Information and Computation, 206(8):931–952.

Comparing Communication Primitives via their Relative Expressive Power

Daniele Gorla
Dipartimento di Informatica
Università di Roma "La Sapienza"

Abstract

In this paper, we study sixteen communication primitives, arising from the combination of four useful programming features: *synchronism* (synchronous vs asynchronous primitives), *arity* (monadic vs polyadic data), *communication medium* (message passing vs shared dataspaces) and *pattern-matching*. Some of these primitives have already been used in at least one language which has appeared in the literature; however, to reason uniformly on such primitives, we plug them into a common framework based on the π-calculus. By means of possibility/impossibility of 'reasonable' encodings, we compare every pair of primitives to obtain a hierarchy of languages based on their relative expressive power.

Contents

1 Introduction

In the last 25 years, several languages and formalisms for distributed and concurrent systems have appeared in the literature. Some of them (e.g., CCS [28] and the π-calculus [45]) are mostly mathematical models, mainly used to rigorously reason on concurrent systems; other ones (e.g., LINDA [22]) are closer to actual programming languages and are mainly focused on issues like usability and flexibility. As a consequence, the former ones are usually minimal, whereas the latter ones provide more sophisticated and powerful programming constructs.

Despite their differences, there are, however, some basic features that are implemented to some extent in all these languages. Roughly speaking, these features can be described as the possibility of having different *execution threads* (or *processes*) that *run concurrently* by interacting via some form of *communication*. At least at a first glance, the last feature (i.e., inter-process communication) has yielded the highest variety of proposals. These arose from the possibility of having synchronous/asynchronous primitives, monadic/polyadic data, first-order/higher-order values, dataspace-based/channel-based communication media, local/remote exchanges (whenever processes are explicitly distributed, like in [14, 17]), built-in pattern-matching mechanisms, point-to-point/broadcasting primitives, and so on. The aim of this work is to rigorously study some of these proposals and to organize them in a clear hierarchy, based on their expressive power. Hopefully, our results should help to understand the peculiarities of every communication primitive and, as a consequence, they could be exploited to choose the 'right' primitive when designing new languages and formalisms.

Among the features mentioned above, we focus on *synchronism*, *arity of data*, *communication medium* and possibility of *pattern-matching*. The expressiveness of the omitted features has been already dealt with elsewhere [17, 19, 43]; we leave as future work the integration of these results in our framework. Notice that we studied pattern-matching because it is nowadays becoming more and more important, especially in languages that deal with complex data like XML [1, 6, 15]. However, for the sake of simplicity, we consider here a very basic form of pattern-matching, that only checks for name equality while retrieving a datum; the rigorous study of more flexible and powerful mechanisms (e.g., those in [18]) is left for future work.

By combining the four features chosen, we obtain sixteen communication primitives, some of which have already been used elsewhere, e.g. in [5, 14, 15, 17, 22, 26, 30]. However, to reason uniformly on such primitives, we plug them in a common framework based on the π-calculus; we choose the π-calculus because nowadays it is one of the best-established workbenches for theoretical reasoning on concurrent systems.

Assessing language expressiveness. Several techniques can be exploited to study the expressive power of a programming language; of course, different techniques have different merits and yield different results. A possible approach is based on the *absolute* expressive power of a language and it consists in studying which problems can be solved in the language. For example, if one is interested in the computational power of a language, the natural problem is the implementability of any Turing complete formalism; however, since most languages allow such an implementation, this problem is not adequate to compare different languages. In particular, all the languages we are going to consider are Turing complete (it is easy to encode the Turing complete language $\mathbf{L}_0$ from [9] into the bottom element of our hierarchy). Thus, the crucial aspect of this approach is the identification of more sophisticated problems that can be solved in a language under some conditions that cannot be met

by any solution in another language. For example, in [13, 35] several variants of the π-calculus have been compared by showing a problem (namely, leader election in [35] and matching systems in [13]) that can be solved in one variant and not in another one.

However, the identification of a problem solvable in a language but not in another one is usually very difficult. Thus, another interesting approach to comparing two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ consists in studying their *relative* expressive power, by trying to encode one in the other and studying the properties of the encoding function. This is the approach we shall follow in this paper and it is very appealing for at least two reasons. Firstly, it is a natural way to show how the key features of a language can be rendered in the other one, or why this is not possible. Secondly, it would allow us to also carry out quantitative measures of language expressiveness: we could consider aspects like the size and the complexity of the encoding of a term with respect to the source term and, consequently, quantitatively assess the encoding proposed.

Of course, the encoding function must preserve the 'essence' of the translated term, i.e. to be meaningful an encoding should not change the functionalities and the behaviours of source terms. This requirement can be formalized in different ways (see [34, 37] for a good discussion). A first possibility (usually called *semantical equivalence*) is to fix an equivalence, say $\sim$, and require that the encoding maps every source term into a $\sim$-equivalent target term. The main problem behind this property is that it can only be investigated when the considered languages are very similar, i.e. whenever they share some notion of equivalence. This property can be weakened by choosing an abstract semantic theory $\mathcal{S}$ and considering the equivalences generated by $\mathcal{S}$ in $\mathcal{L}_1$ and $\mathcal{L}_2$, say $\sim_1^{\mathcal{S}}$ and $\sim_2^{\mathcal{S}}$. Then, the so called *full abstraction* property requires that the encoding respects $\mathcal{S}$, i.e. it maps $\sim_1^{\mathcal{S}}$-equivalent terms into $\sim_2^{\mathcal{S}}$-equivalent terms and vice versa.

Semantical equivalence and full abstraction are both defined with respect to a fixed notion of equivalence (viz., $\sim$ or $\mathcal{S}$). In concurrency, we have an incredibly wide range of equivalences; thus, fixing one or another is highly debatable. Moreover, full abstraction is mostly focused on the discriminating power of the equivalences; this can be useful, e.g., if one uses the encoding to exploit powerful proof-techniques for $\sim_2^{\mathcal{S}}$ to prove $\sim_1^{\mathcal{S}}$-equivalences, but it is not deeply related to expressiveness issues.

Indeed, if one is interested in comparing what different languages can implement, we think that it is more natural to fix a set of 'reasonable' properties that every encoding should satisfy; in this way, we can prove that $\mathcal{L}_1$ is strictly more powerful than $\mathcal{L}_2$ by showing that $\mathcal{L}_1$ can reasonably encode $\mathcal{L}_2$, whereas no such encoding of $\mathcal{L}_1$ in $\mathcal{L}_2$ exists. This is a well-established approach to proving impossibility results in concurrency theory [12, 13, 16, 19, 35, 39, 46, 49], though no common agreement has yet been reached on which properties make an encoding 'reasonable' [33]. Moreover, the requirements identified for every impossibility result are intentionally minimal, in the sense that each requirement is strictly needed to prove the result. This makes the impossibility result very strong and informative, but makes the requirements not suitable for properly evaluating encodability results.

For this reason, we identify a set of criteria suitable for both encodability and impossibility results, by finding a compromise between 'minimality' (typical of impossibility results) and 'maximality' (typical of encodability results, where one wants to show that the encoding satisfies as many properties as possible). In particular, we require that a reasonable encoding is *compositional* (i.e., the encoding of a compound term is defined by combining the encoding of its sub-terms) and *name invariant* (i.e., it translates source terms that only differ in their names into target terms with the same property). Moreover, it must preserve and reflect *divergence* and the possibility of *interacting*

with an external observer. Finally, it must be *operationally corresponding*, in the sense that source and target computations must correspond. Notice that formulations of all these requirements have already appeared in the literature when developing encodings or proving impossibility results. However, to the best of our knowledge, they have never been grouped together to uniformly build up a hierarchy of languages.

Main contributions. A first contribution of this paper is the definition of a set of criteria that allows both the evaluation of encodings and of proving impossibility results; moreover, the proof-technique developed for our impossibility results is new, relatively simple and usable in frameworks different from the present one. A second contribution consists in giving a number of statements on communication primitives; some of them are common sense, but here we can rigorously state and prove them; some other ones are more surprising and, to the best of our knowledge, this is the first work that points them out. We now briefly sum up the ones which, in our opinion, are the most important achievements (for a full picture, see Figure 1 in Section 3).

1. Our results show that the communication paradigm underlying LINDA [22] (asynchronous, polyadic, dataspace-based and with pattern-matching) is at the top of the hierarchy; not incidentally, LINDA's paradigm has been used in actual programming languages [3, 20]. On the opposite extreme, we have the communication paradigm used in Mobile Ambients [14] (asynchronous, monadic, dataspace-based but without pattern-matching); such a paradigm is very simple but also very poor and, not incidentally, the expressive power of Mobile Ambients mostly arises from the mobility primitives [10]. Strictly in the middle, we find the π-calculus (channel-based and without pattern-matching), in its synchronous/asynchronous and monadic/polyadic versions. This result stresses the fact that the π-calculus is a good compromise between expressiveness and simplicity.

2. As a further contribution, we prove that the untyped polyadic π-calculus is strictly more expressive than the monadic one; thus, the introduction of types in the polyadic π-calculus reduces its expressive power. A posteriori, this fact justifies the use of type-systems [29, 42, 47, 48] to obtain a fragment of the polyadic π-calculus that can be reasonably translated in the monadic π-calculus.

3. We also discuss the interplay between synchrony and channels: in particular, we show that, when communications exploit channels (or features that can encode them), the impact of synchrony is irrelevant, in the sense that synchronous primitives can be reasonably encoded via their asynchronous counterpart. This result allows us to freely implement the primitives asynchronously, that usually poses fewer implementation problems.

4. Our results show that, among the four features studied (synchrony, polyadicity, channels and pattern matching), the most powerful one is the use of channels: by exploiting only (possibly restricted) channels, we can encode every other feature in isolation. This result further supports the choice of the asynchronous π-calculus as the reference calculus for mobility: it is 'small', quite powerful and easily implementable [40].

5. Finally, we also show that the introduction of pattern-matching always increases the expressive power of a language. Indeed, a language with pattern-matching can always *atomically* check more names than the corresponding language without pattern-matching; as already pointed

out in [13], the possibility of checking more names cannot be rendered without changing the behaviour of the term (more specifically, without introducing divergence).

Of course, all our results strongly rely on the reasonableness criteria mentioned above. By changing the set of criteria, also the lattice of languages is likely to change. However, we believe that our results are meaningful because we think that our criteria are natural and acceptable. It has also to be said that our investigation is not the first one that compares different forms of communication in the π-calculus: almost every variant present in the literature comes equipped with a comparison against the original formulation of [30]. The problem is that (almost) every paper assumes a different criterion for evaluating its results. For example, in [29] the encoding of the polyadic π-calculus in its monadic version is given by only arguing on its correctness; in [42, 48] it has been shown that such an encoding enjoys full abstraction with respect to typed barbed congruence. Similarly, in [5, 26] there are encodings of the synchronous in the asynchronous π-calculus. The former one is proved correct by showing a full abstraction result with respect to some notion of weak bisimulation. The second one is proved correct by showing an adequacy result with respect to a Morris-like equivalence; moreover, such result is also proved sound with respect to weak barbed congruence, may testing and fair testing restricted to encoded contexts [11] and with respect to typed barbed congruence [41]. However, [12] proves that there exists no compositional and must preserving encoding of the synchronous in the asynchronous π-calculus. This fact emphasizes the difficulties behind the choice of the equivalence when trying to establish full abstraction properties.

Overview of the paper. In Section 2, we present the family of sixteen concurrent languages arising from the combination of the four features studied. In Section 3, we present the criteria that, in our opinion, an encoding should satisfy to be a reasonable means for language comparison; there, we also sum up in detail the results of the paper, that are proved in Sections 4 and 5. For the sake of presentation, we start in Section 4 by restricting our study to synchronous communication primitives; then, in Section 5, we include in the resulting hierarchy also the asynchronous versions of these primitives. Section 6 concludes the paper by also touching upon related work.

This paper merges together the results contained in [23, 24] to obtain a full hierarchy of the sixteen languages studied. With respect to [23], we improve some of the encodings and we also consider synchronous primitives; with respect to [24], we place in the right place of the hierarchy the synchronous primitives and rectify some wrong results. In both cases, we give more details on some technical proofs and consider more liberal encodability criteria (that strengthen our impossibility results). For the sake of uniformity, we formulate both the encodability and the impossibility results in terms of reasonableness. Full abstraction results are only hinted at here; some of them are rigorously proved in [24].

2 A Family of Process Calculi

We now define the syntax, the operational and the behavioural semantics of our calculi. In doing this, we shall strongly rely on well-known notions developed for the π-calculus, our reference framework, and simply adapt them (whenever needed) to cope with the different features of our languages.

2.1 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

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

The different languages will be obtained by plugging into this basic syntax a proper definition for input prefixes (*IN*) and output processes (*OutProc*). As usual, $P|Q$ denotes the parallel composition of processes; $(\nu n)P$ restricts to P the visibility of n; finally, **if** $n = m$ **then** P **else** Q and $*P$ are the standard constructs for conditional evolution and process replication. Notationally, **if** $n = m$ **then** P denotes a conditional construct with a terminated else-branch; moreover, trailing occurrences of $\mathbf{0}$ will be systematically omitted. We have intentionally chosen a very simple form of recursive construct, i.e. process replication; more sophisticated constructs can be exploited without changing our results (that do not crucially rely on this aspect).

In this paper, we study the possible combinations of four features for communications: *synchronism* (synchronous vs. asynchronous communication primitives), *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 $\Lambda_{s,a,m,p}$, whose generic element is 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.

Thus, the full syntax of every language is obtained from the following productions:

$$\begin{array}{rlll}
L_{\text{A}\cdots}: & \mathit{OutProc} ::= \mathit{OUT} & & \\
L_{\text{S}\cdots}: & \mathit{OutProc} ::= \mathit{OUT}.P & & \\
L_{\text{-,M,D,NO}}: & P, Q, R ::= \ldots & \mathit{IN} ::= (x) & \mathit{OUT} ::= \langle b \rangle \\
L_{\text{-,M,D,PM}}: & P, Q, R ::= \ldots & \mathit{IN} ::= (T) & \mathit{OUT} ::= \langle b \rangle \\
L_{\text{-,M,C,NO}}: & P, Q, R ::= \ldots & \mathit{IN} ::= a(x) & \mathit{OUT} ::= \overline{a}\langle b \rangle \\
L_{\text{-,M,C,PM}}: & P, Q, R ::= \ldots & \mathit{IN} ::= a(T) & \mathit{OUT} ::= \overline{a}\langle b \rangle \\
L_{\text{-,P,D,NO}}: & P, Q, R ::= \ldots & \mathit{IN} ::= (\widetilde{x}) & \mathit{OUT} ::= \langle \widetilde{b} \rangle \\
L_{\text{-,P,D,PM}}: & P, Q, R ::= \ldots & \mathit{IN} ::= (\widetilde{T}) & \mathit{OUT} ::= \langle \widetilde{b} \rangle \\
L_{\text{-,P,C,NO}}: & P, Q, R ::= \ldots & \mathit{IN} ::= a(\widetilde{x}) & \mathit{OUT} ::= \overline{a}\langle \widetilde{b} \rangle \\
L_{\text{-,P,C,PM}}: & P, Q, R ::= \ldots & \mathit{IN} ::= a(\widetilde{T}) & \mathit{OUT} ::= \overline{a}\langle \widetilde{b} \rangle
\end{array}$$

where

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

Here and in what follows, $\widetilde{\ \cdot\ }$ denotes a (possibly empty) sequence of elements of kind $\cdot$; whenever useful, we shall write a tuple $\widetilde{\ \cdot\ }$ as the sequence of its elements separated by a comma, or consider it just as a set. Templates of kind x are called *formal* and can be instantiated by every name in a communication; templates of kind $\ulcorner n \urcorner$ are called *actual* and impose that the datum received 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, $\mathrm{FN}(P)$ and $\mathrm{BN}(P)$, and of alpha-conversion, $=_\alpha$, are assumed. We let $\mathrm{N}(P)$ denote $\mathrm{FN}(P) \cup \mathrm{BN}(P)$.

Notice that $L_{\mathrm{A,M,C,NO}}$, $L_{\mathrm{A,P,C,NO}}$, $L_{\mathrm{S,M,C,NO}}$, $L_{\mathrm{S,P,C,NO}}$ exploit the communication paradigm of the monadic/polyadic asynchronous/synchronous π-calculus [5, 26, 29, 30]; $L_{\mathrm{A,P,D,PM}}$ relies on the communication paradigm adopted in LINDA [22]; $L_{\mathrm{A,M,D,NO}}$ and $L_{\mathrm{A,P,D,NO}}$ rely on the communication paradigm adopted in (monadic/polyadic) Mobile Ambients [14]; finally, $L_{\mathrm{A,P,C,PM}}$ relies on the communication paradigm adopted, e.g., in μKLAIM [17] or in semantic-π [15].

