}$ with an encoding function that does not introduce divergence (trivially) and that enjoys full abstraction w.r.t. translated barbed equivalence (see [23]).

$L^{\mathrm{S}}_{\mathrm{P,C,PM}}$ and $L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ have the same expressive power. To prove that $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$ can be reasonably encoded in $L^{\mathrm{A}}_{\mathrm{P,C,PM}}$, it suffices to impose that the first name of every datum is a restricted channel used to unleash the continuation of the output prefix; conversely, every template starts with a new variable over which an acknowledgement is sent upon reception of the datum. This discipline is rendered by the following encoding:

$$[\![\, \overline{a}\langle\widetilde{b}\rangle.P \,]\!] \triangleq (\nu c)(\overline{a}\langle c, \widetilde{b}\rangle \mid c(\,).[\![\, P \,]\!]) \qquad \text{for } c \text{ fresh}$$

$$[\![\, a(\widetilde{T}).P \,]\!] \triangleq a(x, \widetilde{T}).(\overline{x}\langle\rangle \mid [\![\, P \,]\!]) \qquad \text{for } x \text{ fresh}$$

The encoding just presented is satisfying because it does not introduce divergence and enjoys full abstraction, as proved in the following theorem.

Theorem 5.1 *The encoding* $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,C,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ *does not introduce divergence; moreover,* $P \cong Q$ *if and only if* $[\![\, P \,]\!] \cong^{tr} [\![\, Q \,]\!]$.

Proof. See Appendix A. □

$L^{\mathrm{S}}_{\mathrm{P,C,NO}}$ **and** $L^{\mathrm{A}}_{\mathrm{P,C,NO}}$ **have the same expressive power.** This result is an easy corollary of the encodability of $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$ in $L^{\mathrm{A}}_{\mathrm{P,C,PM}}$: it suffices to restrict both the domain and the range of the encoding function to the sub-calculi of $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$ and $L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ with templates made up only by formal fields.

$L^{\mathrm{S}}_{\mathrm{P,D,PM}}$ **and** $L^{\mathrm{A}}_{\mathrm{P,D,PM}}$ **have the same expressive power.** To prove that $L^{\mathrm{S}}_{\mathrm{P,D,PM}}$ can be encoded in $L^{\mathrm{A}}_{\mathrm{P,D,PM}}$, consider the following translation:

$$[\![\, \langle b_1, \ldots, b_k\rangle.P \,]\!] \triangleq (\nu c)(\langle c, c, b_1, \ldots, b_k\rangle \;\mid\; (\ulcorner c \urcorner).[\![\, P \,]\!]) \qquad \text{for } c \text{ fresh}$$

$$[\![\, (T_1, \ldots, T_k).P \,]\!] \triangleq (x, y, T_1, \ldots, T_k).(\langle x\rangle \;\mid\; [\![\, P \,]\!]) \qquad \text{for } x \text{ and } y \text{ fresh}$$

Intuitively, data of length one in a translated term are ‘auxiliary’ messages used as acknowledgements (ack, for short), to activate the continuation of an output action. The translation of output prefixes guarantees that ‘actual’ data in the source term are translated to data whose length is at least two; this clear distinction ensures us that no interference between an ‘actual’ data exchange and an ‘auxiliary’ ack exchange can ever happen. Moreover, the fact that acks rely on restricted names rules out interferences between different acks.

Theorem 5.2 *The encoding* $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,D,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,D,PM}}$ *does not introduce divergence; moreover,* $P \cong Q$ *if and only if* $[\![\, P \,]\!] \cong^{tr} [\![\, Q \,]\!]$.

Proof. The proof is similar to the one for Theorem 5.1; all details are in Appendix A. □

$L^{\mathrm{S}}_{\mathrm{P,D,NO}}$ **and** $L^{\mathrm{A}}_{\mathrm{P,D,NO}}$ **have the same expressive power.** Let us define the following notation: $\langle \overset{k}{\ldots} \rangle$ denotes $\langle b_1, \ldots, b_k\rangle$, where the b_i's are any names; similarly, $(\overset{k}{\ldots})$ denotes $(x_1, \ldots, x_k)$, where the x_i's are pairwise and distinct names. Now, consider the following encoding of $L^{\mathrm{S}}_{\mathrm{P,D,NO}}$ in $L^{\mathrm{A}}_{\mathrm{P,D,NO}}$:

$$\llbracket \langle b_1, \ldots, b_k \rangle.P \rrbracket \triangleq \langle \overset{4k+1}{\ldots} \rangle \mid (\overset{4k+2}{\ldots}).(\langle b_1, b_1, b_1, b_1, \cdots, b_k, b_k, b_k, b_k \rangle \mid (\overset{4k+3}{\ldots}).\llbracket P \rrbracket)$$

$$\llbracket (x_1, \ldots, x_k).P \rrbracket \triangleq (\overset{4k+1}{\ldots}).(\langle \overset{4k+2}{\ldots} \rangle \mid (x_1, y_1, w_1, z_1, \cdots, x_k, y_k, w_k, z_k).(\langle \overset{4k+3}{\ldots} \rangle \mid \llbracket P \rrbracket))$$

for $y_1, w_1, z_1, \ldots, y_k, w_k, z_k$ fresh and pairwise distinct names and with the input variables in $(\overset{4k+1}{\ldots})$, $(\overset{4k+2}{\ldots})$ and $(\overset{4k+3}{\ldots})$ fresh for the continuation process. Intuitively, data of arity $4k$ within translated terms correspond to actual source data; data of arity $4k+1$, $4k+2$ and $4k+3$ are, instead, only used for synchronisation purposes. In particular, an exchange of arity $4k+1$ (that, from now on will be called *preliminary*) intuitively means "a datum of arity k is available"; an exchange of arity $4k+2$ (that, from now on will be called *initial*) intuitively means "a datum of arity k is going to be consumed"; finally, an exchange of arity $4k+3$ (that, from now on will be called *final*) intuitively means "a datum of arity k has been consumed". Consumption of a k-ary source level datum happens within a $4k$-ary exchange (that, from now on will be called *consumptive*).

Of course, it is easy to have interferences between the auxiliary data introduced by the encoding of different processes, but this does not create any problem since such data only depend on the length of the translated actions. Consider, e.g., the encoding of the $L^{\mathrm{S}}_{\mathrm{P,D,NO}}$-process $(x).P \mid \langle b \rangle \mid (y).Q \mid \langle c \rangle$ and the reduction that replaces x with b in P and y with c in Q. It is immaterial which of the two 5-ary 'preliminary' data (either the one from $\llbracket \langle b \rangle \rrbracket$ or the one from $\llbracket \langle c \rangle \rrbracket$) is accessed by $\llbracket (x).P \rrbracket$, since these are top-level asynchronous outputs and the names appearing in it are irrelevant. A similar argument holds also for the 'initial' 6-ary and the 'final' 7-ary data.

