Towards a Unified Approach to Encodability and Separation Results for Process Calculi

Daniele Gorla[a]

[a]*Dipartimento di Informatica, "Sapienza" Università di Roma*

Abstract

We present a unified approach to evaluating the relative expressive power of process calculi. In particular, we identify a small set of criteria (that have already been somehow presented in the literature) that an encoding should satisfy to be considered a valid means for language comparison. We argue that the combination of such criteria is a valid proposal by noting that: (*i*) several well-known encodings appeared in the literature satisfy them; (*ii*) this notion is not trivial, because some known encodings do not satisfy all the criteria we have proposed; (*iii*) several well-known separation results can be formulated in terms of our criteria; and (*iv*) some widely believed (but never formally proved) separation results can be proved by using the criteria we propose. Moreover, the criteria defined induce general proof techniques for separation results that can be easily instantiated to cover known case-studies.

Keywords: Process calculi, Expressiveness, Encodings

1. Introduction

As argued in [43], one of the hottest topics in concurrency theory, and mainly in process calculi, is the identification of a uniform way to formally compare different languages from the expressiveness point of view. Indeed, while the literature contains several results and claims concerning the expressive power of a language, such results are usually difficult to appreciate because they are proved sound by using different criteria. For a very good overview of the problem, we refer the reader to [49].

In the 1980s, the trend was to adopt the approach followed in computability theory and study the *absolute* expressive power of languages, e.g. by studying which problems were solvable or which operators were definable in a given language. In the 1990s, the focus moved to the *relative* expressive power: it became more interesting to understand the extent to which a language could be encoded into another one, also because of the proliferation of different process calculi. Nevertheless, some instructive absolute expressiveness results have also appeared in the last few years: for example, [8, 9, 47] have recently compared the power of recursion and replication in process calculi, by proving that in some case the languages with replication are not Turing powerful. However, working with absolute expressiveness only yields a bipartition of languages: the ones able to solve a given problem (for example, the possibility of simulating Turing machines) and the one unable to do so. For this reason, we think that relative expressiveness is more adequate when one wants to build up lattices of languages.

[☆]This work is an extended and revised version of [27] and it includes some material taken from [24].

Email address: gorla@di.uniroma1.it (Daniele Gorla)

A very common approach to proving soundness of encodings is based on the notion of *full abstraction*. This concept was introduced in the 1970s to require an exact correspondence between a denotational semantics of a program and its operational semantics. Intuitively, a denotational semantics is fully abstract if it holds that two observably equivalent programs (i.e., two programs that 'behave in the same way' in any execution context) have the same denotation, and vice versa. The notion of full abstraction has been adapted to prove soundness of encodings by requiring that an encoding maps equivalent source terms into equivalent target terms, and vice versa. This adaptation was justified by the fact that an encoding resembles a denotation function: they both map elements of a formalism (viz., terms of the source language) into elements of a different formalism (another language, in the case of an encoding, or a mathematical object, in the case of a denotation function). In this way, the stress is put on the requirement that the encoding must translate a language into another one while respecting some associated equivalences. This can be very attractive, e.g., if in the target we can exploit automatic tools to prove equivalences and then pull back the obtained result to the source. However, we believe that full abstraction is too focused on the equivalences and thus it gives very little information on the computation capabilities of the two languages.

Operational and structural criteria have been developed in the years to state and prove separation results [12, 28, 45, 50, 51], that are a crucial aspect of building a hierarchy of languages. Indeed, to prove that a language $\mathcal{L}_1$ is more expressive than another language $\mathcal{L}_2$, we need to show that there exists a "valid" encoding of the latter into the former, but not vice versa. Usually, the latter fact is very difficult to prove and is obtained by: (1) identifying a problem that can be solved in $\mathcal{L}_1$ but not in $\mathcal{L}_2$, and (2) finding the least set of criteria that an encoding should meet to translate a solution in $\mathcal{L}_1$ into a solution in $\mathcal{L}_2$. Such criteria are problem-driven, in that different problems call for different criteria (compare, for example, the criteria in [45, 50, 51] with those in [12, 28]). Moreover, the criteria used to prove separation results are usually not enough to testify to the quality of an encoding: they are considered minimal requirements that any encoding should satisfy to be considered a valid means for language comparison.