Remark 2.1 $\Lambda_{s,a,m,p}$ can be easily ordered; in particular, $L_{\beta_1\beta_2\beta_3\beta_4}$ can be encoded in $L_{\beta'_1\beta'_2\beta'_3\beta'_4}$ if and only if, for every $i \in \{1,2,3,4\}$, it holds that $\beta_i \leq \beta'_i$, where '$\leq$' is the least reflexive relation satisfying the following axioms:

$$\mathrm{A} \leq \mathrm{S} \qquad \mathrm{M} \leq \mathrm{P} \qquad \mathrm{D} \leq \mathrm{C} \qquad \mathrm{NO} \leq \mathrm{PM}$$

As an extremal example, consider $L_{\mathrm{A,M,D,NO}}$ and $L_{\mathrm{S,P,C,PM}}$: asynchrony is a particular case of synchrony (all output prefixes are followed by $\mathbf{0}$); monadic data are a particular case of polyadic data (all of length one); a shared dataspace can be modeled by letting all k-ary communications happen on the same fixed channel (e.g., mnemonically named $\mathtt{k}$); finally, absence of pattern-matching can be obtained by only considering formal templates.

Remark 2.2 The polyadic versions of channel-based languages are usually typed to ensure that every channel always carries data of the same length [29, 40, 42, 45, 47]. This choice is justified by the fact that channel-based interaction is similar to port-based access to services and, usually, ports are accessed by respecting some predefined message format; accessing a port by using a wrong message format is clearly a programming error and should be avoided. Unless stated otherwise, from now on we shall only consider well-typed $L_{\cdot,\mathrm{P,C},\cdot}$ processes (under any type system that ensures arity matching); thus, $L_{\cdot,\mathrm{P,C},\cdot}$ will denote the set of all (and only) such processes.

2.2 Operational semantics

We shall give the operational semantics of the languages by means of a *labeled transition system* (LTS) describing the actions a process can perform to evolve. This is just one of the possibilities developed in the last decades to describe the operational semantics of a concurrent language; another successful style is via reductions [27, 29]. Moreover, there are several possible formulations of a LTS (early vs late, including structural equivalence or not, ...) [36]. Here, the particular LTS we are going to develop is the one that will allow us to carry out our proofs in the simplest possible way. All the results we are going to present do not depend on this choice and can be rephrased under any 'compatible' operational semantics.

Judgments in the operational semantics take the form $P \xrightarrow{\alpha} P'$, meaning that P can become P' upon exhibition of label α. *Labels* take the form

$$\alpha \ ::= \ \tau \ \mid \ a?\widetilde{b} \ \mid \ (\nu\widetilde{c})a!\widetilde{b} \ \mid \ ?\widetilde{b} \ \mid \ (\nu\widetilde{c})!\widetilde{b}$$

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; similarly, in dataspace-based languages $?\widetilde{b}$ 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, $(\nu\widetilde{c})^{-}!\widetilde{b}$ stands for either $(\nu\widetilde{c})a!\widetilde{b}$ or $(\nu\widetilde{c})!\widetilde{b}$; similarly, $^{-}?\widetilde{b}$ stands for either $a?\widetilde{b}$ or $?\widetilde{b}$. As usual, $\textsc{Bn}((\nu\widetilde{c})^{-}!\widetilde{b}) \triangleq \widetilde{c}$; $\textsc{Fn}(\alpha)$ 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 common rules, reported below, are an easy adaptation of an early-style LTS for the π-calculus; thus, we do not comment on them and refer the interested reader to [30, 36, 45].

$$\frac{P \xrightarrow{?\widetilde{b}} P' \qquad Q \xrightarrow{!\widetilde{b}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{a?\widetilde{b}} P' \qquad Q \xrightarrow{a!\widetilde{b}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{(\nu\widetilde{c})^{-}!\widetilde{b}} P' \qquad n \in \widetilde{b} \setminus \{^{-}, \widetilde{c}\}}{(\nu n)P \xrightarrow{(\nu n,\widetilde{c})^{-}!\widetilde{b}} P'}$$

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

Structural equivalence, $\equiv$, rearranges a process to let it evolve according to the rules of the LTS. Its defining axioms are the standard π-calculus ones:

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

$$\textbf{if } n = n \textbf{ then } P \textbf{ else } Q \equiv P \qquad \textbf{if } n = m \textbf{ then } P \textbf{ else } Q \equiv Q \ \text{ if } n \neq m$$

$$P \equiv P' \ \text{ if } P =_{\alpha} P' \qquad (\nu n)\mathbf{0} \equiv \mathbf{0} \qquad (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$$

$$P \mid (\nu n)Q \equiv (\nu n)(P \mid Q) \ \text{ if } n \notin \textsc{Fn}(P) \qquad *P \equiv P \mid *P$$

We are left with the key rules of every language, i.e. those for process actions. The rules for output actions in languages $L_{\mathrm{A},^{-},\mathrm{D},^{-}}$, $L_{\mathrm{A},^{-},\mathrm{C},^{-}}$, $L_{\mathrm{S},^{-},\mathrm{D},^{-}}$ and $L_{\mathrm{S},^{-},\mathrm{C},^{-}}$ are, respectively

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

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: $\ulcorner n \urcorner$ matches n and x matches every name. This can be formalized via a partial function, called *pattern-matching* and written 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 formalized by the following rules:

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

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

where 'ϵ' denotes the empty substitution and '$\uplus$' denotes union of partial functions with disjoint domains. Now, the operational rules for input actions in languages $L_{\text{-,-,D,-}}$ and $L_{\text{-,-,C,-}}$ are

$$(\widetilde{T}).P \xrightarrow{?\widetilde{b}} P\sigma \qquad \text{if } \textsc{Match}(\widetilde{T}\,;\,\widetilde{b}) = \sigma$$
$$a(\widetilde{T}).P \xrightarrow{a?\widetilde{b}} P\sigma$$

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 $\xrightarrow{\tau}$, $\stackrel{\alpha}{\Longrightarrow}$ 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 $P \stackrel{\alpha}{\Longrightarrow}$. Moreover, we let ϕ range over visible actions (i.e. labels different from τ) and ρ 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, $N \stackrel{\varepsilon}{\Longrightarrow}$ is defined as $N \Rightarrow$ and $N \stackrel{\phi\cdot\rho}{\Longrightarrow}$ is defined as $N \stackrel{\phi}{\Longrightarrow}\stackrel{\rho}{\Longrightarrow}$.

We conclude this part with a proposition that collects together some properties of the LTSs we have just defined; the proof of these results easily follows from the definition of the LTSs.

Proposition 2.1 *The following facts hold:*

1. *if $P \in L_{\text{-,-,-,NO}}$ and $P \xrightarrow{\text{-}?\widetilde{b}}$, then $P \xrightarrow{\text{-}?\widetilde{c}}$ for every $\widetilde{c}$ of the same length as $\widetilde{b}$;*
2. *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 $P_2 \xrightarrow{a!\widetilde{b}} P'_2$;*
3. *if $P \in L_{\text{A,-,-,-}}$ and $P \xrightarrow{(\nu\widetilde{c})\text{-}!\widetilde{b}} \xrightarrow{\alpha} P'$, for $\widetilde{c} \cap \text{N}(\alpha) = \emptyset$, then $P \xrightarrow{\alpha} \xrightarrow{(\nu\widetilde{c})\text{-}!\widetilde{b}} P'$; moreover, if $\alpha = \text{-}?\widetilde{b}$, then $P \xrightarrow{\tau} (\nu\widetilde{c})P'$.*

2.3 Behavioural semantics

To conclude the presentation of the languages, we now define a very natural notion of equivalence that equates terms that behave in the same way. There are several possible notions of behaviour and, correspondingly, several possible equivalences. Here, we present the most basic one, namely *(strong) barbed congruence* [31]; such an equivalence provides the minimum abstraction level from the operational semantics of processes.

Intuitively, barbed congruence requires that, in any execution context, two equivalent processes offer the same observable behaviour along every possible computation. To formally define this requirement, we need to fix two notions: what is a context and what is observable in a process.

Definition 2.1 (Context) *A* context $C[\_]$ *is a process with one occurrence of* $\mathbf{0}$ *replaced with the hole* $\_$*; the hole can be filled with any process P and the resulting process is denoted as $C[P]$.*

Definition 2.2 (Barbs) *P offers the* barb $(\nu\widetilde{c})\text{-}!\widetilde{b}$ *iff* $P \xrightarrow{(\nu\widetilde{c})\text{-}!\widetilde{b}}$.

Usually, in a π-calculus-based framework, observables (usually called *barbs*) are visible actions [31]; however, as argued in [2, 7], in an asynchronous setting only output actions are observable. Moreover, in π-calculus only the channel where the output happens is relevant, since the argument can be checked by the execution context; however, this feature is not straightforwardly adaptable to dataspace-based languages. So, to uniformly define equivalences in $\Lambda_{s,a,m,p}$, we consider as a barb any (full-fledged) output action; it is easy to prove that, in frameworks like the π-calculus (where weaker forms of barb are usually assumed), the congruences resulting from these barbs and from more traditional barbs do coincide.

Definition 2.3 (Barbed bisimulation and congruence) *A symmetric relation $\mathfrak{R}$ between processes is a* barbed bisimulation *if, whenever $(P, Q) \in \mathfrak{R}$, it holds that*

- *every barb in P is also a barb in Q; and*
- *if $P \xrightarrow{\tau} P'$ then there exists a Q' such that $Q \xrightarrow{\tau} Q'$ and $(P', Q') \in \mathfrak{R}$.*

Barbed bisimilarity*, written $\dot{\simeq}$, is the largest barbed bisimulation.*

Two processes P and Q are barbed congruent*, written $P \simeq Q$, whenever $C[P] \dot{\simeq} C[Q]$, for every context $C[\_]$.*

Proposition 2.2 *Barbed congruence is an equivalence relation.*

3 Quality of an Encoding and Overview of our Results

3.1 Reasonable Encodings

We now study the relative expressive power of the languages in $\Lambda_{s,a,m,p}$ by trying to encode one in another. Formally, an *encoding* $[\![\cdot]\!]$ is a function mapping terms of the source language into terms of the target language. Associated to every encoding, there is a *renaming policy* that establishes how names are translated. For example, it is possible that an encoding fixes some names to play a precise role (see, e.g., the simple encoding of dataspaces via channels described in Remark 2.1) or it can translate a single name into a tuple of names (a sample of this kind of encoding will be given at the end of Section 5.1). This fact can be obtained either by assuming that the target language has more names than the source one, or by relying on renaming policies, that we now formally define.

Definition 3.1 (Renaming policy) *Given an encoding $[\![\cdot]\!]$, its underlying renaming policy is a function $\varphi_{[\![\,]\!]} : \mathcal{N} \mapsto \mathcal{N}^k$, for some constant $k > 0$, such that $\forall a, b \in \mathcal{N}$ with $a \neq b$, it holds that $\varphi_{[\![\,]\!]}(a) \cap \varphi_{[\![\,]\!]}(b) = \emptyset$ (where $\varphi_{[\![\,]\!]}(\cdot)$ is simply considered a set here).*

The disjointness requirement we put on $\varphi_{[\![\,]\!]}$ states that the renaming policy is, in some sense, 'minimal'. Indeed, if two different names are associated to non-disjoint tuples, then any pair of names should satisfy this property (names are all at the same level); but then, the names present in every tuple can be considered 'reserved' and every name could be mapped to a shorter tuple.

We now define *reasonable* encodings.

Definition 3.2 (Reasonable Encoding) *An encoding $[\![\cdot]\!]$ is* reasonable *if it enjoys the following properties:*

1. (compositionality): *for every unary operator* op *there exists a context* $C_{\mathsf{op}}[\_]$ *such that, for every* P*, it holds that* $[\![\, \mathsf{op}(P) \,]\!] \triangleq C_{\mathsf{op}}[[\![P]\!]]$*; for every binary operator* op *there exists a two-holes context* $C_{\mathsf{op}}[\__1; \__2]$ *such that, for every* P_1 *and* P_2*, it holds that* $[\![\, \mathsf{op}(P_1, P_2) \,]\!] \triangleq C_{\mathsf{op}}[[\![P_1]\!]; [\![P_2]\!]]$.
2. (name invariance): *for every name substitution* σ*, it holds that* $[\![P\sigma]\!] = [\![P]\!]\sigma'$*, where* σ' *ordinately replaces* $\varphi_{[\![\ ]\!]}(a)$ *with* $\varphi_{[\![\ ]\!]}(b)$*, for every* a *replaced with* b *in* σ.
3. (faithfulness): $P \Downarrow$ *iff* $[\![P]\!] \Downarrow$*, where* $P \Downarrow$ *means that* $P \stackrel{\alpha}{\Longrightarrow}$*, for any* $\alpha \neq \tau$*;* $P \Uparrow$ *iff* $[\![P]\!] \Uparrow$*, where* $P \Uparrow$ *means that* $P \stackrel{\tau}{\longrightarrow}^{\omega}$.
4. (operational correspondence):
   (a) *if* $P \Rightarrow P'$ *then* $[\![P]\!] \Rightarrow \simeq [\![P']\!]$*;*
   (b) *if* $[\![P]\!] \Rightarrow Q$ *then there exists a* P' *such that* $P \Rightarrow P'$ *and* $Q \Rightarrow \simeq [\![P']\!]$.

Let us now briefly discuss the properties just defined. The encoding should be compositional, i.e. the encoding of a compound process must be defined by plugging the encoding of its components in a context that only depend on the operator under translation; of course, we must generalize in the expected way the notion of context to deal with binary operators of $\Lambda_{s,a,m,p}$. Notice that some form compositionality has been assumed for specific operators in [12, 13, 16, 35, 39]: mainly, it is required that the parallel composition must be mapped homomorphically. By giving minimality up, here we assume that every operator must be translated compositionally, since every concrete encoding satisfies this property. Moreover, we do not assume any form of homomorphism, to strengthen our impossibility results; notice that this does not undermine our encodability results, since we usually map language operators different from input and output prefixes homomorphically.

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; for this reason, we required name invariance, that is related to a similar property in [13, 35, 39]. Their formulation could be considered more liberal (because no constraint is posed on σ'), but in practice our formulation is just more detailed, since it fully describes the way in which σ' must be chosen.

The idea behind faithfulness is that 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. There are several ways to formalize this idea. We decided to be quite liberal and consider only the possibility of interacting with an external observer and of having non-terminating computations; similar notions can also be found, e.g., in [13, 35, 39, 49]. Interaction is one of the key aspects in concurrency theory; thus, mapping a process able (resp., unable) to interact with an observer into a process unable (resp., able) to do the same is clearly a radical change in the semantics of the process translated. Notice that our formulation of this property is really minimal: it only tests the possibility of performing any visible action. This fact strengthens our impossibility results, but our encodability results are not undermined by this choice: they would also enjoy properties expressed in terms of more significant observables, such as those in [2, 7, 31] (see [24]). Concerning divergence, one may argue that it does not matter if it arises with negligible probability or in unfair computations. However, suppose that every encoding of $\mathcal{L}_2$ in $\mathcal{L}_1$ introduces some kind of divergence; this can be used as an evidence of the fact that the constructs of $\mathcal{L}_1$ are not powerful enough to mimic the constructs of $\mathcal{L}_2$: to preserve all the functionalities of a terminating source term, every encoding has to add further behaviours to the encoded term. Thus, $\mathcal{L}_1$ cannot be as expressive as $\mathcal{L}_2$. A similar property is crucial in [25] to prove that the `test_and_set` primitive is strictly more expressive than `read` and `write`.

Finally, operational correspondence is traditionally not included among the criteria used to prove impossibility results, whereas it is (almost) always present to prove soundness of encodings (see, e.g., [41, 42, 43, 48]) or of implementations (see, e.g., [21, 38, 44]). We decided to include it as a reasonableness criterion for two reasons: first, we want to use reasonableness also for encodability results; second, if (almost) every known encoding is designed to enjoy it, we pragmatically argue that it is one of the properties that every encoding should satisfy to be acceptable.

On the Formulation of Operational Correspondence. In Definition 3.2(4), we have adopted (a slight generalization of) the most general formulation of operational correspondence put forward in [34]. One of the main advantages of defining the latter property up to *strong* barbed congruence is that such an equivalence allows us to get rid of dead code (possibly arising from the encoding) by also keeping divergence into account. As we have argued, divergence is a key aspect when dealing with expressiveness issues; thus, any equivalence (like *weak* barbed congruence) that equates a divergent and a non-divergent process would not be appropriate in this setting.

Of course, our impossibility results would be stronger if operational correspondence were formulated up to a coarser relation. However, all our impossibility results are proved by relying on the fact that operational correspondence is formulated up to a τ-sensitive relation (where a relation $\mathfrak{R}$ is *τ-sensitive* whenever $P \mathfrak{R} Q$ and $Q \stackrel{\tau}{\longrightarrow}$ imply that $P \stackrel{\tau}{\longrightarrow}$); indeed, we could replace strong barbed congruence with any other such a relation (e.g., the expansion preorder [4]) without breaking our proofs. On the contrary, we do not know how to prove such results without this assumption, though we strongly conjecture that they all hold under any 'meaningful' behavioural relation.