We believe that also this encoding enjoys full abstraction w.r.t. translated barbed equivalence; however, because of the interferences just discussed, we have still not been able to prove this result, though no counter-example against this conjecture has emerged yet. We leave this aspect for future work; for the moment, we prove the (not trivial) reasonableness of this encoding and argue that $L^{\mathrm{S}}_{\mathrm{P,D,NO}}$ and $L^{\mathrm{A}}_{\mathrm{P,D,NO}}$ have a comparable expressive power.

Lemma 5.3 *If* $\llbracket P \rrbracket \xrightarrow{\tau}{}^{n_p+n_i+n_c+n_f} Q$, *where* $n_p/n_i/n_c/n_f$ *are the number of preliminary/initial/consumptive/final steps in the reduction from* $\llbracket P \rrbracket$ *into* Q, *then* $n_p \geq n_i \geq n_c \geq n_f$.

Proof. Trivial, by construction of the encoding. □

Lemma 5.4 *Let* $\llbracket P \rrbracket \xrightarrow{\tau}{}^{n_p+n_i+n_c+n_f} Q$, *where* $n_p/n_i/n_c/n_f$ *are the number of preliminary/initial/consumptive/final steps in the reduction from* $\llbracket P \rrbracket$ *into* Q. *Then,*

$$\begin{aligned} Q \equiv (\nu\widetilde{n})(& \textstyle\prod_{h=1}^{n_p-n_i}((\overset{4k_h+2}{\ldots}).(\langle \overset{4k_h}{\ldots} \rangle \mid (\overset{4k_h+3}{\ldots}).\llbracket P^1_h \rrbracket) \mid \langle \overset{4k_h+2}{\ldots} \rangle \mid (\overset{4k_h}{\ldots}).(\langle \overset{4k_h+3}{\ldots} \rangle \mid \llbracket Q^1_h \rrbracket)) \mid \\ & \textstyle\prod_{j=1}^{n_i-n_c}(\langle \overset{4k_j}{\ldots} \rangle \mid (\overset{4k_j+3}{\ldots}).\llbracket P^2_j \rrbracket \mid (\overset{4k_j}{\ldots}).(\langle \overset{4k_j+3}{\ldots} \rangle \mid \llbracket Q^2_j \rrbracket)) \mid \\ & \textstyle\prod_{m=1}^{n_c-n_f}((\overset{4k_m+3}{\ldots}).\llbracket P^3_m \rrbracket \mid \langle \overset{4k_m+3}{\ldots} \rangle) \mid \llbracket R \rrbracket) \end{aligned}$$

where $\prod_{i=1}^{k} P_i$ *denotes* $P_1 \mid \ldots \mid P_k$, *if* $k > 0$, *and denotes* $\mathbf{0}$, *otherwise.*

Proof. Let $n = n_p + n_i + n_c + n_f$; the proof is by induction on n. The base case ($n = 0$) is trivial. For the inductive case, let $[\![P]\!] \xrightarrow{\tau}^n Q' \xrightarrow{\tau} Q$; by induction,

$$
\begin{aligned}
Q' \equiv (\nu \widetilde{n})(\; & \textstyle\prod_{h=1}^{n_p-n_i}((\overset{4k_h+2}{\ldots}).(\langle \overset{4k_h}{\ldots} \rangle \mid (\overset{4k_h+3}{\ldots}).[\![P_h^1]\!]) \mid \langle \overset{4k_h+2}{\ldots} \rangle \mid (\overset{4k_h}{\ldots}).(\langle \overset{4k_h+3}{\ldots} \rangle \mid [\![Q_h^1]\!])) \mid \\
& \textstyle\prod_{j=1}^{n_i-n_c}(\langle \overset{4k_j}{\ldots} \rangle \mid (\overset{4k_j+3}{\ldots}).[\![P_j^2]\!] \mid (\overset{4k_j}{\ldots}).(\langle \overset{4k_j+3}{\ldots} \rangle \mid [\![Q_j^2]\!])) \mid \\
& \textstyle\prod_{m=1}^{n_c-n_f}((\overset{4k_m+3}{\ldots}).[\![P_m^3]\!] \mid \langle \overset{4k_m+3}{\ldots} \rangle) \mid [\![R]\!] \,)
\end{aligned}
$$

where $n_p + n_i + n_c + n_f = n$. We consider two sub-cases, according to whether the step $Q' \xrightarrow{\tau} Q$ is preliminary or not.

- if $Q' \xrightarrow{\tau} Q$ is preliminary, then it must be (by construction) that $[\![R]\!] \xrightarrow{\tau} R'$, where

  $$
  \begin{aligned}
  R' \equiv (\nu \widetilde{c})(\; & (\overset{4k+2}{\ldots}).(\langle b_1, b_1, b_1, b_1, \ldots, b_k, b_k, b_k, b_k \rangle \mid (\overset{4k+3}{\ldots}).[\![R_1]\!]) \\
  & \mid \langle \overset{4k+2}{\ldots} \rangle \mid (x_1, y_1, w_1, z_1, \ldots, x_k, y_k, w_k, z_k).(\langle \overset{4k+3}{\ldots} \rangle \mid [\![R_2]\!]) \mid [\![R_3]\!])
  \end{aligned}
  $$

  Then,

  $$
  \begin{aligned}
  Q \equiv (\nu \widetilde{n}, \widetilde{c})(\; & \textstyle\prod_{h=1}^{n_p+1-n_i}((\overset{4k_h+2}{\ldots}).(\langle \overset{4k_h}{\ldots} \rangle \mid (\overset{4k_h+3}{\ldots}).[\![P_h^1]\!]) \mid \langle \overset{4k_h+2}{\ldots} \rangle \mid (\overset{4k_h}{\ldots}).(\langle \overset{4k_h+3}{\ldots} \rangle \mid [\![Q_h^1]\!])) \mid \\
  & \textstyle\prod_{j=1}^{n_i-n_c}(\langle \overset{4k_j}{\ldots} \rangle \mid (\overset{4k_j+3}{\ldots}).[\![P_j^2]\!] \mid (\overset{4k_j}{\ldots}).(\langle \overset{4k_j+3}{\ldots} \rangle \mid [\![Q_j^2]\!])) \mid \\
  & \textstyle\prod_{m=1}^{n_c-n_f}((\overset{4k_m+3}{\ldots}).[\![P_m^3]\!] \mid \langle \overset{4k_m+3}{\ldots} \rangle) \mid [\![R_3]\!] \,)
  \end{aligned}
  $$

  by letting $k_{n_p+1-n_i} = k$, $\langle \overset{4k_{n_p+1-n_i}}{\ldots} \rangle = \langle b_1, b_1, b_1, b_1, \ldots, b_k, b_k, b_k, b_k \rangle$, $P^1_{n_p+1-n_i} = R_1$, $(\overset{4k_{n_p+1-n_i}}{\ldots}) = (x_1, y_1, w_1, z_1, \ldots, x_k, y_k, w_k, z_k)$ and $Q^1_{n_p+1-n_i} = R_2$.