In this paper, we present a new proposal for assessing the quality of an encoding, tailored to aspects that are strictly related to relative expressiveness. We isolate a small set of requirements that, in our opinion, are very well-suited to proving both soundness of encodings and separation results. In this way, we obtain a notion of encodability that can be used to place two (or more) languages in a clearly organized hierarchy. A preliminary proposal appeared in [23] but it was formulated in a too demanding way.

Of course, in order to support our proposal, we have to give evidence of its reasonableness. To this aim, we exhibit both philosophical and pragmatic arguments. From the pragmatic side, we notice that most of the encodings appearing in the literature satisfy our criteria and that their combination is not trivial, because there exist some encodings (e.g., the encodings of π-calculus in Mobile Ambients proposed in [14, 16]) that do not satisfy all the criteria we propose. Moreover, we also prove that several known separation results can be straightforwardly obtained as instances of the framework we present; furthermost, some new separation results can be now formally proved by using the criteria we propose. The philosophical part is, by contrast, more delicate because we have to convince the reader that every proposed criterion is deeply related to relative expressiveness. To this aim, we split the criteria in two groups: structural and semantic. We think that structural criteria are difficult to criticize: we simply require that the encoding is compositional and that it does not depend on the specific names appearing in the source term. Semantic criteria are, as usual, more debatable, because different people have different views on the semantics of a calculus and because the same semantic notions can be defined in different ways. Here, we assume that an encoding should be: *operationally corresponding*, in the sense that it preserves and reflects the computations of the source terms; *divergence reflecting*, in that we do not want to turn a terminating term

into a non-terminating one; and *success sensitive*, i.e., once defined a notion of successful computation of a term, we require that successful source terms are mapped into successful target terms and vice versa.

Although intuitively quite clear, the above mentioned criteria can be formulated in different ways. In particular, operational correspondence is usually defined up to some semantic equivalence/preorder to ignore dead processes yielded by the encoding. However, there is a wide range of equivalences/preorders and choosing one or another is always highly debatable. In Section 2 we start by leaving the notion of equivalence/preorder unspecified; this is, in our opinion, the ideal scenario, where encodability and separation results do not depend on the particular semantic theory chosen. However, when we want to prove some concrete result, we are forced to make assumptions on the equivalence used in operational correspondence. In doing this, we try to work at the highest possible abstraction level; in particular, we never commit to any specific equivalence/preorder and always consider meaningful families of such relations.

The paper is organized as follows. In Section 2, we present the criteria that we are going to consider and compare them with other ones already presented in the literature. To support our criteria, we then put them at work on some mainstream process calculi: CCS [37], the asynchronous π-calculus (π_a) [6], the separate and mixed choice π-calculus (π_{sep} and π_{mix}) [57], Mobile Ambients (MA) [16] and the π-calculus with polyadic synchronizations (e^{π} and π^n) [12]. A sketch of their syntax and operational semantics is in Section 3. In Section 4 we start revisiting some very well-known encodings and show that they all meet our criteria: this shows that our criteria accord with the community's common sense. Then, we show that, nevertheless, our criteria are not trivial, since all the encodings of π_a into MA do not satisfy at least one of them; this is not due to the impossibility of developing such an encoding and, indeed, we do provide a valid encoding. In Section 5 we move to separation results. First, we give some general proof techniques that can be easily instantiated to prove (in a simpler and more uniform way) known separation results appearing in the literature; to this aim, we specialize in three ways the semantic theory used to define operational correspondence. Second, we show a couple of separation results (of MA into π_{mix} and of CCS into MA) that, to the best of our knowledge, have never been proved yet. In Section 6 we discuss if and how our approach can be scaled for dealing with more sophisticated kinds of encodings, like *two-level* [4, 7] or *parameterized* [34, 38, 58] ones. In Section 7 we conclude by summing up our main contributions and discussing future work.