It has to be said that all encodability results we are aware of (e.g., [21, 38, 41, 42, 43, 44, 48]) enjoy operational correspondence up to a τ-sensitive relation; the only notable exceptions are the encodings of separate and of input-guarded choice π-calculus into the asynchronous π-calculus [32, 34]. We now sketch and discuss a different formulation of operational correspondence that covers all the encodings we know. Since the need for '$\simeq$' in Definition 3.2(4) is usually to get rid of dead processes left by the encoding, we could define such a property as

(a) if $P \Rightarrow P'$ then $[\![P]\!] \Rightarrow\!\propto [\![P']\!]$;

(b) if $[\![P]\!] \Rightarrow Q$ then there exists a P' such that $P \Rightarrow P'$ and $Q \Rightarrow\!\propto [\![P']\!]$

where

$$K' \propto K \;\text{ whenever }\; K' \equiv (\nu\widetilde{n})(K \mid H), \text{ for some } H \text{ and } \widetilde{n} \text{ such that } (\nu\widetilde{n})H \;\mathfrak{R}\; \mathbf{0}$$

and $\mathfrak{R}$ is an *arbitrary* behavioural relation (in particular, notice that we do *not* need $\mathfrak{R}$ be τ-sensitive). Under this formulation of operational correspondence, we would have that all the best known encodings appearing in the literature (including those in [32, 34]) are deemed reasonable; moreover, all the impossibility proofs we are going to develop still hold, since '$\propto$' is τ-sensitive. However, the definition of '$\propto$' is ad hoc and so more debatable, even if it exactly captures the intuition that we want to express via operational correspondence; for this reason, we prefer to work with the more standard formulation of operational correspondence presented in Definition 3.2(4).

3.2 Technical Preliminaries

One of the most critical things to prove in our encodability results will be Definition 3.2(4b) and that the encoding does not introduce divergence. In several cases, we shall prove the following property

that, as we now show, implies both Definition 3.2(4b) and divergence reflection:

$$\text{If } [\![P]\!] \xrightarrow{\tau} Q \text{ then there exists a } P' \text{ such that } P \xrightarrow{\tau} P' \text{ and } Q \gtrsim [\![P']\!] \tag{1}$$

Intuitively, $Q \gtrsim [\![P']\!]$ means that Q can only reduce to a process barbed congruent to $[\![P']\!]$. Relation '$\gtrsim$', that we call *confluence*, resembles the expansion preorder [4] and is formally defined as follows:

Definition 3.3 (Confluence) *We write $P \gtrsim Q$ whenever there exist $P_0, P_1, \ldots, P_k$ such that*

- $P \triangleq P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_k \simeq Q$*; and*
- *for every $i = 0, \ldots, k-1$ it holds that $P_i \xrightarrow{\tau} P'$ implies $P' \simeq P_{i+1}$.*

Lemma 3.1 *If $P \simeq Q$ and $Q \gtrsim R$, then $P \gtrsim R$.*

Proof: Let $Q \triangleq Q_0 \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_k \simeq R$. Since $P \simeq Q$, we can find $P_0, \ldots, P_k$ such that $P \triangleq P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_k$ and $P_i \simeq Q_i$, for every $i = 0, \ldots, k$. The thesis follows by transitivity of $\simeq$. ∎

Lemma 3.2 *Let $P \gtrsim Q$; then, $P \Uparrow$ if and only if $Q \Uparrow$.*

Proof: Let $P \triangleq P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_k \simeq Q$. If $Q \Uparrow$, then also P_k diverges (since $\simeq$ is sensitive to divergence) and, trivially, $P \Uparrow$. Vice versa, let $P \Uparrow$, i.e. $P \xrightarrow{\tau} P'_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P'_k \xrightarrow{\tau} \ldots$. By definition, $P'_i \simeq P_i$, for every $i = 1, \ldots, k$; this implies that P_k and Q diverge. ∎

Proposition 3.3 *If $[\![\cdot]\!]$ satisfies* (1)*, then it does not introduce divergence and it satisfies Definition 3.2(4b).*

Proof: Assume that $[\![P]\!] \Uparrow$, i.e. $[\![P]\!] \xrightarrow{\tau} Q \xrightarrow{\tau}{}^{\omega}$; by (1), there exists a $P \xrightarrow{\tau} P'$ such that $Q \gtrsim [\![P']\!]$. By Lemma 3.2, $[\![P']\!] \Uparrow$, i.e. $[\![P']\!] \xrightarrow{\tau} Q' \xrightarrow{\tau}{}^{\omega}$; again by (1), there exists a $P' \xrightarrow{\tau} P''$ such that $Q' \gtrsim [\![P'']\!]$. By iterating this reasoning, we can build up a divergent computation from P; thus, $[\![\cdot]\!]$ does not introduce divergence.

We now prove that $[\![\cdot]\!]$ satisfies the following property:

$$\text{If } [\![P]\!] \xrightarrow{\tau}{}^{n} Q, \text{ for } n \geq 1, \text{ then } P \xrightarrow{\tau}{}^{+} P' \text{ for some } P' \text{ such that } Q \gtrsim [\![P']\!] \tag{2}$$

Then, (2) and Definition 3.3 imply Definition 3.2(4b). The proof of (2) is by induction on n. The base case is (1). For the inductive case, let $[\![P]\!] \xrightarrow{\tau}{}^{n} Q' \xrightarrow{\tau} Q$; the inductive hypothesis states that $P \xrightarrow{\tau}{}^{+} P'$ and $Q' \gtrsim [\![P']\!]$, i.e. $Q' \xrightarrow{\tau}{}^{k} \simeq [\![P']\!]$. If $k > 0$, then Q is barbed congruent to a process $Q'' \gtrsim [\![P']\!]$; by Lemma 3.1, $Q \gtrsim [\![P']\!]$. If $k = 0$, then $[\![P']\!] \xrightarrow{\tau} Q''$ such that $Q'' \simeq Q$. By (1), there exists a P'' such that $P' \xrightarrow{\tau} P''$ and $Q'' \gtrsim [\![P'']\!]$. By Lemma 3.1, $Q \gtrsim [\![P'']\!]$, for $P \xrightarrow{\tau}{}^{+} P''$; this suffices to conclude. ∎

To prove our impossibility results, we shall exploit the following property of any reasonable encoding.

Proposition 3.4 *Let P be a process such that $P \xrightarrow{\tau}\!\!\!\!\!/\;$ but $[\![P]\!] \xrightarrow{\tau}$; then, $[\![\cdot]\!]$ is not reasonable.*

Proof: By contradiction. Since $[\![\cdot]\!]$ is reasonable, it is operationally corresponding. Then, $[\![P]\!] \stackrel{\tau}{\longrightarrow} Q$ implies that $P \Rightarrow P'$, for some P' such that $Q \Rightarrow \simeq [\![P']\!]$. But the only P' such that $P \Rightarrow P'$ is P itself; thus, $[\![P]\!] \stackrel{\tau}{\longrightarrow}^{+} \simeq [\![P]\!]$ and this implies that $[\![P]\!]$ diverges. ∎

The previous proposition shows that our notion of reasonableness implies *promptness* of our encodings, as defined in [34]. This is due to the fact that we have formulated operational correspondence up to a τ-sensitive relation, viz. strong barbed congruence. Indeed, if '$\simeq$' were not τ-sensitive, then $[\![P]\!] \stackrel{\tau}{\longrightarrow}^{+} \simeq [\![P]\!]$ would not imply that $[\![P]\!]$ diverges. Nevertheless, given any non-prompt but confluent encoding (i.e., where the preliminary 'administrative' τ-actions form a confluent reduction, as defined in Definition 3.3), we can easily define a corresponding prompt encoding. Thus, if we prove the impossibility for a prompt encoding of $\mathcal{L}_1$ into $\mathcal{L}_2$, we can also conclude the impossibility for a non-prompt but confluent encoding. On the contrary, our proofs do not formally state anything about existence/non-existence of encodings that are both non-prompt and non-confluent. We strongly conjecture that all the impossibility results we are going to prove hold also for such encodings, but we have still not been able to prove them.

A simple corollary of the previous result that we shall extensively exploit in our proofs is the following proposition, that regulates the possible evolutions of the encoding of a compound parallel process.

Proposition 3.5 *Let $[\![\cdot]\!]$ be reasonable, $P_1 \stackrel{\tau}{\not\longrightarrow}$ and $P_2 \stackrel{\tau}{\not\longrightarrow}$; then $[\![P_1 \mid P_2]\!] \stackrel{\tau}{\longrightarrow}$ is possible only if $[\![P_1]\!]$ and $[\![P_2]\!]$ communicate.*

Proof: By compositionality, $[\![P_1 \mid P_2]\!] = C_{|}[[\![P_1]\!]; [\![P_2]\!]]$; then, $[\![P_1 \mid P_2]\!] \stackrel{\tau}{\longrightarrow}$ can be generated only in four ways:

1. either by $[\![P_i]\!]$, for $i \in \{1, 2\}$;
2. or by $C_{|}[\__1; \__2]$;
3. or by a communication between $C_{|}[\__1; \__2]$ and $[\![P_i]\!]$, for $i \in \{1, 2\}$;
4. or by a communication between $[\![P_1]\!]$ and $[\![P_2]\!]$.

We now prove that only the last possibility does not lead to a contradiction; this suffices to conclude. The first possibility is directly ruled out by Proposition 3.4; the second possibility is ruled out by the fact that otherwise $[\![\mathbf{0} \mid \mathbf{0}]\!]$ would reduce (and, again by Proposition 3.4, $[\![\cdot]\!]$ would not be reasonable); similarly, the third possibility is ruled out by that fact that otherwise either $[\![P_1 \mid \mathbf{0}]\!]$ or $[\![\mathbf{0} \mid P_2]\!]$ would reduce, according to whether $i = 1$ or $i = 2$. ∎

3.3 Overview of the Results and Structure of our Proofs

The results of our paper are summarized in Figure 1. There, we write $\mathcal{L}_1 \leftrightarrow \mathcal{L}_2$ whenever $\mathcal{L}_1$ can be reasonably encoded in $\mathcal{L}_2$ and vice versa. On the contrary, we write $\mathcal{L}_2 \rightarrow \mathcal{L}_1$ whenever $\mathcal{L}_2$ can be reasonably encoded in $\mathcal{L}_1$ but not vice versa. We shall say that $\mathcal{L}_1$ and $\mathcal{L}_2$ *have the same expressive power* if $\mathcal{L}_1 \leftrightarrow \cdots \leftrightarrow \mathcal{L}_2$; similarly, we shall say that $\mathcal{L}_1$ is *(strictly) more expressive than* $\mathcal{L}_2$ if $\mathcal{L}_2 \geq \cdots \geq \rightarrow \geq \cdots \geq \mathcal{L}_1$, for $\geq \in \{\leftrightarrow, \rightarrow\}$. We shall say that $\mathcal{L}_1$ and $\mathcal{L}_2$ *are incomparable* if neither $\mathcal{L}_1$ and $\mathcal{L}_2$ have the same expressive power, nor one is more expressive than the other. Finally, the dashed arrow placed between $L_{\text{S,M,D,PM}}$ and $L_{\text{S,M,C,NO}}$ denotes existence of an 'almost' reasonable encoding of the former in the latter; indeed, the encoding we are going to present does not satisfy

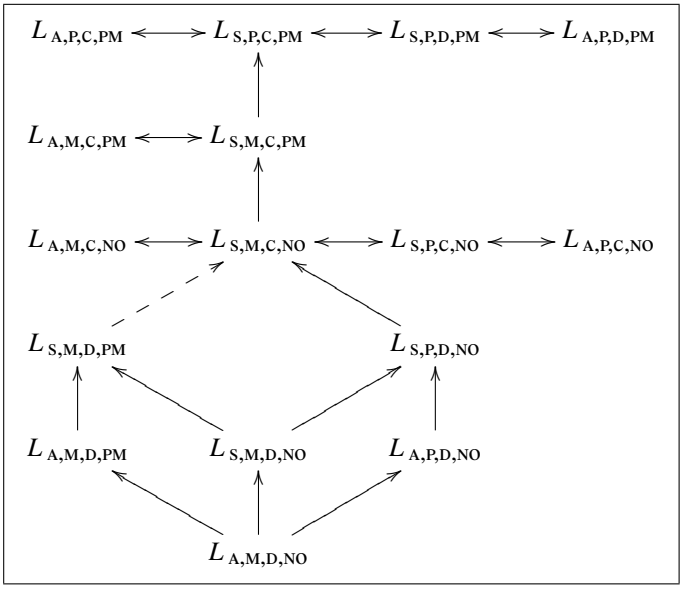

Figure 1: Overview of the Results

operational correspondence as formulated in Definition 3.2, but a slightly weaker form, akin to the one satisfied by the encodings in [32, 34]. We have not been able to define a reasonable encoding (as defined in Definition 3.2) nor to prove the impossibility of such a result.

For impossibility results, we shall work by contradiction and prove that existence of a reasonable encoding $[\![\cdot]\!]$ leads to contradict some reasonableness property, usually Proposition 3.4 or divergence reflection. In particular, we shall find a non-evolving (or terminating) source process whose encoding turns out to be evolving (or divergent). This way of working is somehow similar to [12, 16, 49] but different from [13, 35, 39], where non-encodability is proved as a corollary of the fact that the source language can solve a problem that the target cannot solve.

For encodability results, we shall recall Remark 2.1 whenever the encoding is trivial. Otherwise, we shall present an encoding by only describing the translation of the key operators, usually input and output prefixes; the remaining operators will be translated homomorphically (this trivially satisfies Definition 3.2(1)). Then, we are going to explicitly prove only some of the reasonableness conditions, usually that the encoding does not introduce divergence and Definition 3.2(4b). Definition 3.2(2) and the first part of Definition 3.2(3) hold by construction of the encoding. Definition 3.2(4a) can be routinely proved by a double induction: the first one is over the number of τ-steps in the $\Rightarrow$ of the premise; the second one is used to prove the claim

$$\text{if } P \xrightarrow{\tau} P' \text{ then } [\![P]\!] \stackrel{\tau}{\Longrightarrow} \simeq [\![P']\!] \tag{3}$$

and it is carried out over the shortest inference for $\xrightarrow{\tau}$. Finally, preservation of divergence is a trivial consequence of (3).

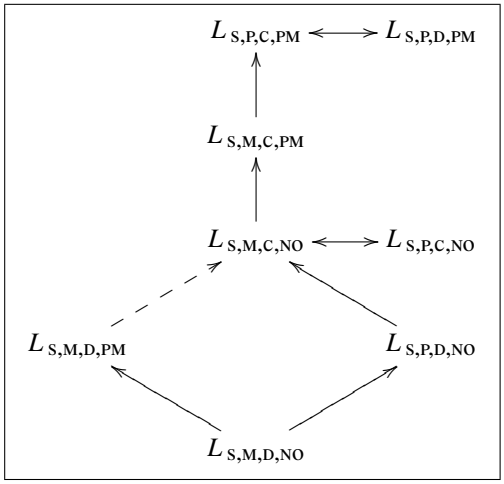

Figure 2: The Hierarchy of Synchronous Communication Primitives

4 On the Relative Expressive Power of Synchronous Communication Primitives

In this section, for the sake of presentation, we first restrict our attention to *synchronous* communication primitives and rigorously prove the relationships depicted in Figure 2.

$L_{\text{S,P,C,PM}}$ and $L_{\text{S,P,D,PM}}$ have the same expressive power

To prove this claim, it suffices to prove that $L_{\text{S,P,C,PM}}$ can be reasonably encoded in $L_{\text{S,P,D,PM}}$, since the latter can be encoded in the former (see Remark 2.1). The only feature of $L_{\text{S,P,C,PM}}$ not present in $L_{\text{S,P,D,PM}}$ is the possibility of specifying the name of a channel where the exchange happens. However, thanks to polyadicity and pattern-matching, this feature can be very easily encoded in $L_{\text{S,P,D,PM}}$: it suffices to impose that the first name of every datum represents the name of the channel where the interaction is scheduled and that every input argument starts with the corresponding actual template. This discipline is rendered by the following encoding:

$$[\![\, \overline{a}\langle\widetilde{b}\rangle.P \,]\!] \triangleq \langle a, \widetilde{b}\rangle.[\![\, P \,]\!] \qquad\qquad [\![\, a(\widetilde{T}).P \,]\!] \triangleq (\ulcorner a \urcorner, \widetilde{T}).[\![\, P \,]\!]$$

It is interesting to notice that this discipline is assumed in the original presentation of LINDA [22].

Proposition 4.1 *There exists a reasonable encoding of $L_{\text{S,P,C,PM}}$ into $L_{\text{S,P,D,PM}}$.*

Proof: For the encoding we have presented, Definition 3.2(4b) can be proved in the following formulation (that strengthens (1)): if $[\![\, P \,]\!] \xrightarrow{\tau} Q$, then $Q = [\![\, P' \,]\!]$ for some P' such that $P \xrightarrow{\tau} P'$. This result[1] is proved by an easy induction over the shortest inference for $\xrightarrow{\tau}$ and it entails that the encoding cannot introduce divergence. The remaining reasonableness requirements are routinely proved. ∎

[1] Such a formulation of Definition 3.2(4b) would also enable us to prove full abstraction with respect to strong barbed congruence.