- Otherwise, it must be that either $\prod_{h=1}^{n_p-n_i} \cdots$, or $\prod_{j=1}^{n_i-n_c} \cdots$, or $\prod_{m=1}^{n_c-n_f} \cdots$ perform the τ-step, according to whether $Q' \xrightarrow{\tau} Q$ is initial, consumptive or final. We then work like in the previous case. □

Lemma 5.5 *If $[\![P]\!] \xrightarrow{\tau}^n Q$, then $P \xrightarrow{\tau}^{n_p} P'$ and $Q \xrightarrow{\tau}^{4n_p-n} [\![P']\!]$, where n_p is the number of preliminary steps in the reduction from $[\![P]\!]$ into Q.*

Proof. By induction on n; the base case is trivial. For the inductive case, let $[\![P]\!] \xrightarrow{\tau}^n Q' \xrightarrow{\tau} Q$ with

$$
\begin{aligned}
Q' \equiv (\nu \widetilde{n})(\; & \textstyle\prod_{h=1}^{n_p-n_i}((\overset{4k_h+2}{\ldots}).(\langle \overset{4k_h}{\ldots} \rangle \mid (\overset{4k_h+3}{\ldots}).[\![P_h^1]\!]) \mid \langle \overset{4k_h+2}{\ldots} \rangle \mid (\overset{4k_h}{\ldots}).(\langle \overset{4k_h+3}{\ldots} \rangle \mid [\![Q_h^1]\!])) \mid \\
& \textstyle\prod_{j=1}^{n_i-n_c}(\langle \overset{4k_j}{\ldots} \rangle \mid (\overset{4k_j+3}{\ldots}).[\![P_j^2]\!] \mid (\overset{4k_j}{\ldots}).(\langle \overset{4k_j+3}{\ldots} \rangle \mid [\![Q_j^2]\!])) \mid \\
& \textstyle\prod_{m=1}^{n_c-n_f}((\overset{4k_m+3}{\ldots}).[\![P_m^3]\!] \mid \langle \overset{4k_m+3}{\ldots} \rangle) \mid [\![R]\!] \,)
\end{aligned}
$$

by Lemma 5.4. It is then easy to see that $Q' \xrightarrow{\tau}^{4n_p-n} [\![P']\!]$, where

$$
\begin{aligned}
[\![P']\!] \equiv (\nu \widetilde{n})(\; & \textstyle\prod_{h=1}^{n_p-n_i}([\![P_h^1]\!] \mid [\![Q_h^1 \sigma_h^1]\!]) \mid \prod_{j=1}^{n_i-n_c}([\![P_j^2]\!] \mid [\![Q_j^2 \sigma_j^2]\!]) \mid \\
& \textstyle\prod_{m=1}^{n_c-n_f} [\![P_m^3]\!] \mid [\![R]\!] \,)
\end{aligned}
$$

for some substitutions σ_h^1's and σ_j^2's. If $Q' \xrightarrow{\tau} Q$ is not preliminary, then it cannot have been performed by $[\![R]\!]$; thus, $Q \xrightarrow{\tau}{}^{4n_p-n-1} [\![P']\!]$, i.e. $Q \xrightarrow{\tau}{}^{4n_p-(n+1)} [\![P']\!]$. Otherwise,

$$R \equiv (\nu\widetilde{c})(\langle\widetilde{b}\rangle.R_1 \mid (\widetilde{x}).R_2 \mid R_3)$$

for $|\widetilde{b}| = |\widetilde{x}| = k$. Now, consider

$$P'' \triangleq (\nu\widetilde{n},\widetilde{c})(\prod_{h=1}^{n_p-n_i}(P_h^1 \mid Q_h^1\sigma_h^1) \mid \prod_{j=1}^{n_i-n_c}(P_j^2 \mid Q_j^2\sigma_j^2) \mid \prod_{m=1}^{n_c-n_f} P_m^3 \mid R_1 \mid R_2\{^{\widetilde{b}}/_{\widetilde{x}}\} \mid R_3)$$

Trivially, $P' \xrightarrow{\tau} P''$ and $Q \xrightarrow{\tau}{}^{4n_p-n+3} [\![P'']\!]$, i.e. $Q \xrightarrow{\tau}{}^{4(n_p+1)-(n+1)} [\![P'']\!]$. □

Proposition 5.6 *The encoding $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,D,NO}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,D,NO}}$ is reasonable.*

Proof. By Lemma 5.5, operational correspondence is easy to prove; divergence freedom is a corollary of Lemmata 5.3 and 5.5; the remaining requirements are trivial. □

$L^{\mathrm{S}}_{\mathrm{M,D,NO}}$ is more expressive than $L^{\mathrm{A}}_{\mathrm{M,D,NO}}$.

Theorem 5.7 *There exists no reasonable encoding of $L^{\mathrm{S}}_{\mathrm{M,D,NO}}$ in $L^{\mathrm{A}}_{\mathrm{M,D,NO}}$.*

Proof. Consider the processes $P \triangleq \langle b\rangle.(x).\mathbf{if}\ x = b\ \mathbf{then}\ \Omega$ and $Q \triangleq (x).\langle a\rangle.\mathbf{if}\ x = a\ \mathbf{then}\ \Omega$, where Ω denotes a divergent process. Clearly, $P|Q$ does not diverge while, as we shall now prove, its encoding diverges. First, observe that in the evolution of $[\![P|Q]\!]$ to $[\![\mathbf{0}]\!]$ (that must happen, because of operational correspondence), both $[\![P]\!]$ and $[\![Q]\!]$ must perform an input and an output action: $[\![P]\!]$ must send b and $[\![Q]\!]$ must send a. Then, consider the sequence of actions performed by $[\![P]\!]$, say ρ, and its first input label, say $?n$; thus, $\rho = \rho_1 \cdot ?n \cdot \rho_2$. Notice that, by barb preservation, ρ_1 cannot be empty and must contain at least an output label, say $(\nu\widetilde{m})!m$; by Proposition 3.1(3,1), $[\![P]\!] \xrightarrow{?m}$ that, again by Proposition 3.1(3), implies that $[\![P]\!] \xrightarrow{\tau}$. By Proposition 4.5, $[\![\cdot]\!]$ cannot be reasonable.□

$L^{\mathrm{S}}_{\mathrm{M,D,PM}}$ is more expressive than $L^{\mathrm{A}}_{\mathrm{M,D,PM}}$. The impossibility proof relies on a preliminary Lemma.