$L_{\text{S,P,C,PM}}$ is more expressive than $L_{\text{S,M,C,PM}}$

To prove this claim, it suffices to prove that there exists no reasonable encoding of $L_{\text{S,P,C,PM}}$ in $L_{\text{S,M,C,PM}}$, since the latter is a sub-language of the former.

Theorem 4.2 *There exists no reasonable encoding of $L_{\text{S,P,C,PM}}$ in $L_{\text{S,M,C,PM}}$.*

Proof: Assume that $[\![\cdot]\!]$ is reasonable and consider the process $a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{a}\langle b, c\rangle$, for a, b and c pairwise distinct; such a process evolves into $\mathbf{0}$. By operational correspondence and faithfulness, $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{a}\langle b, c\rangle]\!] \stackrel{\tau}{\Longrightarrow} \simeq [\![\mathbf{0}]\!]$. By Proposition 3.5, $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner)]\!]$ and $[\![\overline{a}\langle b, c\rangle]\!]$ must communicate; thus, we have that $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner)]\!] \xrightarrow{n?m}$ and $[\![\overline{a}\langle b, c\rangle]\!] \xrightarrow{(\nu \widetilde{m})n!m}$ (or vice versa – and this case is handled similarly), for some n, $\widetilde{m}$ and m.

If the input of $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner)]\!]$ has been generated by relying on a formal template, then $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner)]\!] \xrightarrow{n?l}$, for every l. Hence, $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{a}\langle c, b\rangle]\!] \xrightarrow{\tau}$, if $n \notin \varphi_{[\![\,]\!]}(b) \cup \varphi_{[\![\,]\!]}(c)$; else, $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{b}\langle a, c\rangle]\!] \xrightarrow{\tau}$, if $n \in \varphi_{[\![\,]\!]}(c)$, and $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{c}\langle b, a\rangle]\!] \xrightarrow{\tau}$, otherwise. So, assume that the input of $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner)]\!]$ relies on an actual template (thus, $\widetilde{m}$ must be $\emptyset$); we then consider the following possibilities for n and m:

1. $\{n, m\} \cap \varphi_{[\![\,]\!]}(c) = \emptyset$: let $d \neq c$ and $\varphi_{[\![\,]\!]}(d) \cap \{n, m\} = \emptyset$. Then, consider the permutation that swaps c and d; thus, $[\![\overline{a}\langle b, d\rangle]\!] \xrightarrow{n!m}$ and $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{a}\langle b, d\rangle]\!] \xrightarrow{\tau}$.
2. $n \in \varphi_{[\![\,]\!]}(c)$, $m \notin \varphi_{[\![\,]\!]}(b)$: let $d \neq b$ and $\varphi_{[\![\,]\!]}(d) \cap \{n, m\} = \emptyset$. Now, consider the permutation that swaps b and d; like before, $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{a}\langle d, c\rangle]\!] \xrightarrow{\tau}$.
3. $n \in \varphi_{[\![\,]\!]}(c)$, $m \in \varphi_{[\![\,]\!]}(b)$: let $d \neq a$ and $\varphi_{[\![\,]\!]}(d) \cap \{n, m\} = \emptyset$. Consider the permutation that swaps a and d, and conclude that $[\![a(\ulcorner b \urcorner, \ulcorner c \urcorner) \mid \overline{d}\langle b, c\rangle]\!] \xrightarrow{\tau}$.
4. $m \in \varphi_{[\![\,]\!]}(c)$, $n \notin \varphi_{[\![\,]\!]}(b)$: like case 2.
5. $m \in \varphi_{[\![\,]\!]}(c)$, $n \in \varphi_{[\![\,]\!]}(b)$: like case 3.

In all these cases, we have that $[\![\cdot]\!]$ is not reasonable because of Proposition 3.4. ∎

$L_{\text{S,M,C,PM}}$ is more expressive than $L_{\text{S,M,C,NO}}$

To prove this claim, it suffices to prove that there exists no reasonable encoding of $L_{\text{S,M,C,PM}}$ in $L_{\text{S,M,C,NO}}$, since the latter is a sub-language of the former.

Theorem 4.3 *There exists no reasonable encoding of $L_{\text{S,M,C,PM}}$ in $L_{\text{S,M,C,NO}}$.*

Proof: Consider the process $[\![a(\ulcorner b \urcorner) \mid \overline{a}\langle b\rangle]\!]$, for $a \neq b$. Like in Theorem 4.2, it must be that $[\![a(\ulcorner b \urcorner)]\!] \xrightarrow{n?m}$ and $[\![\overline{a}\langle b\rangle]\!] \xrightarrow{(\nu \widetilde{m})n!m}$ (or vice versa, that is handled similarly), for some n, $\widetilde{m}$ and m; by Proposition 2.1(1), this fact implies that $[\![a(\ulcorner b \urcorner)]\!] \xrightarrow{n?l}$, for every l.

If $n \notin \varphi_{[\![\,]\!]}(b)$, then choose any $c \neq b$ such that $n \notin \varphi_{[\![\,]\!]}(c)$ and the permutation of names that swaps b and c; by name invariance, it holds that $[\![\overline{a}\langle c\rangle]\!] \xrightarrow{(\nu \widetilde{m'})n!m'}$, where m' and $\widetilde{m'}$ are the renamings of m and $\widetilde{m}$. Then, $[\![a(\ulcorner b \urcorner) \mid \overline{a}\langle c\rangle]\!] \xrightarrow{\tau}$, whereas $a(\ulcorner b \urcorner) \mid \overline{a}\langle c\rangle \not\xrightarrow{\tau}$. By Proposition 3.4, $[\![\cdot]\!]$ is not reasonable.

If $n \in \varphi_{[\![\,]\!]}(b)$, then choose $c \neq a$ such that $n \notin \varphi_{[\![\,]\!]}(c)$, the permutation of names that swaps a and c, and work like before, with process $[\![a(\ulcorner b \urcorner) \mid \overline{c}\langle b\rangle]\!]$. ∎

$L_{\text{S,P,C,NO}}$ and $L_{\text{S,M,C,NO}}$ have the same expressive power

Clearly, $L_{\text{S,M,C,NO}}$ is a sub-language of $L_{\text{S,P,C,NO}}$ and can be reasonably encoded in it. Also the converse holds, by exploiting, e.g., Milner's encoding of polyadic communications in monadic ones (see [29]). Notice that, for the latter encoding, it is crucial that $L_{\text{S,P,C,NO}}$ is typed. Without this assumption we can break reasonableness, as shown below.

Theorem 4.4 *There exists no reasonable encoding of the untyped $L_{\text{S,P,C,NO}}$ in $L_{\text{S,M,C,NO}}$.*

Proof: Consider the process $a(x,y) \mid \overline{a}\langle b,c\rangle$; again, $[\![\, \overline{a}\langle b,c\rangle \,]\!] \xrightarrow{(\nu\widetilde{d})n!d}$ and $[\![\, a(x,y) \,]\!] \xrightarrow{n?d}$ (or vice versa, that is handled similarly), for some n, d and $\widetilde{d}$. We consider the following sub-cases:

1. If $n \notin \varphi_{[\![\,]\!]}(a)$, choose $e \neq a$ with $n \notin \varphi_{[\![\,]\!]}(e)$; by name invariance, $[\![\, \overline{e}\langle b,c\rangle \,]\!] \xrightarrow{(\nu\widetilde{d'})n!d'}$ and $[\![\, a(x,y) \mid \overline{e}\langle b,c\rangle \,]\!] \xrightarrow{\tau}$, whereas $a(x,y) \mid \overline{e}\langle b,c\rangle \not\xrightarrow{\tau}$.

2. If $n \in \varphi_{[\![\,]\!]}(a)$, consider $a(x,y,z) \mid \overline{a}\langle b,c,c\rangle$; like before, $[\![\, \overline{a}\langle b,c,c\rangle \,]\!] \xrightarrow{(\nu\widetilde{e'})m!e}$ and $[\![\, a(x,y,z) \,]\!] \xrightarrow{m?e}$. Now,

   (a) if $m \notin \varphi_{[\![\,]\!]}(a)$, we work like in case 1 above;

   (b) if $m = n$, we have that $[\![\, a(x,y) \mid \overline{a}\langle b,c,c\rangle \,]\!] \xrightarrow{\tau}$;

   (c) otherwise, sequentially consider $a(x_1,\ldots,x_h) \mid \overline{a}\langle b_1,\ldots,b_h\rangle$, for $h > 3$, until either $[\![\, \overline{a}\langle b_1,\ldots,b_h\rangle \,]\!]$ outputs on a name not in $\varphi_{[\![\,]\!]}(a)$ or it outputs on a name already used by a $[\![\, \overline{a}\langle b_1,\ldots,b_{h'}\rangle \,]\!]$, for $h' < h$ (this surely happens since $\varphi_{[\![\,]\!]}$ associates to every name a k-tuple of names, for k fixed). In the first case, we fall in a situation similar to 2(a); in the second case, we fall in a situation similar to 2(b). ∎

$L_{\text{S,M,C,NO}}$ is more expressive than $L_{\text{S,M,D,PM}}$

To prove this result, we should show that $L_{\text{S,M,D,PM}}$ can be reasonably encoded in $L_{\text{S,M,C,NO}}$ and prove that the converse cannot hold. We start with the second task.

Theorem 4.5 *There exists no reasonable encoding of $L_{\text{S,M,C,NO}}$ in $L_{\text{S,M,D,PM}}$.*

Proof: By contradiction, assume that there exists a reasonable encoding $[\![\,\cdot\,]\!]$. Let a, b, c and d be pairwise distinct names, let Ω denote a divergent process and define $P \triangleq \textbf{if } x = d \textbf{ then } \Omega$. Operational correspondence and faithfulness imply that $[\![\, a(x).P \mid \overline{a}\langle b\rangle \,]\!]$ must perform at least one τ-step and reduce to (a process equivalent to) $[\![\, P\{^b/_x\} \,]\!]$. Clearly, in such a computation at least one name in $\varphi_{[\![\,]\!]}(b)$ must be transmitted, otherwise $[\![\, a(x).P \mid \overline{a}\langle d\rangle \,]\!]$ would reduce to $[\![\, P\{^b/_x\} \,]\!]$. To make the proof lighter, we shall assume that $|\varphi_{[\![\,]\!]}(\cdot)| = 1$ and let $\varphi_{[\![\,]\!]}(n) = n'$, for every name n; the general case can be obtained by adapting what follows to every component of $\varphi_{[\![\,]\!]}(b)$ transmitted in the computation leading $[\![\, a(x).P \mid \overline{a}\langle b\rangle \,]\!]$ to $[\![\, P\{^b/_x\} \,]\!]$.

By Proposition 3.5, it must be that $[\![\, a(x).P \,]\!]$ and $[\![\, \overline{a}\langle d\rangle \,]\!]$ communicate; thus, since $[\![\,\cdot\,]\!]$ is compositional, $C_{\mid}[\_1;\_2]$, the context used to compositionally translate the parallel composition operator, must be structurally equivalent to $(\nu\widetilde{r})(\_1 \mid \_2 \mid \hat{P})$, thanks to Proposition 3.5. By exploiting Proposition 2.1(2), we have that $[\![\, a(x).P \,]\!] \overset{\rho_1}{\Longrightarrow} R_1 \xrightarrow{?b'} R_2 \overset{\rho_2}{\Longrightarrow} R$ and $[\![\, \overline{a}\langle b\rangle \,]\!] \mid \hat{P} \overset{\overline{\rho_1}}{\Longrightarrow} R_3 \xrightarrow{!b'} R_4 \overset{\overline{\rho_2}}{\Longrightarrow} R'$,

for b' not occurring in ρ_1, $?b'$ generated by an input action with a formal template and $(\nu\widetilde{r},\widetilde{s}_1,\widetilde{s}_2)(R \mid R') \simeq [\![\, P\{b/x\} \,]\!]$, with $\widetilde{s}_i = \mathrm{BN}(\rho_i) \cup \mathrm{BN}(\overline{\rho_i})$ for $i \in \{1,2\}$. In particular, $\rho_2 \triangleq \phi_1 \cdot \ldots \cdot \phi_k$, for $\phi_i \in \{?n_i, (\nu\widetilde{n_i})!n_i\}$, and $\overline{\rho_2} \triangleq \overline{\phi}_1 \cdot \ldots \cdot \overline{\phi}_k$, for $\overline{\phi}_i = (\nu\widetilde{n_i})!n_i$, if $\phi_i = ?n_i$, and $\overline{\phi}_i = ?n_i$, otherwise.

Let σ be the permutation that swaps a with c and b with d. By name invariance, $[\![\, c(x).P\sigma \,]\!] \stackrel{\rho'_1}{\Longrightarrow} R_1\sigma' \stackrel{?d'}{\longrightarrow} R_2\sigma \stackrel{\rho'_2}{\Longrightarrow} R\sigma'$ and $[\![\, \overline{c}\langle d\rangle \,]\!] \mid \hat{P} \stackrel{\overline{\rho'_1}}{\Longrightarrow} R_3\sigma' \stackrel{!d'}{\longrightarrow} R_4\sigma \stackrel{\overline{\rho'_2}}{\Longrightarrow} R'\sigma'$, for $\rho'_1 = \rho_1\sigma'$ and $\rho'_2 = \rho_2\sigma'$; here σ' denotes the permutation of names induced by σ, as defined in Definition 3.2(2) (in the simplified case where $|\varphi_{[\![\,]\!]}(\cdot)| = 1$, σ' only swaps a' with c' and b' with d'). More precisely, $\rho'_2 \triangleq \phi'_1 \cdot \ldots \cdot \phi'_k$ and $\overline{\rho'_2} \triangleq \overline{\phi'_1} \cdot \ldots \cdot \overline{\phi'_k}$, for $\phi'_i \triangleq \phi_i\sigma'$ and $\overline{\phi'_i} \triangleq \overline{\phi}_i\sigma'$.

Now, consider $Q \triangleq (a(x).P \mid \overline{a}\langle b\rangle) \mid (\overline{c}\langle d\rangle \mid c(x).P\sigma)$; trivially, $Q \not\Uparrow$ whereas, as we shall see, $[\![\, Q \,]\!] \Uparrow$. This yields the desired contradiction. By compositionality, $[\![\, Q \,]\!]$ is structurally equivalent to

$$(\nu\widetilde{r})(\ (\nu\widetilde{r})([\![\, a(x).P \,]\!] \mid [\![\, \overline{a}\langle b\rangle \,]\!] \mid \hat{P}) \ \mid\ (\nu\widetilde{r})([\![\, c(x).P\sigma \,]\!] \mid ([\![\, \overline{c}\langle d\rangle \,]\!] \mid \hat{P}) \ \mid \hat{P})$$

Then, consider

$$\begin{aligned}[\![\, Q \,]\!] &\Rightarrow (\nu\widetilde{r})((\nu\widetilde{r},\widetilde{s}_1)(R_1 \mid R_3) \mid (\nu\widetilde{r},\widetilde{s}_1)(R_1\sigma' \mid R_3\sigma') \mid \hat{P}) \\ &\rightarrow\!\!\rightarrow (\nu\widetilde{r})((\nu\widetilde{r},\widetilde{s}_1)(R_2\{d'/b'\} \mid R_4) \mid (\nu\widetilde{r},\widetilde{s}_1)((R_2\sigma')\{b'/d'\} \mid R_4\sigma') \mid \hat{P})\end{aligned}$$

where R_1 received d' in place of b' and $R_1\sigma'$ received b' in place of d' (this is possible since these inputs do not rely on actual templates). Now, $R_2\{d'/b'\} \stackrel{\phi''_1 \cdot \ldots \cdot \phi''_k}{\Longrightarrow}$, where $\phi''_i \triangleq \phi'_i\{d'/b'\}$, and $(R_2\sigma)\{b'/d'\} \stackrel{\overline{\phi''_1} \cdot \ldots \cdot \overline{\phi''_k}}{\Longrightarrow}$, where $\overline{\phi''_i} \triangleq \overline{\phi'_i}\{b'/d'\}$. Finally, consider the computation

$$\begin{aligned}&(\nu\widetilde{r})((\nu\widetilde{r},\widetilde{s}_1)(R_2\{d'/b'\} \mid R_4) \mid (\nu\widetilde{r},\widetilde{s}_1)((R_2\sigma')\{b'/d'\} \mid R_4\sigma') \mid \hat{P}) \\ &\quad\Rightarrow\ (\nu\widetilde{r})((\nu\widetilde{r},\widetilde{s}_1,\widetilde{s}_2)(R\{d'/b'\} \mid R'\{d'/b'\}) \mid (\nu\widetilde{r},\widetilde{s}_1,\widetilde{s}_2)((R\sigma')\{b'/d'\} \mid (R'\sigma')\{b'/d'\}) \mid \hat{P})\end{aligned}$$

obtained by synchronizing

- ϕ''_i with $\overline{\phi}_i$ and $\overline{\phi''_i}$ with $\overline{\phi'_i}$, if $b' \notin \mathrm{N}(\phi_i)$ (and hence $d' \notin \mathrm{N}(\phi'_i)$), or
- ϕ''_i with $\overline{\phi'_i}$ and $\overline{\phi''_i}$ with $\overline{\phi}_i$, otherwise.

Now, $(\nu\widetilde{r},\widetilde{s}_1,\widetilde{s}_2)((R\sigma')\{b'/d'\} \mid (R'\sigma')\{b'/d'\}) \triangleq ((\nu\widetilde{r},\widetilde{s}_1,\widetilde{s}_2)(R \mid R'))(\{b'/d'\} \circ \sigma') \simeq [\![\, P\{b/x\} \,]\!]\{b'/d', b'/b', a'/c', c'/a'\} = [\![\, \textbf{if } b = b \textbf{ then } \Omega \,]\!]$, that is a divergent process. ∎

Let us now consider the possibility of reasonably encoding $L_{\mathrm{S,M,D,PM}}$ in $L_{\mathrm{S,M,C,NO}}$; this task is more problematic and, indeed, we have still not been able to develop a reasonable encoding, nor to prove an impossibility result. We now present two possible (but not fully satisfactory) encodings that should give a feeling of the encodability of $L_{\mathrm{S,M,D,PM}}$ in $L_{\mathrm{S,M,C,NO}}$.

If we had assumed non-deterministic choice in our languages, a reasonable encoding could have been obtained by translating all the operators homomorphically, except for

- $\langle b\rangle.P$, that is translated into $\overline{\mathtt{ether}}\langle b\rangle.[\![\, P \,]\!] \ +\ \overline{b}\langle b\rangle.[\![\, P \,]\!]$;
- $(x).P$, that is translated into $\mathtt{ether}(x).[\![\, P \,]\!]$; and
- $(\ulcorner b \urcorner).P$, that is translated into $b(y).[\![\, P \,]\!]$.

Here, `ether` is a reserved name and '+' denotes non-deterministic choice between two processes. Intuitively, $[\![\langle b \rangle.P]\!]$ must enable two different kinds of input: a 'generic' input, viz. $(x).P$, that receives b (via the channel `ether`, that models the shared dataspace), and an 'exact' input, viz. $(\ulcorner b \urcorner).P$, in which b is only used for testing purposes. This latter kind of interaction can be naturally implemented via the channel-based communication of $L_{\text{S,M,C,NO}}$ in which the exchanged datum is useless. If we extend $\Lambda_{s,a,m,p}$ with (guarded) choice, the encoding just described is reasonable (this is easy to prove, by exploiting Proposition 3.3 and the fact that $(\nu c)(\overline{c}\langle b \rangle \mid c(x).P) \gtrsim P\{^b/_x\}$ holds in $L_{\text{S,M,C,NO}}$). However, non-deterministic choice in an asynchronous setting is usually omitted; hence, for the sake of uniformity, we prefer to leave $\Lambda_{s,a,m,p}$ without it.

Let us now try to adapt the philosophy underlying the encoding just described to a setting without non-deterministic choice; to this aim, we shall exploit ideas from [32, 34]. Let `ether` be a reserved name that can be isolated in the following way: (1) linearly order the set of names $\mathcal{N}$ as $\{n_0, n_1, n_2, \ldots\}$; (2) let $\varphi_{[\![\ ]\!]}$ map n_i to n_{i+1}, for every i; (3) the reserved name `ether` is n_0. For the sake of presentation, we shall not explicitly use this renaming policy in the presentation of the encoding, but it implicitly holds. The encoding translates all the operators homomorphically, except for:

$$
\begin{array}{rcll}
[\![\langle b \rangle.P]\!] & \triangleq & (\nu c, d_1, d_2, g)(\ \overline{c}\langle b \rangle \ \mid \overline{\texttt{ether}}\langle c, d_1, d_2, g \rangle & \\
 & & \quad \mid \overline{b}\langle c, d_1, d_2, g \rangle \mid g().[\![P]\!]) & \text{for } c, d_1, d_2, g \text{ fresh} \\
[\![(x).P]\!] & \triangleq & \texttt{ether}(y, z_1, z_2, w).(\ y(x).(\overline{z_2}\langle\rangle \mid \overline{w}\langle\rangle.[\![P]\!]) & \\
 & & \quad \mid z_1().[\![(x).P]\!]) & \text{for } y, z_1, z_2, w \text{ fresh} \\
[\![(\ulcorner b \urcorner).P]\!] & \triangleq & b(y, z_1, z_2, w).(y(x).(\overline{z_1}\langle\rangle \mid \overline{w}\langle\rangle.[\![P]\!]) \ \mid \ z_2().[\![(\ulcorner b \urcorner).P]\!]) & \text{for } x, y, z_1, z_2, w \text{ fresh}
\end{array}
$$

For the sake of presentation, we have used polyadic communications and recursive process definitions that, however, can be easily implemented in $L_{\text{S,M,C,NO}}$. Intuitively, the output along c is used for choosing whether the encoding of an output interacts with the encoding of a formal or of an actual input; in the former case, the datum is used for replacing x with b; in the latter case, the datum is useless. Channels d_1 and d_2 are used to properly activate a continuation process: if there is an output available along d_1, then a formal input has succeeded and any other actual input must be restored; the situation is symmetric whenever there is an output along d_2. Finally, channel g is used to unleash the continuation of the output process.

The problem of this encoding is that it does not enjoy operational correspondence as formulated in Definition 3.2. Indeed, we have that (the case with an actual input is symmetric):

$$[\![\langle b \rangle \mid (x)]\!] \Rightarrow (\nu c, d_1, d_2, g)(\overline{d_2}\langle\rangle \ \mid \ \overline{b}\langle c, d_1, d_2, g \rangle \mid d_1().[\![(x)]\!]) \triangleq R$$

and R is not barbed congruent to $[\![\langle b \rangle \mid (x)]\!]$ nor to $[\![\mathbf{0}]\!]$ (of course, a similar problem also arises in the converse direction of operational correspondence, viz. Definition 3.2(4a)). However, R does not affect the behaviour of any encoded term. Indeed, whenever put in parallel with $[\![P]\!]$, its presence is either transparent to $[\![P]\!]$ or, if R interacts with $[\![P]\!]$, then P evolves to a process with a parallel component starting with $(\ulcorner b \urcorner)$; but in that case, R annihilates itself in two τ-steps and restores the encoding of $(\ulcorner b \urcorner)$. It is worth noting that also the encodings of the separate and of the input-guarded choice π-calculus into the asynchronous π-calculus [32, 34] suffer from similar problems. For these reasons, we believe that the encoding we have just presented testifies to the fact that $L_{\text{S,M,D,PM}}$ can be encoded in $L_{\text{S,M,C,NO}}$ (for this reason we put a dashed arrow from the former to the latter in Figures 1 and 2); however, a definitive answer to the possibility of reasonably encoding the former in the latter is still missing.

$L_{\text{S,M,C,NO}}$ is more expressive than $L_{\text{S,P,D,NO}}$

We start with a reasonable encoding of $L_{\text{S,P,D,NO}}$ in $L_{\text{S,M,C,NO}}$. The only feature of $L_{\text{S,P,D,NO}}$ is that it can check the arity of a datum before retrieving it (see the definition of function Match). This, however, can be mimicked by the channel-based communication of $L_{\text{S,M,C,NO}}$ by assuming a reserved channel for every possible arity: a datum of arity k will be represented as an output over channel $\mathtt{k}$; an input of arity k will be represented as an input from $\mathtt{k}$; a communication over $\mathtt{k}$ in $L_{\text{S,M,C,NO}}$ can happen if and only if pattern-matching succeeds in $L_{\text{S,P,D,NO}}$; finally, the exchanged datum is a restricted name that will be used for the actual data exchange.

The encoding assumes that $\mathtt{0}, \mathtt{1}, \ldots, \mathtt{k}, \ldots$ are reserved names, that can be obtained as expected: (1) linearly order the set of names $\mathcal{N}$ as $\{n_0, n_1, n_2, \ldots\}$; (2) let $\varphi_{[\![\ ]\!]}$ map n_i to n_{2i+1}, for every i; (3) the generic reserved name $\mathtt{k}$ is n_{2k}.

$$
\begin{array}{lcll}
[\![\ \langle b_1, \cdots, b_k \rangle.P\ ]\!] & \triangleq & (\nu n)\, \overline{\mathtt{k}}\langle n \rangle.\overline{n}\langle b_1 \rangle.\overline{n}\langle b_2 \rangle. \cdots .\overline{n}\langle b_k \rangle.[\![\ P\ ]\!] & \text{for } n \text{ fresh} \\
[\![\ (x_1, \cdots, x_k).P\ ]\!] & \triangleq & \mathtt{k}(x).x(x_1).x(x_2). \cdots .x(x_k).[\![\ P\ ]\!] & \text{for } x \text{ fresh}
\end{array}
$$

Also here, for the sake of simplicity, the renaming policy is kept implicit in the presentation of the encoding.

Reasonableness of this encoding[2] can be easily proved, by exploiting Proposition 3.3 and the fact that $(\nu c)(\overline{c}\langle b \rangle \mid c(x).P) \gtrsim P\{^b/_x\}$ holds in $L_{\text{S,M,C,NO}}$. We now have to prove that the converse is not possible.

Theorem 4.6 *There exists no reasonable encoding of $L_{\text{S,M,C,NO}}$ in $L_{\text{S,P,D,NO}}$.*

Proof: We start with process $\overline{a}\langle b \rangle \mid a(x)$, for $a \neq b$; it holds that $[\![\ \overline{a}\langle b \rangle\ ]\!] \xrightarrow{(\nu \widetilde{c'})!\widetilde{c}}$ and $[\![\ a(x)\ ]\!] \xrightarrow{?\widetilde{c}}$ (or vice versa, that is similar), for some $\widetilde{c'}$ and $\widetilde{c}$. By name invariance, $[\![\ \overline{b}\langle a \rangle\ ]\!] \xrightarrow{(\nu \widetilde{d'})!\widetilde{d}}$, where $\widetilde{d}$ and $\widetilde{d'}$ are obtained by ordinately swapping $\varphi_{[\![\ ]\!]}(a)$ and $\varphi_{[\![\ ]\!]}(b)$ in $\widetilde{c}$ and $\widetilde{c'}$; thus, $|\widetilde{d}| = |\widetilde{c}|$. Now, by Proposition 2.1(1), $[\![\ a(x)\ ]\!] \xrightarrow{?\widetilde{d}}$; hence, $[\![\ \overline{b}\langle a \rangle \mid a(x)\ ]\!] \xrightarrow{\tau}$, whereas $\overline{b}\langle a \rangle \mid a(x) \stackrel{\tau}{\not\rightarrow}$. This suffices to conclude. ∎

$L_{\text{S,M,D,PM}}$ and $L_{\text{S,P,D,NO}}$ are incomparable

To prove this claim, we must show the impossibility of a reasonable encoding of $L_{\text{S,M,D,PM}}$ in $L_{\text{S,P,D,NO}}$ and vice versa.

Theorem 4.7 *There exists no reasonable encoding of $L_{\text{S,M,D,PM}}$ in $L_{\text{S,P,D,NO}}$.*

Proof: Easily derivable from the proof of Theorem 4.6, by using process $\langle a \rangle \mid (\ulcorner a \urcorner)$. ∎

Theorem 4.8 *There exists no reasonable encoding of $L_{\text{S,P,D,NO}}$ in $L_{\text{S,M,D,PM}}$.*

[2] This encoding enjoys full abstraction with respect to barbed congruence restricted to the translation of $L_{\text{S,M,C,NO}}$-contexts. This is quite an expectable property because it states that contexts abiding by the protocol put forward by the encoding cannot distinguish the translation of equivalent source language terms. A more liberal property consists in defining a type system that characterizes the contexts abiding by the protocol of the encoding and proving a *typed* full abstraction result, in the same vein as, e.g., [41]; we believe that such a result holds for this encoding, though we have not spelled the details out.

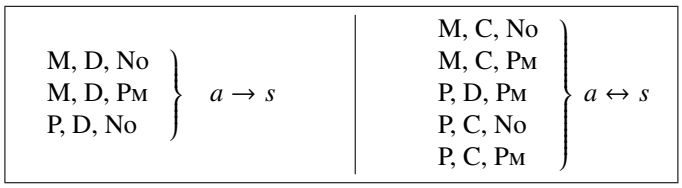

Figure 3: The Expressiveness of Synchrony

Proof: The proof is similar to that of Theorem 4.5. Assume that $[\![\cdot]\!]$ is reasonable; consider the process $P \triangleq (x, y).\textbf{if } x = a \textbf{ then if } y = d \textbf{ then } \Omega$; choose $c \neq a$ and $b \neq d$; consider the permutation of names σ swapping a with c and b with d; finally, show that $Q \triangleq (P \mid \langle a, b \rangle) \mid (P\sigma \mid \langle c, d \rangle)$ is not divergent, whereas $[\![Q]\!] \Uparrow$. ∎

$L_{\text{S,M,D,PM}}$ and $L_{\text{S,P,D,NO}}$ are more expressive than $L_{\text{S,M,D,NO}}$

Clearly, $L_{\text{S,M,D,NO}}$ is a sub-language of both $L_{\text{S,M,D,PM}}$ and $L_{\text{S,P,D,NO}}$; of course, it can be reasonably encoded into them. The converse does not hold, as proved in the following theorem.

Theorem 4.9 *There exist no reasonable encodings of $L_{\text{S,M,D,PM}}$ and $L_{\text{S,P,D,NO}}$ in $L_{\text{S,M,D,NO}}$.*

Proof: Easy consequence of Theorems 4.7 and 4.8. ∎

5 Adding Asynchronous Communications

We now extend the hierarchy in Figure 2 by adding asynchronous communication primitives. We start by considering those languages in which synchrony does not play a crucial role (i.e., the asynchronous versions of the primitives have the same expressive power as the synchronous ones). We then move to analyze those primitives in which the presence of synchrony matters (i.e., the asynchronous versions of the primitives are less expressive than the synchronous ones). Finally, we give some more results needed to properly place all the asynchronous primitives in the hierarchy.

Our results are summarized in Figure 3. There, $a \rightarrow s$ means that the asynchronous version of the primitive can be reasonably encoded in its synchronous counterpart but not vice versa, whereas $a \leftrightarrow s$ means that the two versions have the same expressive power (i.e., one can be encoded in the other).

It is evident that channels are the only features that ensure reasonable encodings of synchrony in asynchrony: reserved channels can be used for synchronization purposes. The only exception seems to be $L_{\text{S,P,D,PM}}$, that can be reasonably encoded in $L_{\text{A,P,D,PM}}$: however, $L_{\text{A,P,D,PM}}$ can encode channels (since it is more expressive than $L_{\text{A,M,C,NO}}$). On the contrary, the remaining dataspace-based languages 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., Mobile Ambients [14] or CCS [28]) cannot freely interchange their synchronous and asynchronous versions.

5.1 When Synchrony does not Matter

$L_{\text{S,P,C,PM}}$ and $L_{\text{A,P,C,PM}}$ have the same expressive power

Clearly, $L_{\text{A,P,C,PM}}$ can be seen as a sub-language of $L_{\text{S,P,C,PM}}$, see Remark 2.1; we now prove that $L_{\text{S,P,C,PM}}$ can be reasonably encoded in $L_{\text{A,P,C,PM}}$. It suffices to let the first name of every datum be a restricted channel used to unleash the continuation of the output prefix; conversely, every template starts with a new variable over which an acknowledgment is sent upon reception of the datum. This discipline is rendered by the following encoding:

$$\begin{array}{lcll} [\![\, \overline{a}\langle\widetilde{b}\rangle.P \,]\!] & \triangleq & (\nu c)(\overline{a}\langle c, \widetilde{b}\rangle \mid c(\,).[\![\, P \,]\!]) & \text{for } c \text{ fresh} \\ [\![\, a(\widetilde{T}).P \,]\!] & \triangleq & a(x, \widetilde{T}).(\overline{x}\langle\rangle \mid [\![\, P \,]\!]) & \text{for } x \text{ fresh} \end{array}$$

The proof of reasonableness[3] relies on Proposition 3.3 and on the fact that in $L_{\text{A,P,C,PM}}$ it holds that $(\nu c)(\overline{c}\langle\rangle \mid c(\,).P) \gtrsim P$.