Lemma 5.8 *Let $[\![\cdot]\!]$ be a reasonable encoding of $L^{\mathrm{S}}_{\mathrm{M,C,PM}}$ in $L^{\mathrm{A}}_{\mathrm{M,C,PM}}$. Then,*

1. *$[\![\langle b\rangle.P]\!] \xrightarrow{!b}$ and $[\![(\ulcorner b\urcorner)]\!] \xrightarrow{?b}$, with the input action relying on an actual input prefix;*
2. *$[\![\langle b\rangle.P]\!] \xrightarrow{!k}$ implies that $k = b$.*

Proof. Easy derivable from the more complex proof of Lemma 5.10 later on. □

Theorem 5.9 *There exists no reasonable encoding of $L^{\mathrm{S}}_{\mathrm{M,D,PM}}$ in $L^{\mathrm{A}}_{\mathrm{M,D,PM}}$.*

Proof. Consider $[\![\langle b\rangle.\Omega \mid (\ulcorner b\urcorner)]\!]$; by operational correspondence, such a process must evolve in $[\![\Omega]\!]$ that diverges, because of faithfulness. Since $[\![\langle b\rangle.\Omega]\!]$ cannot perform a τ-step, $[\![\langle b\rangle.\Omega]\!]$ must exhibit at least an input label in every trace. Moreover, by using Proposition 3.1(2) and Lemma 5.8(1), we can say that

$$[\![(\ulcorner b\urcorner)]\!] \xrightarrow{?b} \xRightarrow{\rho_1} \xrightarrow{!m} \xRightarrow{\rho_2} P_1 \qquad \text{and} \qquad [\![\langle b\rangle.\Omega]\!] \xrightarrow{!b} \xRightarrow{\rho_3} \xrightarrow{?m} \xRightarrow{\rho_4} P_2$$

where ρ_1 and ρ_2 are synchronisable with ρ_3 and ρ_4, resp., $(\nu\widetilde{n})(P_1 \mid P_2)$ is structurally equivalent to $[\![\Omega]\!]$, where $\widetilde{n} = \mathrm{B{\scriptsize N}}(\rho_1, \rho_2, \rho_3, \rho_4)$, and $?m$ is the first input in the trace from $[\![\langle b\rangle.\Omega]\!]$. Moreover, $m \neq b$ and the input $?m$ relies on an actual template (otherwise, by Proposition 3.1(1) and (3), $[\![\langle b\rangle.\Omega]\!] \xrightarrow{\tau}$ and $[\![\cdot]\!]$ would not be reasonable). Finally, by construction, ρ_3 is only made up by output labels that, by Lemma 5.8(2), are either of the form $!b$ or $(\nu d)!d$.

Now, choose $a \notin \{b, m\}$ and consider the process

$$P \triangleq \langle a\rangle \mid (\ulcorner \overline{a} \urcorner) \mid \langle b\rangle.\Omega \mid !\langle a\rangle \mid !\langle b\rangle$$

Clearly, P does not diverge while, as we shall now prove, $[\![P]\!]$ diverges. Let ρ'_i be ρ_i with a and b swapped, for $i = 1, \ldots, 4$. Now, synchronise

- $?a \cdot \rho'_1$ of $[\![(\ulcorner \overline{a} \urcorner)]\!]$ with $!a \cdot \rho'_3$ of $[\![\langle a\rangle]\!]$;
- $!m$ of the prosecution of $[\![(\ulcorner \overline{a} \urcorner)]\!]$ with $?m$ of $[\![\langle b\rangle.\Omega]\!]$;
- ρ'_2 of the prosecution of $[\![(\ulcorner \overline{a} \urcorner)]\!]$ with ρ_4 of the prosecution of $[\![\langle b\rangle.\Omega]\!]$; this can be freely done except when the action involves a or b. In such cases, synchronise
  - every $?a$ in ρ'_2 with one of the $!a$ from the encoding of $!\langle a\rangle$, and
  - every $?b$ in ρ_4 with one of the $!b$ from the encoding of $!\langle b\rangle$.

This yields a process containing the component

$$P' \triangleq P'_1 \mid P_2 \mid [\![!\langle a\rangle]\!] \mid [\![!\langle b\rangle]\!]$$

where P'_1 is P_1 with a and b swapped. Since $(\nu\widetilde{n})(P_1 \mid P_2)$ diverges, we also have that P' diverges: every time that P'_1 or P_2 need a $?a$ or a $?b$ to evolve, we can synchronise such actions with a corresponding $!a$ or $!b$ from the encoding of the replicated processes. □

$\mathcal{L}^{\mathrm{S}}_{\mathrm{M,C,PM}}$ is more expressive than $\mathcal{L}^{\mathrm{A}}_{\mathrm{M,C,PM}}$. Intuitively, communications in $\mathcal{L}^{\mathrm{S}}_{\mathrm{M,C,PM}}$ atomically verify the channel and the sent value (if pattern matching is involved) and simultaneously activate the continuation of the sending process. Thus, $\mathcal{L}^{\mathrm{A}}_{\mathrm{M,C,PM}}$ should provide the possibility of atomically verifying the channel and the sent value as well, but this excludes any information for synchronisation purposes.

The impossibility proof relies on a preliminary Lemma, that generalises Lemma 5.8.

Lemma 5.10 *Let a, b and c be pairwise distinct names and $[\![\cdot]\!]$ be a reasonable encoding of $\mathcal{L}^{\mathrm{S}}_{\mathrm{M,C,PM}}$ in $\mathcal{L}^{\mathrm{A}}_{\mathrm{M,C,PM}}$. Then,*

1. *$[\![\overline{a}\langle b\rangle.\overline{a}\langle c\rangle]\!] \xrightarrow{a!b}$ and $[\![a(\ulcorner b \urcorner).a(\ulcorner c \urcorner)]\!] \xrightarrow{a?b}$, with the input action relying on an actual input prefix;*
2. *$[\![\overline{a}\langle b\rangle.\overline{a}\langle c\rangle]\!] \xrightarrow{h!k}$ implies that $h = a$ and $k = b$;*
3. *$[\![a(x)]\!] \xrightarrow{a?b}$, with the input action relying on a formal input prefix;*
4. *$[\![\overline{a}\langle b\rangle.\overline{a}\langle c\rangle]\!] \xrightarrow{(\nu k)h!k}$ implies that $h = a$;*

or the same claims with a and b swapped in every label.

Proof.