$L_{\text{S,P,D,PM}}$ and $L_{\text{A,P,D,PM}}$ have the same expressive power

Again, $L_{\text{A,P,D,PM}}$ can be seen as a sub-language of $L_{\text{S,P,D,PM}}$, see Remark 2.1; we now prove that $L_{\text{S,P,D,PM}}$ can be reasonably encoded in $L_{\text{A,P,D,PM}}$. Consider the following translation:

$$\begin{array}{lcll} [\![\, \langle b_1, \ldots, b_k\rangle.P \,]\!] & \triangleq & (\nu c)(\langle c, c, b_1, \ldots, b_k\rangle \mid (\ulcorner c \urcorner).[\![\, P \,]\!]) & \text{for } c \text{ fresh} \\ [\![\, (T_1, \ldots, T_k).P \,]\!] & \triangleq & (x, y, T_1, \ldots, T_k).(\langle x\rangle \mid [\![\, P \,]\!]) & \text{for } x, y \text{ fresh} \end{array}$$

Intuitively, data of length one in a translated term are 'auxiliary' messages used as acknowledgments that 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 that no interference between an 'actual' data exchange and an 'auxiliary' acknowledgment exchange can ever happen. Moreover, the fact that acknowledgments rely on restricted names rules out interferences between them.

Also for this encoding, the proof of reasonableness[4] relies on Proposition 3.3 and on the fact that in $L_{\text{A,P,D,PM}}$ it holds that $(\nu c)(\langle c\rangle \mid (\ulcorner c \urcorner).P) \gtrsim P$.

$L_{\text{S,P,C,NO}}$ and $L_{\text{A,P,C,NO}}$ have the same expressive power

This fact is an easy corollary of the encodability of $L_{\text{S,P,C,PM}}$ in $L_{\text{A,P,C,PM}}$: it suffices to restrict both the domain and the range of the encoding function to the sub-calculi of $L_{\text{S,P,C,PM}}$ and $L_{\text{A,P,C,PM}}$ with formal templates only.

$L_{\text{S,M,C,NO}}$ and $L_{\text{A,M,C,NO}}$ have the same expressive power

On one hand, $L_{\text{A,M,C,NO}}$ can be seen as a sub-language of $L_{\text{S,M,C,NO}}$; on the other hand, $L_{\text{S,M,C,NO}}$ can be reasonably encoded in $L_{\text{A,M,C,NO}}$, see [5, 26].[5]

[3] In [24], we proved full abstraction with respect to barbed congruence restricted to the translation of $L_{\text{S,P,C,PM}}$-contexts; moreover, we conjecture that such a result can be extended to full abstraction with respect to typed barbed congruence.

[4] In [24] we proved that it enjoys full abstraction with respect to barbed congruence restricted to the translation of $L_{\text{S,P,D,PM}}$-contexts; a similar result should hold also in terms of typed barbed congruence.

[5] The first encoding also enjoys full abstraction with respect to barbed congruence restricted to translated $L_{\text{S,M,C,NO}}$-contexts and typed barbed congruence, as proved in [11, 41].

$L_{\text{S,M,C,PM}}$ and $L_{\text{A,M,C,PM}}$ have the same expressive power

Trivially, $L_{\text{A,M,C,PM}}$ can be encoded in $L_{\text{S,M,C,PM}}$. The converse can be proved by means of the following encoding. First, assume the renaming policy $\varphi_{[\![\ ]\!]}$ that maps every name a to a triple of names that we symbolically denote a_N, a_0, a_1: a_N represents the 'name' of the channel, whereas a_0 and a_1 are used for synchronization purposes. Then, encode all the operators homomorphically, except for

$$
\begin{aligned}
[\![\ \overline{a}\langle b\rangle.P\ ]\!] &\triangleq \overline{a_0}\langle b_N\rangle \mid a_1(\ulcorner b_N \urcorner).(\overline{b_N}\langle b_0, b_1\rangle \mid [\![\ P\ ]\!]) \\
[\![\ a(x).P\ ]\!] &\triangleq a_0(x_N).(\overline{a_1}\langle x_N\rangle \mid x_N(x_0, x_1).[\![\ P\ ]\!]) \\
[\![\ a(\ulcorner b \urcorner).P\ ]\!] &\triangleq a_0(\ulcorner b_N \urcorner).(\overline{a_1}\langle b_N\rangle \mid b_N(x_0, x_1).[\![\ P\ ]\!]) \qquad \text{for } x_0, x_1 \text{ fresh} \\
[\![\ (\nu a)P\ ]\!] &\triangleq (\nu\, a_N, a_0, a_1)\ [\![\ P\ ]\!] \\
[\![\ \textbf{if } a = b \textbf{ then } P \textbf{ else } Q\ ]\!] &\triangleq \textbf{if } a_N = b_N \textbf{ then } [\![\ P\ ]\!] \textbf{ else } [\![\ Q\ ]\!]
\end{aligned}
$$

Notice that polyadic communications are just a shortcut: Honda and Tokoro's [26] encoding of polyadic asynchronous channel-based communication (without pattern matching) into monadic exchanges can be exploited here.

Intuitively, the output on a_0 signals the existence of an output, that can be consumed either by (the encoding of) a formal input or by (the encoding of) an actual input. In the first case, the argument of the action is used to (partially) instantiate the input variable; in the second case, the argument is used for matching purposes. The following two communications are used to activate the continuation processes; moreover, the last one is also needed to complete the instantiation of the input variable, when a formal input is involved. Notice the similarities between this encoding and the one in [5]; we just want to remark that restricted channels are not needed here, thanks to pattern-matching.

To prove reasonableness of this encoding, we cannot rely on Proposition 3.3 because the encoding we have just presented does not satisfy (1); this will require more work for proving operational correspondence and divergence reflection. To this aim, let us fix a simplifying notation: A_0 and $\overline{A}_0$ will denote the starting input and output of the encoding of a communication (i.e., the transmission of b_N along a_0, for some a and b); A_1 and $\overline{A}_1$ will denote the successive input and output (viz., along a_1); and B_N and $\overline{B}_N$ will denote the final input and output (viz., along b_N). To be precise, B_N and $\overline{B}_N$ are sequences of message exchanges but, since they are confluent, we can be sloppy on this point. Moreover, let us denote with $\#_0$ the number of synchronizations between some actions of kind A_0 and $\overline{A}_0$ in a given computation; $\#_1$ and $\#_N$ are defined in a similar way. The crucial lemma that will enable us to prove reasonableness now follows.

Lemma 5.1 *Let* $[\![\ P\ ]\!] \xrightarrow{\tau}{}^{h} Q$*, then:*

1. $Q \equiv (\nu \widetilde{u})(\ [\![\ P_0\ ]\!] \ \mid\ \prod_{i=1}^{m} A_1^i.(\overline{B}_N^i \mid [\![\ P_1^i\ ]\!]) \ \mid\ \prod_{j=1}^{m+n} B_N^j.R_2^j \ \mid\ \prod_{p=1}^{m} \overline{A}_1^p \ \mid\ \prod_{q=1}^{n} \overline{B}_N^q\ )$ *where* $\prod_{r=1}^{n} \cdots$ *denotes the parallel composition of n processes '*$\cdots$*'; moreover, we let* $R_2^j \triangleq [\![\ P_2^j\ ]\!]\{b_N/x_N\}$ *whenever* $B_N^j = b_N(x_0, x_1)$ *and* $x \in \text{Fn}(P_2^j)$*, and we let* $R_2^j \triangleq [\![\ P_2^j\ ]\!]$*, otherwise.*

2. $P \xrightarrow{\tau}{}^{\#_0} (\nu \widetilde{v})(P_0 \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j)$*, for* $\widetilde{u} = \varphi_{[\![\ ]\!]}(\widetilde{v})$*; moreover,* $\sigma_j = \{b/x\}$*, whenever* $B_N^j = b_N(x_0, x_1)$ *and* $x \in \text{Fn}(P_2^j)$*, and* $\sigma_j = \epsilon$*, otherwise.*

Proof: Let us call Q_1 the sub-process $\prod_{i=1}^{m}$, Q_2 the sub-process $\prod_{j=1}^{m+n}$, Q_3 the sub-process $\prod_{p=1}^{m}$ and Q_4 the sub-process $\prod_{q=1}^{n}$. Both claims are proved by induction on h. The base case ($h = 0$) is trivial. For the inductive case, let $[\![P]\!] \xrightarrow{\tau}{}^{h-1} Q' \xrightarrow{\tau} Q$; by induction,

$$Q' \equiv (\nu\widetilde{u})([\![P_0]\!] \mid \prod_{i=1}^{m} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+n} B_N^j.R_2^j \mid \prod_{p=1}^{m} \overline{A}_1^p \mid \prod_{q=1}^{n} \overline{B}_N^q)$$

Moreover, $P \xrightarrow{\tau}{}^{k} (\nu\widetilde{v})(P_0 \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j)$, where k is $\#_0$ referred to the computation $[\![P]\!] \xrightarrow{\tau}{}^{h-1} Q'$. We consider all the possible ways in which $Q' \xrightarrow{\tau} Q$ can be generated.

1. It is generated by $[\![P_0]\!]$: this is possible only if it is a synchronization between some A_0 and $\overline{A}_0$; thus, $[\![P_0]\!] \equiv [\![P_0']\!] \mid [\![\overline{a}\langle b\rangle.P_3]\!] \mid [\![a(T).P_4]\!]$, for some T matching against b. Then,

$$Q \equiv (\nu\widetilde{u})([\![P_0']\!] \mid \prod_{i=1}^{m+1} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+1+n} B_N^j.R_2^j \mid \prod_{p=1}^{m+1} \overline{A}_1^p \mid \prod_{q=1}^{n} \overline{B}_N^q)$$

   where the $(m+1)$-th component of Q_1 is $a_1(\ulcorner b_N \urcorner).(\overline{b_N}\langle b_0, b_1\rangle \mid [\![P_3]\!])$, the $(m+1)$-th component of Q_3 is $\overline{a_1}\langle b_N\rangle$ and the $(m + 1 + n)$-th component of Q_2 is $b_N(x_0, x_1).R_2^{m+1+n}$, with $R_2^{m+1+n} = [\![P_4]\!]$ if $T = \ulcorner b \urcorner$ and $R_2^{m+1+n} = [\![P_4]\!]\{^{b_N}/_{x_N}\}$ if $T = x$. Then,

$$P \xrightarrow{\tau}{}^{k} (\nu\widetilde{v})(P_0 \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j) \xrightarrow{\tau} (\nu\widetilde{v})(P_0' \mid P_3 \mid P_4\sigma \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j)$$

   for $\sigma = \textsc{Match}(T; b)$; moreover, $k + 1$ is $\#_0$ in the computation $[\![P]\!] \xrightarrow{\tau}{}^{h} Q$.

2. It is a synchronization between Q_1 and Q_3: in this case,

$$Q \equiv (\nu\widetilde{u})([\![P_0]\!] \mid [\![P_1^m]\!] \mid \prod_{i=1}^{m-1} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+n} B_N^j.R_2^j \mid \prod_{p=1}^{m-1} \overline{A}_1^p \mid \prod_{q=1}^{n+1} \overline{B}_N^q)$$

   where the $(n + 1)$-th component of Q_4 is $\overline{B}_N^m$ and the first claim holds because $n + m = (n + 1) + (m - 1)$. The second claim, holds by inductive hypothesis and by the fact that k is $\#_0$ in the computation $[\![P]\!] \xrightarrow{\tau}{}^{h} Q$.

3. It is a synchronization between Q_2 and Q_4: this case is similar to the previous one; just notice that now

$$Q \equiv (\nu\widetilde{u})([\![P_0]\!] \mid R \mid \prod_{i=1}^{m} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+n-1} B_N^j.R_2^j \mid \prod_{p=1}^{m} \overline{A}_1^p \mid \prod_{q=1}^{n-1} \overline{B}_N^q)$$

   where R is $[\![P_2^{m+n}\{^b/_x\}]\!]$, if $R_2^{m+n} = [\![P_2^{m+n}]\!]\{^{b_N}/_{x_N}\}$, and is $[\![P_2^{m+n}]\!]$, otherwise.

4. It is a synchronization between Q_3 and $[\![P_0]\!]$: in this case, $[\![P_0]\!]$ exhibits at top-level the encoding of an output involving the same names (both of the channel and of the argument) as some of the $\overline{A}_1$ in Q_3; correspondingly, there must be a component in Q_2

starting with a formal input over the argument name. Moreover, such an $\overline{A}_1$ belongs to Q_3 because it communicated with some output with the same names. Thus, $[\![P_0]\!] \equiv [\![P'_0]\!] \mid [\![\overline{a^m}\langle b^m\rangle.\hat{P}]\!]$, $Q_1 \equiv a_1^m(\ulcorner b_N^m \urcorner).(\overline{b_N^m}\langle b_0^m, b_1^m\rangle \mid [\![P_1^m]\!]) \mid \prod_{i=1}^{m-1} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!])$, $Q_2 \equiv b_N^{m+n}(x_0^{m+n}, x_1^{m+n}).R_2^{m+n} \mid \prod_{j=1}^{m+n-1} B_N^j.R_2^j$ and $Q_3 \equiv \overline{a_1^m}\langle b_N^m\rangle \mid \prod_{p=1}^{m-1} \overline{A}_1^p$. Now,

$$Q \equiv (\nu\widetilde{u})([\![P'_0 \mid \hat{P} \mid \overline{a^m}\langle b^m\rangle.P_1^m]\!] \mid \prod_{i=1}^{m-1} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+n} B_N^j.R_2^j \mid \prod_{p=1}^{m-1} \overline{A}_1^p \mid \prod_{q=1}^{n+1} \overline{B}_N^q)$$

Indeed, the $\overline{a_0^m}\langle b_N^m\rangle$ left by $[\![\overline{a^m}\langle b^m\rangle.\hat{P}]\!]$ after the synchronization can be joined with $a_1^m(\ulcorner b_N^m \urcorner).(\overline{b_N^m}\langle b_0^m, b_1^m\rangle \mid [\![P_1^m]\!])$ to restore the encoding of $\overline{a^m}\langle b^m\rangle.P_1^m$; moreover, $[\![\hat{P}]\!]$ is unleashed and the associated output $\overline{b_N^m}\langle b_0^m, b_1^m\rangle$ becomes the $(n+1)$-th component of Q_4.

Concerning the second claim, we have that the $(m+n)$-th component of Q_2 came from an input $a^m(T).P_2^{m+n}$, for some T such that $\textsc{Match}(T; b) = \sigma_{m+n}$ and $R_2^{m+n} = P_2^{m+n}$, if $T = \ulcorner b \urcorner$, while $R_2^{m+n} = P_2^{m+n}\{b_N/x_N\}$, if $T = x$. Thus, we have that $P \xrightarrow{\tau}{}^{k_1} (\nu\widetilde{v'})(\overline{a^m}\langle b^m\rangle.P_1^m \mid a^m(T).P_2^{m+n} \mid P') \xrightarrow{\tau}{}^{k_2} (\nu\widetilde{v})(P_0 \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j)$, for $k = k_1 + k_2$. Now, notice that both $[\![P_1^m]\!]$ and R_2^{m+n} are blocked in Q and, thus, they cannot contribute to $[\![P]\!] \Rightarrow Q$; thus,

$$\begin{aligned} P &\xrightarrow{\tau}{}^{k-1} (\nu\widetilde{v})(\overline{a^m}\langle b^m\rangle.P_1^m \mid \overline{a^m}\langle b^m\rangle.\hat{P} \mid a^m(T).P_2^{m+n} \mid P'_0 \mid \prod_{i=1}^{m-1} P_1^i \mid \prod_{j=1}^{m+n-1} P_2^j\sigma_j) \\ &\xrightarrow{\tau} (\nu\widetilde{v})(\overline{a^m}\langle b^m\rangle.P_1^m \mid \hat{P} \mid P_2^{m+n}\sigma_{m+n} \mid P'_0 \mid \prod_{i=1}^{m-1} P_1^i \mid \prod_{j=1}^{m+n-1} P_2^j\sigma_j) \\ &\equiv (\nu\widetilde{v})(P'_0 \mid \hat{P} \mid \overline{a^m}\langle b^m\rangle.P_1^m \mid \mid \prod_{i=1}^{m-1} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j) \end{aligned}$$

and we can conclude because k is $\#_0$ also in the computation $[\![P]\!] \Rightarrow Q$. ∎

Proposition 5.2 *The encoding* $[\![\cdot]\!] : L_{\text{S,M,C,PM}} \longrightarrow L_{\text{A,M,C,PM}}$ *is reasonable.*

Proof: Definition 3.2(4b) is a corollary of Lemma 5.1. Indeed, $[\![P]\!] \Rightarrow Q$ implies, by Lemma 5.1(1), that $Q \equiv (\nu\widetilde{u})([\![P_0]\!] \mid \prod_{i=1}^{m} A_1^i.(\overline{B}_N^i \mid [\![P_1^i]\!]) \mid \prod_{j=1}^{m+n} B_N^j.R_2^j \mid \prod_{p=1}^{m} \overline{A}_1^p \mid \prod_{q=1}^{n} \overline{B}_N^q)$; thus, $Q \Rightarrow (\nu\widetilde{u})([\![P_0]\!] \mid \prod_{i=1}^{m} [\![P_1^i]\!] \mid \prod_{j=1}^{m+n} [\![P_2^j\sigma_j]\!]) \triangleq [\![(\nu\widetilde{v})(P_0 \mid \prod_{i=1}^{m} P_1^i \mid \prod_{j=1}^{m+n} P_2^j\sigma_j)]\!]$ and the claim holds because of Lemma 5.1(2).