1. By barb preservation, $[\![\overline{a}\langle b\rangle.\overline{a}\langle c\rangle]\!] \xrightarrow{(\nu\widetilde{m})n!m}$ and $[\![a(\ulcorner b \urcorner).a(\ulcorner c \urcorner)]\!] \xrightarrow{n?m}$. We prove that $\widetilde{m} = \emptyset$ and that $\{n, m\} = \{a, b\}$; let us reason by contradiction. If $\widetilde{m} \neq \emptyset$,

then the input label $n?m$ must come from a formal input action in $[\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!]$; thus, $[\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!] \xrightarrow{n?m'}$, for every name m', and this would imply that $[\![\, \overline{c}\langle b\rangle.\overline{c}\langle a\rangle \,]\!] \mid [\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!] \xrightarrow{\tau}$, if $n \neq a$, and $[\![\, \overline{a}\langle c\rangle.\overline{a}\langle b\rangle \,]\!] \mid [\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!] \xrightarrow{\tau}$, otherwise. Now, assume that $\{n, m\} \neq \{a, b\}$; we have three possible cases:

(a) $\{n, m\} \cap \{a, b\} = \emptyset$: pick up any $d \notin \{a, b, c, n, m\}$ and the permutation swapping a and d; then, $[\![\, \overline{d}\langle b\rangle.\overline{d}\langle c\rangle \,]\!] \xrightarrow{n!m}$ and so $[\![\, \overline{d}\langle b\rangle.\overline{d}\langle c\rangle \,]\!] \mid [\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!] \xrightarrow{\tau}$.

(b) $\{n, m\} \cap \{a, b\} = \{n\}$: if $n = b$ we work like in case (a); otherwise, pick up $d \notin \{a, b, c, n, m\}$, consider the permutation swapping b and d and conclude that $[\![\, \overline{a}\langle d\rangle.\overline{a}\langle c\rangle \,]\!] \mid [\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!] \xrightarrow{\tau}$.

(c) $\{n, m\} \cap \{a, b\} = \{m\}$: similar to case (b).

This proves that $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!]$ must exhibit either label $a!b$ or label $b!a$ and, consequently, that $[\![\, a(\ulcorner b \urcorner).a(\ulcorner c \urcorner) \,]\!]$ must exhibit either label $a?b$ or label $b?a$.

2. By point 1 of this Lemma, we have that $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \xrightarrow{a!b}$ and $[\![\, a(\ulcorner b \urcorner) \,]\!] \xrightarrow{a?b}$ (the other case is similar). By name invariance, we have that $[\![\, h(\ulcorner k \urcorner) \,]\!] \xrightarrow{h?k}$; this fact, together with the hypothesis $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \xrightarrow{h!k}$, would be in contradiction with reasonableness of $[\![\, \cdot \,]\!]$ whenever $h \neq a$ or $k \neq b$.

3. Consider now the process $\overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x)$; like before, $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \xrightarrow{(\nu\widetilde{k})h!k}$ and $[\![\, a(x) \,]\!] \xrightarrow{h?k}$ but, as we shall now prove, the input label $h?k$ must come from a formal input action in $[\![\, a(x) \,]\!]$. Indeed, if $\widetilde{k} \neq \emptyset$, the input must be formal. If $\widetilde{k} = \emptyset$, because of point 2 of this Proposition, it must be that $h = a$ and $k = b$, or vice versa; in both cases, the input cannot rely on an actual template, otherwise $[\![\, a(x) \,]\!]$ would have an infinite number of parallel components (one for every name n, since $\overline{a}\langle n\rangle.\overline{a}\langle c\rangle \mid a(x) \xrightarrow{\tau}$).

4. By contradiction, let $h \neq a$; then, we would have that $[\![\, a(x) \,]\!] \xrightarrow{h?k}$ (since the input is formal, see point 3 of this Proposition) and so $[\![\, \overline{d}\langle b\rangle.\overline{d}\langle c\rangle \,]\!] \mid [\![\, a(x) \,]\!] \xrightarrow{\tau}$, for any $d \notin \{a, b, c, h, k\}$. □

Theorem 5.11 *There exists no reasonable encoding of* $L^{\mathrm{S}}_{\mathrm{M,C,PM}}$ *in* $L^{\mathrm{A}}_{\mathrm{M,C,PM}}$.

Proof. From Lemma 5.10(1,2,4), we know that the process $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!]$ (with a, b and c pairwise distinct) must exhibit the label $a!b$ and, possibly, some bound outputs over a (the case for a and b swapped is similar). Moreover, by a reasoning similar to the proof of Theorem 5.7, we have at least a trace of $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!]$ with at least an input action; let $n?m$ be the first of such input actions. If $n = a$ and $m = b$, then, by Proposition 3.1(3), we would have that $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \xrightarrow{\tau}$. Then, it must be that $n \neq a$ or $m \neq b$; if we now prove that no bound output can be produced from $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!]$, then $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \mid [\![\, \overline{n}\langle m\rangle.\overline{n}\langle b\rangle \,]\!] \xrightarrow{\tau}$, since $[\![\, \overline{n}\langle m\rangle.\overline{n}\langle b\rangle \,]\!] \xrightarrow{n!m}$.

Let us consider the process $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x) \,]\!]$ and assume, for the sake of simplicity, that only one $a!b$ and one bound output action can be produced before $n?m$, i.e. $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!] \xrightarrow{a!b} P_1 \xrightarrow{(\nu k)a!k} P_2 \xrightarrow{n?m} P_3$ and $[\![\, a(x) \,]\!] \xrightarrow{a?b} Q_1 \xrightarrow{a?k} Q_2 \xrightarrow{(\nu\widetilde{m})n!m} Q_3$, for $(\nu k, \widetilde{m})(P_3 \mid Q_3) \Rightarrow [\![\, \overline{a}\langle c\rangle \,]\!]$. Notice that both the $a?b$ and the $a?k$ labels must have been originated from formal input actions: the first one because of Lemma 5.10(3), the second one because k was restricted in P_1. Now, consider the process $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x) \mid a(x) \,]\!]$ and the computation $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x) \mid a(x) \,]\!] \xrightarrow{\tau} P_1 \mid Q_1 \mid [\![\, a(x) \,]\!] \xrightarrow{\tau} (\nu k)(P_2 \mid Q_1 \mid Q_1\{^k/_b\})$. Clearly, $Q_1 \xrightarrow{a?\_}$ and $Q_1\{^k/_b\} \xrightarrow{a?\_}$, whereas $P_2 \stackrel{a!\_}{\not\longrightarrow}$. Then, by Definition 4.4(4).b, it must be that either $Q_1\{^k/_b\} \Rightarrow Q' \xrightarrow{a!k} Q'' \Rightarrow [\![\, a(x) \,]\!]$ or $Q_1 \Rightarrow Q' \xrightarrow{a!b} Q'' \Rightarrow [\![\, a(x) \,]\!]$; let us consider the second case, since the first one is similar. It is easy to prove that $Q' \equiv \overline{a}\langle b\rangle \mid Q''$; so, $Q_1 \Rightarrow \overline{a}\langle b\rangle \mid [\![\, a(x) \,]\!] \xrightarrow{\tau} Q_1$. Hence, by assuming that $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \,]\!]$ exhibits one bound output we have proved that $[\![\, \overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x) \mid a(x) \,]\!]$ diverges, whereas $\overline{a}\langle b\rangle.\overline{a}\langle c\rangle \mid a(x) \mid a(x)$ does not; thus, $[\![\, \cdot \,]\!]$ is not reasonable. $\square$

6 Concluding Assessment

We have studied the impact of synchrony in the eight communication primitives that arise when combining three common and useful programming features: arity of data, communication medium and presence of pattern matching. Our results have been summarised in Figure 1; we now briefly discuss them.

It is evident that polyadicity is the only feature that alone ensures fully abstract encodings of synchrony in asynchrony: this is related to the possibility of equipping polyadic data exchanges with auxiliary information (either a restricted channel that will be exploited for acknowledgement purposes, or the length of the data) used to synchronise the sending and the receiving process.

For monadic and channel-based communications, we have that absence of pattern matching makes synchrony encodable asynchronously, whereas presence of pattern matching rules out any such (reasonable) encoding. The problem is that pattern matching introduces the possibility of atomically matching the name transmitted in the communication; this leaves no space for any auxiliary synchronisation information.

Finally, monadic and dataspace-based communications are too weak to ensure any reasonable encoding: the problem is that there is no way to associate a datum with the process that emitted it. The latter fact entails that those languages that exploit such primitives (e.g., Ambient [7] or CCS [15]) cannot freely interchange their synchronous and asynchronous versions, though the latter ones are still Turing powerful [7,4].

Acknowledgements. We would like to thank the EXPRESS'06 reviewers for their positive attitude and for several fruitful comments that improved the presentation of the work.

A Omitted Proofs

To prove full abstraction results for the encodings of $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$ in $L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ and of $L^{\mathrm{S}}_{\mathrm{P,D,PM}}$ in $L^{\mathrm{A}}_{\mathrm{P,D,PM}}$, we rely on a well-know up-to proof-technique for weak (barbed) bisimulation [24], i.e. the *up-to expansion* technique. Intuitively, an expansion relates two weakly barbed equivalent processes by taking into account the number of their τ-steps; roughly speaking, if $P \gtrsim Q$, then $P \cong Q$ but P 'has more' τ-steps than Q.

However, we are interested in proving relations closed only under translated contexts for a fixed encoding function $[\![\, \cdot \,]\!] : \mathcal{L}_1 \to \mathcal{L}_2$; moreover, we want to precisely count the difference between the τ-steps of the processes related by an expansion. Thus, we shall slightly adapt the definition of the expansion preorder [1], as follows. There, we use $\simeq^{tr}$ to denote the strong version of translated barbed equivalence, i.e. the relation defined like in Definition 4.2 with $\downarrow$ in place of $\Downarrow$ and $\xrightarrow{\tau}$ in place of $\Rightarrow$ everywhere.

Definition A.1 [Translated one-step expansion] Given an encoding function $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$, $\gtrsim^{tr}$ is the largest preorder between $\mathcal{L}_2$-processes such that, whenever $P \gtrsim^{tr} Q$, it holds that

- $P \xrightarrow{\tau} P'$ implies that either $Q \xrightarrow{\tau} Q'$ for some Q' such that $P' \gtrsim^{tr} Q'$ or $P' \simeq^{tr} Q$;
- $P \xrightarrow{\alpha} P'$, where α can be consumed by a translated process, implies that $Q \xrightarrow{\alpha} Q'$ for some Q' such that $P' \gtrsim^{tr} Q'$;
- $Q \xrightarrow{\tau} Q'$ implies that either $P \xrightarrow{\tau} P'$ for some P' such that $P' \gtrsim^{tr} Q'$ or $P \xrightarrow{\tau}\xrightarrow{\tau} P'$ for some P' such that $P' \simeq^{tr} Q'$;
- $Q \xrightarrow{\alpha} Q'$, where α can be consumed by a translated process, implies that either $P \xrightarrow{\alpha} P'$ for some P' such that $P' \gtrsim^{tr} Q'$ or $P \xrightarrow{\tau}\xrightarrow{\alpha} P'$ for some P' such that $P' \simeq^{tr} Q'$.

When notationally convenient, $P \gtrsim^{tr} Q$ could be also written as $Q \lesssim^{tr} P$. The first crucial property of $\gtrsim^{tr}$ is that it preserves and reflects divergence, as proved below.

Proposition A.2 *If $P \gtrsim^{tr} Q$ then P diverges if and only if Q diverges.*

Proof. The proof simply follows from Definition A.1, once observed that $\simeq^{tr}$ preserves and reflects divergence. □

The second crucial property of $\gtrsim^{tr}$ is that translated barbed bisimilarity and equivalence up-to translated expansion (defined below) coincide with translated barbed bisimilarity and equivalence, respectively.

Definition A.3 Fix an encoding $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$.

- A symmetric relation $\mathfrak{R}$ between $\mathcal{L}_2$-processes is a *translated barbed bisimulation up-to expansion* if, for every $(P, Q) \in \mathfrak{R}$, it holds that
  - (i) $P \downarrow_o^{tr}$ implies $Q \Downarrow_o^{tr}$, and
  - (ii) $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow Q'$, for some Q' such that $P' \gtrsim^{tr} \mathfrak{R} \lesssim^{tr} Q'$.

  Translated barbed bisimilarity up-to expansion, $\dot{\cong}^{\gtrsim^{tr}}$, is the largest translated barbed bisimulation up-to expansion.
- P and Q are *translated barbed equivalent up-to expansion*, written $P \cong^{\gtrsim^{tr}} Q$, if and only if $C[P] \dot{\cong}^{\gtrsim^{tr}} C[Q]$, for every context $C[\cdot]$ resulting from the translation of a $\mathcal{L}_1$-context via $[\![\cdot]\!]$ extended with $[\![\,[\cdot]\,]\!] \triangleq [\cdot]$.