Let us now prove that the encoding does not introduce divergence; the remaining reasonableness requirements can be routinely proved. Assume that $[\![P]\!]$ diverges. By definition, $[\![P]\!]$ must perform an infinite computation and, hence, $\#_0$ is infinite; indeed, by construction of the encoding, in every given computation it holds that $\#_0 \geq \#_1$ and $\#_0 \geq \#_N$. By Lemma 5.1(2), also P diverges. ∎

5.2 When Synchrony Matters

$L_{\text{S,M,D,NO}}$ is more expressive than $L_{\text{A,M,D,NO}}$

Trivially, $L_{\text{A,M,D,NO}}$ can be encoded in $L_{\text{S,M,D,NO}}$. The converse is impossible, as a corollary of Theorem 5.3 later on.

$L_{\text{S,M,D,PM}}$ is more expressive than $L_{\text{A,M,D,PM}}$

Trivially, $L_{\text{A,M,D,PM}}$ can be encoded in $L_{\text{S,M,D,PM}}$. The converse is impossible, as a corollary of Theorem 5.3 later on.

$L_{\text{S,P,D,NO}}$ is more expressive than $L_{\text{A,P,D,NO}}$

Trivially, $L_{\text{A,P,D,NO}}$ can be encoded in $L_{\text{S,P,D,NO}}$. The converse is impossible, as a corollary of Theorem 5.6 later on.

5.3 Completing the Hierarchy

We still need a few results to properly place all the asynchronous primitives in the hierarchy; such results are needed to properly merge Figures 2 and 3 and obtain the picture in Figure 1. Mainly, we prove that the hierarchy has a single bottom element (viz. $L_{\text{A,M,D,NO}}$) and that $L_{\text{A,M,D,PM}}$ cannot be compared with both $L_{\text{A,P,D,NO}}$ and $L_{\text{S,M,D,NO}}$.

$L_{\text{A,M,D,PM}}$ and $L_{\text{A,P,D,NO}}$ are more expressive than $L_{\text{A,M,D,NO}}$

This fact can be proved like in the synchronous case: in those proofs, synchrony of $L_{\text{S,M,D,PM}}$ and $L_{\text{S,P,D,NO}}$ does not play any role.

$L_{\text{A,M,D,PM}}$ and $L_{\text{A,P,D,NO}}$ are incomparable

The impossibility for a reasonable encoding both of $L_{\text{A,M,D,PM}}$ in $L_{\text{A,P,D,NO}}$ and of $L_{\text{A,P,D,NO}}$ in $L_{\text{A,M,D,PM}}$ can be proved similarly to Theorems 4.7 and 4.8, where synchrony played no role.

$L_{\text{A,M,D,PM}}$ and $L_{\text{S,M,D,NO}}$ are incomparable

The impossibility for a reasonable encoding of $L_{\text{A,M,D,PM}}$ in $L_{\text{S,M,D,NO}}$ is proved similarly to Theorem 4.9. The converse is proved via the following Theorem.

Theorem 5.3 *There exists no reasonable encoding of $L_{\text{S,M,D,NO}}$ in $L_{\text{A,M,D,PM}}$.*

Proof: The proof is somewhat similar to the proof of Theorem 4.5. Consider the processes $P \triangleq (x).P'$ and $Q \triangleq \langle a \rangle.Q'$, for $a \notin \text{Fn}(P')$. By Proposition 3.5, $[\![P]\!]$ and $[\![Q]\!]$ must communicate; this entails that $C_{|}[\__1;\__2]$, the context used to compositionally translate the parallel composition operator, must be structurally equivalent to $(\nu\widetilde{n})(\__1 \mid \__2 \mid \hat{P})$. Now, let $[\![P]\!] \mid \hat{P} \stackrel{\rho}{\Longrightarrow} R$ and $[\![Q]\!] \stackrel{\bar{\rho}}{\Longrightarrow} R'$, for $(\nu\widetilde{n},\widetilde{k})(R \mid R') \simeq [\![P'\{^a/_x\} \mid Q']\!]$ and $\widetilde{k} = \text{Bn}(\rho,\bar{\rho})$.

First of all, ρ contains label $?a'$, for at least one $a' \in \varphi_{[\![\,]\!]}(a)$, and this input must come from a formal template. Like in the proof of Theorem 4.5, we assume, for the sake of simplicity, that $|\varphi_{[\![\,]\!]}(\cdot)| = 1$ and let $\varphi_{[\![\,]\!]}(a) = a'$. Let us consider the first input in ρ, say $?m$ (at least one is present); so, let

$$\rho = \rho_1 . ?m \cdot \rho_2$$

with ρ_1 made up of output labels only. Let us consider $P' \triangleq \textbf{if } x \neq b \textbf{ then } \Omega$, for $b \neq a$ and $m \notin \varphi_{[\![\,]\!]}(b)$, and let us isolate two sub-cases.

1. $\rho_1 \neq \epsilon$. In this case, $?m$ cannot rely on a formal template (thus, $m \notin \varphi_{[\![\ ]\!]}(a)$) and $m \notin \mathrm{N}(\rho_1)$ otherwise, by Proposition 2.1(1,3), $[\![P]\!] \xrightarrow{\tau}$. Let $\overline{\rho} = \overline{\rho}_1 \cdot !m \cdot \overline{\rho}_2$, for $m \notin \mathrm{BN}(\overline{\rho}_1)$, and $\rho = \rho_3 \cdot ?a' \cdot \rho_4$, with $a' \notin \mathrm{N}(\rho_3)$; consequently, $\overline{\rho}$ can be also decomposed as $\overline{\rho}_3 \cdot !a' \cdot \overline{\rho}_4$. Let h be the number of $?a'$ occurring in ρ_4 and $\overline{\rho}_4$. Now, let $Q' \triangleq \mathbf{0}$ and consider the process obtained by putting $h + 2$ copies of $P \mid Q\{^b/_a\}$ in parallel; clearly, it does not diverge, whereas its encoding does. Indeed:

   - synchronize ρ_3 with $\overline{\rho}_3$ in the first copy of $[\![P \mid Q\{^b/_a\}]\!]$;
   - synchronize ρ_1 with $\overline{\rho}_1$ in the remaining copies of $[\![P \mid Q\{^b/_a\}]\!]$;
   - synchronize the $!m$ action from the second copy of $[\![Q\{^b/_a\}]\!]$ with the (formal) input action from the first copy of $[\![P]\!]$ (this has the effect of using m in place of a' within $[\![P']\!]$);
   - synchronize ρ_4 with $\overline{\rho}_4$ in the first copy of $[\![P \mid Q\{^b/_a\}]\!]$, except for the $?m$ actions not present in ρ_4 and $\overline{\rho}_4$ (these are at most h), that are instead synchronized with the $!m$ action from one of the remaining h copies of $[\![Q\{^b/_a\}]\!]$.

   This strategy generates a process with a parallel component barbed congruent to $[\![\mathbf{if}\ a \neq b\ \mathbf{then}\ \Omega]\!]\{^m/_{a'}\}$, that diverges.

2. $\rho_1 = \epsilon$. If $?m$ is actual, we work like in case 1 above, but with the process obtained by putting P in parallel with $h + 2$ copies of $Q\{^b/_a\}$. So, $?m$ relies on a formal input. Then, notice that $\overline{\rho}$ cannot contain output labels only, otherwise $[\![Q]\!] \Uparrow$ by letting $Q' \triangleq \Omega$ (indeed, if $[\![Q]\!] \stackrel{\overline{\rho}}{\Longrightarrow} \xrightarrow{\tau}^{\omega}$, then, by Proposition 2.1(3), $[\![Q]\!] \xrightarrow{\tau}^{\omega}$). Let $?n$ be the first input in $\overline{\rho}$; thus,

$$\rho = ?m \cdot \rho_5 \cdot !n \cdot \rho_6$$

   where $n \notin \mathrm{FN}(\overline{\rho}_5) \cup \{m\}$ and n cannot be restricted, otherwise the input in $\overline{\rho}$ would have been formal and, thus, $[\![Q]\!] \xrightarrow{\tau}$. We then work like in case 1 above, with $h + 2$ copies of $P \mid Q\{^b/_a\}$ in parallel; just notice that now we use n in place of m and all the needed $!n$ actions are taken from the $h + 1$ copies of $[\![P]\!]$. ∎

$L_{\mathrm{A,P,D,NO}}$ and $L_{\mathrm{S,M,D,NO}}$ are incomparable

Impossibility for a reasonable encoding of $L_{\mathrm{A,P,D,NO}}$ in $L_{\mathrm{S,M,D,NO}}$ can be proved like in Theorem 4.9, where synchrony does not play any role. The converse is also impossible, as proved in Theorem 5.6; to prove such a result, we need a few preliminary facts.

Lemma 5.4 *Let $[\![\cdot]\!]$ be a reasonable encoding with target $L_{\mathrm{A,P,D,NO}}$; then, $C_|[\_1; \_2]$, the context used to compositionally translate parallel composition, is barbed congruent to $(\nu\widetilde{n})(\_1 \mid \_2)$.*

Proof: By compositionality and Proposition 3.5, it holds that $C_|[\_1; \_2] \equiv (\nu\widetilde{n})(\_1 \mid \_2 \mid \hat{P})$, for some $\hat{P}$. It suffices to prove that $(\nu\widetilde{n})\hat{P} \simeq \mathbf{0}$; if it were not the case, then $[\![\mathbf{0} \mid \mathbf{0}]\!] \Downarrow$, in contradiction with reasonableness. ∎

Lemma 5.5 *Let $[\![\cdot]\!]$ be a reasonable encoding with target $L_{\mathrm{A,P,D,NO}}$; then, $P \xrightarrow{\tau}$ implies that $[\![P]\!] \xrightarrow{\tau}$.*

Proof: First of all, notice that in $L_{\text{A,P,D,NO}}$ it holds that $C_{\mathsf{op}}[\_] \not\Downarrow$, for every op, otherwise $[\![\, \mathsf{op}(\mathbf{0}) \,]\!] \Downarrow$. This implies that, whenever $C_{\mathsf{op}}[R] \xrightarrow{\tau}$, it must be that $R \xrightarrow{\tau}$: indeed, it cannot be that $C_{\mathsf{op}}[\_] \xrightarrow{\tau}$ nor that $C_{\mathsf{op}}[\_]$ and R communicate, otherwise in $L_{\text{A,P,D,NO}}$ this would imply that $C_{\mathsf{op}}[\_] \Downarrow$.

Now, let $P \xrightarrow{\tau}$; then, $*P \Uparrow$ and, hence, $[\![\, *P \,]\!] \Uparrow$. Thus, $[\![\, *P \,]\!] \xrightarrow{\tau}$ and this implies that $[\![\, P \,]\!] \xrightarrow{\tau}$, because $[\![\, *P \,]\!] \triangleq C_*[[\![\, P \,]\!]]$. ∎

Theorem 5.6 *There exists no reasonable encoding of $L_{\text{S,M,D,NO}}$ in $L_{\text{A,P,D,NO}}$.*

Proof: Consider $[\![\, \langle a \rangle.P \mid (x).Q \,]\!]$; because of Lemma 5.4 and operational correspondence, $[\![\, \langle a \rangle.P \mid (x).Q \,]\!] \simeq (\nu \widetilde{n})([\![\, \langle a \rangle.P \,]\!] \mid [\![\, (x).Q \,]\!]) \Rightarrow \simeq\ [\![\, P \mid Q\{a/x\} \,]\!] \simeq (\nu \widetilde{n})([\![\, P \,]\!] \mid [\![\, Q\{a/x\} \,]\!])$; this can only happen if $[\![\, \langle a \rangle.P \,]\!] \xRightarrow{\phi_1 \cdot \ldots \cdot \phi_t} \simeq\ [\![\, P \,]\!] \mid S_1$ and $[\![\, (x).Q \,]\!] \xRightarrow{\bar{\phi}_1 \cdot \ldots \cdot \bar{\phi}_t} \simeq\ [\![\, Q\{a/x\} \,]\!] \mid S_2$, for some S_1 and S_2 such that $(\nu\, \text{BN}(\phi_1, \ldots, \phi_t, \bar{\phi}_1, \ldots, \bar{\phi}_t))(S_1 \mid S_2) \simeq \mathbf{0}$; in $L_{\text{A,P,D,NO}}$ this entails that $S_1 \simeq \mathbf{0}$ and $S_2 \simeq \mathbf{0}$.

Notice that there must be at least one $i \in \{1, \ldots, t\}$ such that ϕ_i is an input action otherwise, by letting $P \triangleq (x).\mathbf{if}\ x = a\ \mathbf{then}\ \Omega$, we would have that $[\![\, \langle a \rangle.P \,]\!] \xRightarrow{\phi_1 \cdot \ldots \cdot \phi_t} \simeq\ [\![\, P \,]\!] \xRightarrow{\bar{\phi}_1 \cdot \ldots \cdot \bar{\phi}_t} \simeq\ [\![\, \Omega \,]\!]$ that, by Proposition 2.1(3), would imply $[\![\, \langle a \rangle.P \,]\!] \Uparrow$. Let ϕ_i the first input label and let $\phi_i = ?\widetilde{m}$. Thus, we have that

$$C_{\langle a \rangle}[[\![\, P \,]\!]] \xRightarrow{\phi_1 \cdot \ldots \cdot \phi_{i-1}} C_1[[\![\, P \,]\!]] \xrightarrow{?\widetilde{m}} C_2[[\![\, P \,]\!]]\{\widetilde{m}/\widetilde{y}\} \xRightarrow{\phi_{i+1} \cdot \ldots \cdot \phi_t} \simeq\ [\![\, P \,]\!]$$

where $C_1[\_] \equiv (\widetilde{y}).K_1 \mid K_2$ for some K_1 and K_2 such that exactly one of them is a context and the other one is a process. Symmetrically,

$$C_{(x)}[[\![\, Q \,]\!]] \xRightarrow{\bar{\phi}_1 \cdot \ldots \cdot \bar{\phi}_{i-1}} \mathcal{D}_1[[\![\, Q \,]\!]\sigma] \xrightarrow{(\nu \widetilde{h})!\widetilde{m}} \mathcal{D}_2[[\![\, Q \,]\!]\sigma] \xRightarrow{\bar{\phi}_{i+1} \cdot \ldots \cdot \bar{\phi}_t} \simeq\ [\![\, Q\{a/x\} \,]\!]$$

for some substitution σ. Notice that, because of asynchrony, $\mathcal{D}_1[\_] \equiv (\nu \widetilde{h})(\langle \widetilde{m} \rangle \mid \mathcal{D}_2[\_])$.

Now, consider $[\![\, \langle b \rangle.P \mid (x).Q \,]\!]$; by name invariance, we have that

$$C_{\langle b \rangle}[[\![\, P \,]\!]] \xRightarrow{\phi'_1 \cdot \ldots \cdot \phi'_{i-1}} C'_1[[\![\, P \,]\!]] \xrightarrow{?\widetilde{m}'} C'_2[[\![\, P \,]\!]]\{\widetilde{m}'/\widetilde{y}\} \xRightarrow{\phi'_{i+1} \cdot \ldots \cdot \phi'_t} \simeq\ [\![\, P \,]\!]$$

$$C_{(x)}[[\![\, Q \,]\!]] \xRightarrow{\bar{\phi}'_1 \cdot \ldots \cdot \bar{\phi}'_{i-1}} \mathcal{D}'_1[[\![\, Q \,]\!]\sigma'] \xrightarrow{(\nu \widetilde{h})!\widetilde{m}'} \mathcal{D}'_2[[\![\, Q \,]\!]\sigma'] \xRightarrow{\bar{\phi}'_{i+1} \cdot \ldots \cdot \bar{\phi}'_t} \simeq\ [\![\, Q\{b/x\} \,]\!]$$

where here and in what follows the 'primed' items denote the corresponding 'non-primed' items after the substitution of $\varphi_{[\![\ ]\!]}(b)$ for $\varphi_{[\![\ ]\!]}(a)$.