Proposition A.4 (i) *If $P \dot{\cong}^{\gtrsim^{tr}} Q$, then $P \dot{\cong}^{tr} Q$;*
(ii) *if $P \cong^{\gtrsim^{tr}} Q$, then $P \cong^{tr} Q$.*

Proof. For the first claim, it suffices to prove that relation $\{(P, Q) \ : \ P \gtrsim^{tr} P' \dot{\cong}^{tr} Q' \lesssim^{tr} Q\}$ is a translated barbed bisimulation; this follows straightforwardly from Definitions A.1 and 4.2. The second claim is an easy corollary of the first one. □

We are now ready to prove full abstraction and divergence freedom for the encodings presented in the body of the paper. They rely on a slightly enhanced version of the operational correspondence property presented in Definition 4.4.

Lemma A.5 *Consider the encoding $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,C,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ and a $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$-process P. Then,*

1. *$P \xrightarrow{\tau} P'$ implies that $[\![P]\!] \overset{\tau}{\Longrightarrow} [\![P']\!]$;*
2. *$[\![P]\!] \xrightarrow{\tau} Q$ implies that $P \xrightarrow{\tau} P'$ for some P' such that $Q \gtrsim^{tr} [\![P']\!]$.*

Proof. Both claims are proved by a simple induction over the inference of the $\xrightarrow{\tau}$ in their premise. For the second claim, it is crucial to note that $(\nu c)(\bar{c}\langle\rangle \mid c().P) \gtrsim^{tr} P$, whenever $c \notin \mathrm{F{\scriptstyle N}}(P)$. $\square$

Theorem A.6 *Consider the encoding $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,C,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ and a $L^{\mathrm{S}}_{\mathrm{P,C,PM}}$-process P. Then, $P \dot{\cong}^{tr} [\![P]\!]$.*

Proof. We prove that $\{(P, [\![P]\!])\}$ is a translated barbed bisimulation up-to translated expansion. By definition of $[\![\cdot]\!]$, it holds that $P \downarrow_o$ if and only if $[\![P]\!] \downarrow_o$. If $P \xrightarrow{\tau} P'$ then, by Proposition A.5(1), $[\![P]\!] \overset{\tau}{\Longrightarrow} [\![P']\!]$. Finally, if $[\![P]\!] \xrightarrow{\tau} Q$, then, by Proposition A.5(2), $P \xrightarrow{\tau} P'$ and $Q \gtrsim^{tr} [\![P']\!]$; because of Proposition A.4, this suffices to conclude. $\square$

Corollary A.7 (completing Theorem 5.1) *The encoding $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,C,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,C,PM}}$ enjoys full abstraction w.r.t. translated barbed equivalence and does not introduce divergence.*

Proof. Full abstraction w.r.t. translated barbed equivalence easily holds by Theorem A.6 and transitivity of $\dot{\cong}^{tr}$. The fact that $[\![\cdot]\!]$ does not introduce divergence is proved by building up a divergent computation for every P such that $[\![P]\!] \xrightarrow{\tau}{}^{\omega}$. This is an easy task, thanks to Lemma A.5(2): indeed, if $[\![P]\!]$ diverges, then $[\![P]\!] \xrightarrow{\tau} Q$, for some Q that diverges. Then, we can find a P' such that $P \xrightarrow{\tau} P'$ and $Q \gtrsim^{tr} [\![P']\!]$; because of Proposition A.2, also $[\![P']\!]$ diverges. Thus, P reduces to a process whose encoding diverges; by iterating this reasoning arbitrarily, we can conclude that also P diverges. $\square$

We now prove similar results for the encoding of $L^{\mathrm{S}}_{\mathrm{P,D,PM}}$ in $L^{\mathrm{A}}_{\mathrm{P,D,PM}}$; however, since the encoding changes the barbs of any translated process (remember that every source language input/output of arity k is translated in a $(k+2)$-ary input/output), an analogous of Theorem A.6 cannot hold. This makes the proof of full abstraction slightly more complex; on the contrary, divergence freedom is proved exactly in the same way (thus, we shall not mention it anymore).

Lemma A.8 *Consider the encoding $[\![\cdot]\!] : L^{\mathrm{S}}_{\mathrm{P,D,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,D,PM}}$ and a $L^{\mathrm{S}}_{\mathrm{P,D,PM}}$-process P. Then,*

1. *$P \xrightarrow{\tau} P'$ implies that $[\![P]\!] \overset{\tau}{\Longrightarrow} [\![P']\!]$;*
2. *$[\![P]\!] \xrightarrow{\tau} Q$ implies that $P \xrightarrow{\tau} P'$ for some P' such that $Q \gtrsim^{tr} [\![P']\!]$.*

Proof. Like the proof of Lemma A.5, but relying on the fact that $(\nu c)(\langle c\rangle \mid (\ulcorner c \urcorner).P) \gtrsim^{tr} P$, whenever $c \notin \mathrm{F{\scriptstyle N}}(P)$. Indeed, $(\nu c)(\langle c\rangle \mid (\ulcorner c \urcorner).P) \overset{?c}{\not\longrightarrow}$, whereas $(\nu c)(\langle c\rangle \mid (\ulcorner c \urcorner).P) \xrightarrow{(\nu c)!c}$;

however, there exists no translated process able to exhibit a trace $\rho\cdot?c$ without having $c \in \mathrm{BN}(\rho)$. Thus, $(\nu c)(\langle c\rangle \mid (\ulcorner c\urcorner).P) \downarrow^{tr}_{o}$ cannot hold and this suffices to conclude. $\square$

Theorem A.9 *Consider the encoding* $\llbracket\cdot\rrbracket : L^{\mathrm{S}}_{\mathrm{P,C,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,C,PM}}$*; then,* $\llbracket\cdot\rrbracket$ *is fully abstract w.r.t. translated barbed bisimilarity, i.e.* $P \dot{\cong} Q$ *if and only if* $\llbracket P\rrbracket \dot{\cong}^{tr} \llbracket Q\rrbracket$*.*

Proof. For the "only if" part, we prove that

$$\mathfrak{R} \triangleq \{(\llbracket P\rrbracket, \llbracket Q\rrbracket) \ : P \dot{\cong} Q\}$$

is a translated barbed bisimulation up-to $\gtrsim^{tr}$. Let $\llbracket P\rrbracket \downarrow^{tr}_{OUT^k}$ (the case for $\llbracket P\rrbracket \downarrow^{tr}_{IN^k}$ is similar); by definition of $\llbracket\cdot\rrbracket$, $k = h{+}2$ and $P \downarrow^{tr}_{OUT^h}$; then, $Q \Downarrow^{tr}_{OUT^h}$ and hence $\llbracket Q\rrbracket \Downarrow^{tr}_{OUT^k}$. Let $\llbracket P\rrbracket \xrightarrow{\tau} P_1$; then, by Lemma A.8(2), $P \xrightarrow{\tau} P'$ and $P_1 \gtrsim^{tr} \llbracket P'\rrbracket$. Then, $Q \Rightarrow Q'$ and $P' \dot{\cong} Q'$; by Lemma A.8(1), this implies that $\llbracket Q\rrbracket \Rightarrow \llbracket Q'\rrbracket$ and $(P_1, \llbracket Q'\rrbracket) \in \mathfrak{R}$ up-to $\gtrsim^{tr}$.