Let us now write $\phi_1 \cdot \ldots \cdot \phi_t$ as $\rho_0 \cdot ?\widetilde{n}_1 \cdot \ldots \cdot \rho_{k-1} \cdot ?\widetilde{n}_k \cdot \rho_k$, where ρ_i is either ε or it only contains output labels, for every i. If it were that $C_{\langle a \rangle}[[\![\, P \,]\!]] \xRightarrow{\rho_0 \cdot ?\widetilde{n}'_1 \cdot \ldots \cdot \rho_{k-1} \cdot ?\widetilde{n}'_k \cdot \rho_k} \simeq\ [\![\, P \,]\!]$, then

$$[\![\, \langle a \rangle.(x).\mathbf{if}\ x = a\ \mathbf{then}\ \Omega \mid \overbrace{(\langle b \rangle \mid (x)) \mid \ldots \mid (\langle b \rangle \mid (x))}^{k} \,]\!]$$

would diverge. Indeed, the i-th copy of $\langle b \rangle \mid (x)$ could be used to output $\widetilde{n}'_i$ that is consumed by $[\![\, \langle a \rangle.(x).\mathbf{if}\ x = a\ \mathbf{then}\ \Omega \,]\!]$; this unleashes $[\![\, (x).\mathbf{if}\ x = a\ \mathbf{then}\ \Omega \,]\!]$ without consuming any of the data produced by $C_{\langle a \rangle}[\_]$. Thus, $[\![\, (x).\mathbf{if}\ x = a\ \mathbf{then}\ \Omega \,]\!]$ can consume all such data and reduce to a process with a component barbed equivalent to $[\![\, \mathbf{if}\ a = a\ \mathbf{then}\ \Omega \,]\!]$. Thus, there must be a j such that the process obtained after $?\widetilde{n}'_j$ is not equivalent to the process obtained after $?\widetilde{n}_j$; for the sake of simplicity, let us assume that $j = 1$, i.e. the first input can change the behaviour of $C_{\langle a \rangle}[[\![\, P \,]\!]]$ (maybe, because of a name matching).

Let us now consider $\langle a\rangle \mid \langle b\rangle \mid (x)$ and the computation

$$[\![\, \langle a\rangle \mid \langle b\rangle \mid (x) \,]\!] \stackrel{\tau}{\Longrightarrow} [OUT_1 \mid \ldots \mid OUT_{i-1} \mid C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid C'_1 \mid \mathcal{D}'_2] \triangleq R$$

where $[\ldots]$ denotes a process with some top-level restricted names, OUT_j denotes the output process that generates ϕ_j and a context without argument denotes a context filled with $[\![\, \mathbf{0} \,]\!]$. By operational correspondence, it must be that $R \Rightarrow \simeq [\![\, P_r \,]\!]$, for $r \in \{1, 2\}$, $P_1 \triangleq \langle a\rangle$ and $P_2 \triangleq \langle b\rangle$; indeed, it cannot be that $R \Rightarrow \simeq [\![\, \langle a\rangle \mid \langle b\rangle \mid (x) \,]\!]$ otherwise the encoding would introduce divergence. We now prove that both the cases mentioned above contradict reasonableness.

1. If $R \Rightarrow \simeq [\![\, \langle a\rangle \,]\!]$, then C_1 must be restored and $C'_1 \mid \mathcal{D}'_2$ must be consumed. To this aim (recall that $C'_1[\_] \equiv (\widetilde{y}).K'_1 \mid K'_2$), we have that $C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \stackrel{\theta_1}{\Longrightarrow} H_1$, $K'_2 \stackrel{\bar{\theta}_1}{\Longrightarrow} K'_3$, $H_1 \mid K'_3 \stackrel{!\widetilde{v}}{\longrightarrow} H_2 \mid K'_4$, $H_2 \stackrel{\theta_2}{\Longrightarrow} \simeq C_1$ and $K'_1\{\widetilde{v}/\widetilde{y}\} \mid K'_4 \stackrel{\bar{\theta}_2}{\Longrightarrow} \simeq \mathbf{0}$. We can assume that $\theta_1 \cdot \theta_2$ contain at least one input label, otherwise $C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \mid K'_2 \Rightarrow \simeq C_1\{\widetilde{v}/\widetilde{y}\} \mid OUT(\theta_2)$, where $OUT(\theta_2)$ are the output processes that produce θ_2; then, $C_1\{\widetilde{v}/\widetilde{y}\} \mid OUT(\theta_2)$ must produce an output of length $|\widetilde{y}|$. This cannot be repeated indefinitely, otherwise the encoding would introduce divergence; thus, we must reach a process which produces an output of length $|\widetilde{y}|$ that cannot be consumed anymore by C_1; this can happen only if C_1 is activated after a blocking input action. Let us consider the case in which the first input is in θ_1 or in θ_2.

   - Let $\theta_1 = \theta_3 \cdot ?\widetilde{k} \cdot \theta_4$, for θ_3 with output labels only. Let us consider the computation $C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \Rightarrow H \mid OUT(\theta_3) \triangleq \hat{H} \stackrel{\tau}{\not\longrightarrow}$; at least one must exist, because of the blocking input $?\widetilde{k}$ and of divergence reflection. Let us now consider process $\langle a\rangle \mid (x) \mid \langle b\rangle \mid (x)$ and the following computation:

     $$[\![\, \langle a\rangle \mid (x) \mid \langle b\rangle \mid (x) \,]\!] \Rightarrow [C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \mid C'_2\{\widetilde{n_1}/\widetilde{y}\} \mid \mathcal{D}_2] \Rightarrow [\hat{H} \mid \hat{H}']$$

     Now, $\hat{H} \mid \hat{H}'$ cannot reduce: by construction, $\hat{H} \stackrel{\tau}{\not\longrightarrow}$ and $\hat{H}' \stackrel{\tau}{\not\longrightarrow}$; moreover, $\hat{H}$ cannot communicate with $\hat{H}'$, otherwise both $\hat{H}$ and $\hat{H}'$ could reduce. By operational correspondence, $[\hat{H} \mid \hat{H}']$ should be barbed congruent to the encoding of some reduct of $\langle a\rangle \mid (x) \mid \langle b\rangle \mid (x)$, but this is not possible: because of Lemma 5.5, it cannot be equivalent to $[\![\, \langle a\rangle \mid (x) \mid \langle b\rangle \mid (x) \,]\!]$, $[\![\, \langle a\rangle \mid (x) \,]\!]$ and $[\![\, \langle b\rangle \mid (x) \,]\!]$; by faithfulness, it cannot be equivalent to $[\![\, \mathbf{0} \,]\!]$ because $[\hat{H} \mid \hat{H}'] \Downarrow$ (it at least performs two input actions, viz. $?\widetilde{k}$ and $?\widetilde{k'}$).
   - Let $\theta_2 = \theta_3 \cdot ?\widetilde{k} \cdot \theta_4$, for $\theta_1 \cdot \theta_3$ with output labels only. This case is similar to the previous one, with $\hat{H} \triangleq H \mid OUT(\theta_1 \cdot \theta_3) \mid \langle \widetilde{v}\rangle$, if $H_1 \stackrel{!\widetilde{v}}{\longrightarrow}$, and $\hat{H} \triangleq H \mid OUT(\theta_1 \cdot \theta_3)$, otherwise.

2. If $R \Rightarrow \simeq [\![\, \langle b\rangle \,]\!]$, notice that $\widetilde{n'_1}$ only depends on $OUT'_1, \ldots, OUT'_{i-1}$; hence, the fact that $C_2\{\widetilde{n'_1}/\widetilde{y}\}$ behaves differently from $C_2\{\widetilde{n_1}/\widetilde{y}\}$ implies that there exists at least one $j \in \{1, \ldots, i-1\}$ such that $OUT_j \neq OUT'_j$. Thus, $R \Rightarrow \simeq [\![\, \langle b\rangle \,]\!]$ implies that there must be some OUT_j that is turned into OUT'_j, i.e. $C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \stackrel{\bar{\phi}_j}{\Longrightarrow}$, where $\bar{\phi}_j$ corresponds to the first OUT_j that is turned into OUT'_j. Let us now consider $C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \Rightarrow \hat{H} \stackrel{\bar{\phi}_j}{\longrightarrow}$, for $\hat{H} \stackrel{\tau}{\not\longrightarrow}$ and the computation

   $$[\![\, \langle a\rangle \mid (x) \mid \langle b\rangle \mid (x) \,]\!] \Rightarrow [C_2\{\widetilde{n'_1}/\widetilde{y}\} \mid \mathcal{D}'_2 \mid C'_2\{\widetilde{n_1}/\widetilde{y}\} \mid \mathcal{D}_2] \Rightarrow [\hat{H} \mid \hat{H}']$$

   Like in case 1 above, this suffices to violate operational correspondence. ∎

6 Conclusions and Related Work

We have studied the expressive power of sixteen communication primitives, arising from the combination of four features: synchronism, arity of data, communication medium and presence of pattern-matching. By relying on possibility/impossibility of 'reasonable' encodings, we obtained a clear hierarchy of communication primitives. Notably, LINDA's communication paradigm [22] is at the top of this hierarchy, whereas the π-calculus is in the middle. A posteriori, this can justify the fact that the former one is usually exploited in actual programming languages [3, 20], where flexibility and expressive power are the driving issues, whereas the latter one is mostly used for theoretical reasoning. Of course, the step that comes after this theoretical approach is the study of more concrete languages, maybe by encoding them in one of the languages presented in this paper.

Related work. One of the pioneering works in the study of communication primitives for distributed systems is [25]. There, the expressive power of several "classical" primitives (like `test-and-set`, `compare-and-swap`, ...) is studied by associating to every primitive the highest number of parallel processes that can reach a distributed consensus with that primitive, under conditions quite similar to our Definition 3.2. It then follows that a primitive with number n is less expressive than every primitive with number m ($> n$): the latter one can solve a problem (i.e. the consensus among m processes) that the former one cannot reasonably solve. This idea is also exploited in [35] to assess the expressive power of the non-deterministic choice in the π-calculus and in [39] to evaluate the expressiveness of Mobile Ambients.

In [16], the notion of relative expressive power is used to compare different programming languages. In particular, a simple class of three concurrent constraint languages is studied and organized in a strict hierarchy. The languages have guarded constructs and only differ in the features offered by the guards: a guard is always passed in the least expressive language; a guard is passed only if a given constraint is satisfied by the current knowledge; finally, a guard is passed only if a new constraint, that must be atomically added to the knowledge, is consistent with the current knowledge. Roughly, the last kind of guards can be related to the pattern-matching construct of our languages, for the possibility of atomically testing and modifying the environment; in both cases, this feature sensibly increases the expressiveness of the language.

By the way, the form of pattern-matching considered here is very minimal: only the equality of names can be tested while retrieving a datum. However, other forms of pattern-matching can be exploited (e.g., those described in [18]), to have more and more flexible formalisms; some proposals have been investigated from the expressiveness point of view in [49].

Another form of atomic polyadic name matching is presented in [13], but with a different approach with respect to ours. In our $L_{\text{-},\text{P},\text{-},\text{PM}}$, the tuple of names to be matched is in the transmitted/received value (by using a standard π-calculus terminology, the tuple is in the 'object' part of an output/input); on the contrary, [13] exploit composite channel names that must all be matched to enable a communication (thus, the tuple is in the 'subject' part of the output/input). This feature enables a nice modeling of distributed and cryptographic process calculi; nevertheless, our LINDA-like pattern-matching is stronger, since the possibility of using formal and actual templates together provides a more flexible form of input actions (that can easily encode the ones in [13]).

Finally, in [8] three different semantics for asynchronous languages are studied in the setting of a simple LINDA-based process calculus: *instantaneous output* (an output prefix immediately unleashes the corresponding tuple in the dataspace), *ordered output* (a reduction is needed to turn an

output prefix into the corresponding tuple in the dataspace) and *unordered output* (two reductions are needed to turn an output into an available tuple, i.e. one to send the tuple to the dataspace and another one to make the tuple available in the dataspace). In [8, 9] it is proved that the semantics can be strictly ordered according to their expressive power, with the instantaneous semantics being the most expressive one and the unordered semantics being the least expressive one (actually, the latter semantics entails a language which is not Turing complete). According to this terminology, the semantics we used in this paper for the asynchronous languages is instantaneous; it would be interesting to discover whether our results still hold also under different semantics or not.

Acknowledgments I would like to thank Rosario Pugliese, Daniele Varacca and Nobuko Yoshida for their interest in my work and for several suggestions that improved a first draft of this paper. I am also grateful to Catuscia Palamidessi for her encouragements and discussions. Finally, I would like to thank the anonymous reviewers for their positive attitude and for their detailed comments that improved both the presentation and the contents of this work.

References

[1] L. Acciai and M. Boreale. Xpi: A typed process calculus for XML messaging. In *Proc. of FMOODS'05*, volume 3535 of *LNCS*, pages 47–66. Springer, 2005.

[2] R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.

[3] K. Arnold, E. Freeman, and S. Hupfer. *JavaSpaces Principles, Patterns and Practice*. Addison-Wesley, 1999.

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

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

[6] 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*, pages 18–34. Springer, 2005.

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

[8] N. Busi, R. Gorrieri, and G. Zavattaro. Comparing three semantics for LINDA-like languages. *Theoretical Computer Science*, 240(1):49–90, 2000.

[9] N. Busi, R. Gorrieri, and G. Zavattaro. On the expressiveness of LINDA coordination primitives. *Information and Computation*, 156(1-2):90–121, 2000.

[10] N. Busi and G. Zavattaro. On the expressive power of movement and restriction in pure mobile ambients. *Theoretical Computer Science*, 322(3):477-515, 2004.

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

[12] D. Cacciagrano, F. Corradini, and C. Palamidessi. Separation of synchronous and asynchronous communication via testing. In *Proc. of EXPRESS'05*, ENTCS, 154(3): 95–108. Elsevier, 2006.

[13] M. Carbone and S. Maffeis. On the expressive power of polyadic synchronisation in π-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.

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

[15] 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.

[16] F. de Boer and C. Palamidessi. Embedding as a tool for language comparison. *Information and Computation*, 108(1):128–157, 1994.

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

[18] R. De Nicola, D. Gorla, and R. Pugliese. Pattern matching over a dynamic network of tuple spaces. In *Proc. of FMOODS'05*, volume 3535 of *LNCS*, pages 1–14. Springer, 2005.

[19] C. Ene and T. Muntean. Expressiveness of point-to-point versus broadcast communications. In *Proc. of 12th Symp. on Fundamentals of Computation Theory*, volume 1684 of *LNCS*, pages 258–268. Springer, 1999.

[20] D. Ford, T. Lehman, S. McLaughry, and P. Wyckoff. T Spaces. *IBM Systems Journal*, pages 454–474, August 1998.

[21] C. Fournet, J.-J. Levy and A. Schmitt. An Asynchronous Distributed Implementation of Mobile Ambients. In *Proc. of IFIP-TCS*, volume 1872 of *LNCS*, pages 348–364. Springer, 2000.

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

[23] 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.

[24] D. Gorla. Synchrony vs Asynchrony in Communication Primitives. In *Proc. of EXPRESS'06*, ENTCS 175(3):87–108, Elsevier 2007.

[25] M. Herlihy. Wait-Free Synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124–149, 1991.

[26] 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.

[27] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

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

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

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

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

[32] U. Nestmann. What is a 'good' encoding of guarded choice? *Information and Computation*, 156:287–319, 2000.

[33] U. Nestmann. Welcome to the Jungle: A Subjective Guide to Mobile Process Calculi (Invited Tutorial). In *Proc. of CONCUR*, volume 4137 of *LNCS*, pages 52–63. Springer, 2006.

[34] U. Nestmann and B. C. Pierce. Decoding choice encodings. *Information and Computation*, 163:1–59, 2000.

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

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

[37] J. Parrow. Expressiveness of Process Algebras. In *LIX Colloquium on Emerging Trends in Concurrency Theory*, ENTCS (to appear).

[38] A. Phillips, N. Yoshida and S. Eisenbach. A Distributed Abstract Machine for Boxed Ambient Calculi. In *Proc. of ESOP*, volume 2986 of *LNCS*, pages 155–170. Springer, 2004.

[39] I.C.C. Phillips, and M.G. Vigliotti. Electoral systems in ambient calculi. *Proc. FoSSaCS*, volume 2987 of *LNCS*, pages 408–422, 2004.

[40] 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.

[41] 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.

[42] P. Quaglia and D. Walker. Types and full abstraction for polyadic π-calculus. *Information and Computation*, 200(2):215–246, 2005.

[43] D. Sangiorgi. Bisimulation in higher-order process calculi. *Information and Computation*, 131:141–178, 1996.

[44] D. Sangiorgi and A. Valente. A distributed abstract machine for Safe Ambients. In *Proc. of ICALP*, volume 2076 of *LNCS*, pages 408–420. Springer, 2001.

[45] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[46] E. Y. Shapiro. Separating concurrent languages with categories of language embeddings. In *Proc. of 23rd Symposium on Theory of Computing*, pages 198–208. ACM Press, 1991.

[47] V. Vasconcelos and K. Honda. Principal typing schemes in a polyadic π-calculus. In *Proc. of CONCUR'93*, volume 715 of *LNCS*, pages 524–538. Springer, 1993.

[48] N. Yoshida. Graph types for monadic mobile processes. In *Proc. of FSTTCS '96*, volume 1180 of *LNCS*, pages 371–386. Springer, 1996.

[49] G. Zavattaro. Towards a hierarchy of negative test operators for generative communication. In *Proc. of EXPRESS*, ENTCS, 16(2):154–170. Elsevier, 1998.