For the "if" part, we prove that

$$\mathfrak{R} \triangleq \{(P,Q) \ : \llbracket P\rrbracket \dot{\cong}^{tr} \llbracket Q\rrbracket\}$$

is a barbed bisimulation. Let $P \downarrow^{tr}_{OUT^k}$ (the case for $P \downarrow^{tr}_{IN^k}$ is similar); then, $\llbracket P\rrbracket \downarrow^{tr}_{OUT^{k+2}}$ and, hence, $\llbracket Q\rrbracket \Downarrow^{tr}_{OUT^{k+2}}$. Let $\llbracket Q\rrbracket \xrightarrow{\tau}{}^{n} \downarrow^{tr}_{OUT^{k+2}}$; by induction on n, we now prove that $Q \Downarrow^{tr}_{OUT^k}$, as desired. The base case is trivial: $\llbracket Q\rrbracket \downarrow^{tr}_{OUT^{k+2}}$ implies $Q \downarrow^{tr}_{OUT^k}$. For the inductive case, let $\llbracket Q\rrbracket \xrightarrow{\tau} R \xrightarrow{\tau}{}^{n} \downarrow^{tr}_{OUT^{k+2}}$; by Lemma A.8(2), $Q \xrightarrow{\tau} Q'$ and $R \gtrsim^{tr} \llbracket Q'\rrbracket$. By definition of $\gtrsim^{tr}$, $\llbracket Q'\rrbracket \xrightarrow{\tau}{}^{m} \downarrow^{tr}_{OUT^{k+2}}$, for $m \leq n$, that, by induction, allows us to conclude.

Finally, let $P \xrightarrow{\tau} P'$; by Lemma A.8(1), this implies that $\llbracket P\rrbracket \Rightarrow \llbracket P'\rrbracket$ that in turn entails $\llbracket Q\rrbracket \Rightarrow R$ and $\llbracket P'\rrbracket \dot{\cong}^{tr} R$. Let $\llbracket Q\rrbracket \xrightarrow{\tau}{}^{n} R$; by induction on n we now prove that $R \gtrsim^{tr} \llbracket Q'\rrbracket$, for some Q' such that $Q \Rightarrow Q'$. This suffices to conclude, since $(P', Q') \in \mathfrak{R}$ (thanks to $\gtrsim^{tr} \subseteq \dot{\cong}^{tr}$ and transitivity of $\dot{\cong}^{tr}$). The base case is trivial, since $R = \llbracket Q\rrbracket$. For the inductive case, let $\llbracket Q\rrbracket \xrightarrow{\tau} R' \xrightarrow{\tau}{}^{n} R$; by Lemma A.8(2), $Q \xrightarrow{\tau} Q''$ and $R' \gtrsim^{tr} \llbracket Q''\rrbracket$. By definition of $\gtrsim^{tr}$, $\llbracket Q''\rrbracket \xrightarrow{\tau}{}^{m} R$, for $m \leq n$, that, by induction and transitivity of $\gtrsim^{tr}$, allows us to conclude. $\square$

Corollary A.10 (completing Theorem 5.2) *The encoding* $\llbracket\cdot\rrbracket : L^{\mathrm{S}}_{\mathrm{P,D,PM}} \longrightarrow L^{\mathrm{A}}_{\mathrm{P,D,PM}}$ *enjoys full abstraction w.r.t. translated barbed equivalence and does not introduce divergence.*

References

[1] S. Arun-Kumar and M. Hennessy. An efficiency preorder for processes. *Acta Informatica*, 29(8):737–760, 1992.

[2] G. Boudol. Asynchrony and the π-calculus (note). Rapport de Recherche 1702, INRIA Sophia-Antipolis, May 1992.

[3] A. Brown, C. Laneve, and G. Meredith. πduce: a process calculus with native XML datatypes. In *Proc. of 2nd Int. Workshop on Services and Formal Methods*, volume 3670 of *LNCS*. Springer, 2005.

[4] N. Busi, R. Gorrieri, and G. Zavattaro. A process algebraic view of Linda coordination primitives. *Theoretical Computer Science*, 192(2):167–199, 1998.

[5] D. Cacciagrano and F. Corradini. On synchronous and asynchronous communication paradigms. In *Proc. of ICTCS'01*, number 2202 in LNCS, pages 256–268. Springer, 2001.

[6] D. Cacciagrano, F. Corradini, and C. Palamidessi. Separation of synchronous and asynchronous communication via testing. In *Proc. of EXPRESS*, ENTCS. Elsevier, 2005.

[7] L. Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[8] G. Castagna, R. De Nicola, and D. Varacca. Semantic subtyping for the π-calculus. In *Proc. of LICS*, pages 92–101. IEEE Computer Society, 2005.

[9] R. De Nicola, D. Gorla, and R. Pugliese. On the expressive power of KLAIM-based calculi. *Theoretical Computer Science*, 356(3):387–421, 2006.

[10] R. De Nicola and M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[11] C. Fournet and G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of POPL '96*, pages 372–385. ACM, Jan. 1996.

[12] D. Gelernter. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

[13] D. Gorla. On the relative expressive power of asynchronous communication primitives. In *Proc. of FoSSaCS'06*, volume 3921 of *LNCS*, pages 47–62. Springer, 2006.

[14] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proc. of ECOOP '91*, volume 512 of *LNCS*, pages 133–147. Springer, 1991.

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

[16] R. Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.

[17] R. Milner, J. Parrow, and J. Walker. A Calculus of Mobile Processes, I and II. *Information and Computation*, 100(1):1–40, 41–77, 1992.

[18] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

[19] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *Proc. of ICALP'95*, volume 944 of *LNCS*, pages 648–659. Springer, 1995.

[20] C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

[21] J. Parrow. An introduction to the pi-calculus. In *Handbook of Process Algebra*, pages 479–543. Elsevier Science, 2001.

[22] B. C. Pierce and D. N. Turner. Pict: A programming language based on the pi-calculus. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Foundations of Computing. MIT Press, May 2000.

[23] P. Quaglia and D. Walker. On synchronous and asynchronous mobile processes. In *Proceedings of FoSSaCS 2000*, volume 1784 of *LNCS*, pages 283–296. Springer, 2000.

[24] D. Sangiorgi and R. Milner. The problem of "weak bisimulation up to". In *Proc. of CONCUR*, volume 630 of *LNCS*, pages 32–46. Springer, 1992.

[25] G. Tel. *Introduction to distributed algorithms*. Cambridge University Press, 1995.