2. The Encodability Criteria

Process calculi. In this section we discuss the criteria an encoding should satisfy to be considered a valid means for language comparison. For the moment, we work at an abstract level and do not commit to any precise formalism. Indeed, we just assume a (countable) set of names $\mathcal{N}$ and specify a calculus as a triple $\mathcal{L} = (\mathcal{P}, \longmapsto, \asymp)$, where

- $\mathcal{P}$ is the set of language terms (usually called *processes*) that is built up from the terminated process $\mathbf{0}$ and the success process $\surd$ (whose need will be clear when presenting Property 5 later on) by at least using the parallel composition operator '|', that we assume to be unique in every language.[1] Processes are usually identified up to some

[1] This assumption is very realistic; indeed, modern process calculi have a single parallel composition operator. If this was not the case (for example, assume to have a language with two parallel operators, one allowing synchronization and one not), some technicalities of this paper would be difficult to express in a convincing way. For example, the homomorphism criterion of Definition 5.2 in Section 5.1.1 could be formulated in different ways: which parallel has to be translated homomorphically, if the source language has two parallel operators? Or, conversely, which parallel operator of the target has to be used when translating the source parallel operator(s)?

notion of *structural congruence*, written $\equiv$, that intuitively equates different syntactic ways of writing the same process.

- $\longmapsto$ is the operational semantics, needed to specify how a process computes; following common trends in process calculi, we specify the operational semantics by means of *reductions*. Usually, $\longmapsto$ is a binary relation on processes inductively defined by rules in the structural operational semantics style [52]. As usual, $\Longmapsto$ denotes the reflexive and transitive closure of $\longmapsto$.

- $\asymp$ is a behavioural equivalence needed to describe the abstract behaviour of a process. Usually, $\asymp$ is a congruence at least with respect to parallel composition; it is often defined in the form of a barbed equivalence [41] or can be derived directly from the reduction semantics [32].

Encodings. An *encoding* of $\mathcal{L}_1 = (\mathcal{P}_1, \longmapsto_1, \asymp_1)$ into $\mathcal{L}_2 = (\mathcal{P}_2, \longmapsto_2, \asymp_2)$ is a pair $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket})$ where $\llbracket \cdot \rrbracket : \mathcal{P}_1 \to \mathcal{P}_2$ is called *translation* and $\varphi_{\llbracket\rrbracket} : \mathcal{N} \to \mathcal{N}^k$ is called *renaming policy* and it is such that $\varphi_{\llbracket\rrbracket}(u) \cap \varphi_{\llbracket\rrbracket}(v) = \emptyset$, for all $u \neq v$, where $\varphi_{\llbracket\rrbracket}(\cdot)$ is simply considered a set here. The translation turns every source term into a target term; in doing this, it is possible that the translation fixes some names to play a precise rôle or it can translate a single name into a tuple of names. In most of the encodings present in the literature, every name is simply translated to itself. However, it is sometimes necessary to have a set of *reserved* names, i.e. names with a special function within the encoding. Reserved names can be obtained either by assuming that the target language has more names than the source one, or by exploiting what we call a *strict renaming policy*, i.e. a renaming policy $\varphi_{\llbracket\rrbracket} : \mathcal{N} \longrightarrow \mathcal{N}$. For example, we can isolate one reserved name by linearly ordering the set of names $\mathcal{N}$ as $\{n_0, n_1, n_2, \ldots\}$ and by letting $\varphi_{\llbracket\rrbracket}(n_i) \triangleq n_{i+1}$, for every i; the reserved name is n_0.

The requirement that $\varphi_{\llbracket\rrbracket}$ maps names to tuples of the same length can be justified by the fact that names are all 'at the same level' and, thus, they must be treated uniformly. Moreover, such tuples must be finite, otherwise it would be impossible to transmit all $\varphi_{\llbracket\rrbracket}(a)$ in the translation of a communication where name a is exchanged (notice that, since the sender cannot know how the receiver will use a, all $\varphi_{\llbracket\rrbracket}(a)$ must be somehow transmitted). Consequently, the requirement that different names are associated to disjoint tuples can be intuitively justified as follows. Assume that there exists $u \neq v$ such that $\varphi_{\llbracket\rrbracket}(u) \cap \varphi_{\llbracket\rrbracket}(v) \neq \emptyset$; since there is no relationship between different names, this implies that, for every w, $\varphi_{\llbracket\rrbracket}(u) \cap \varphi_{\llbracket\rrbracket}(w) \neq \emptyset$. If the name shared by every pair of tuples is the same, then such a name can be considered reserved and we can define a renaming policy $\varphi'_{\llbracket\rrbracket}$ satisfying the requirement that different names are associated with disjoint tuples. Otherwise, for every v and w, $\varphi_{\llbracket\rrbracket}(v)$ and $\varphi_{\llbracket\rrbracket}(w)$ must have a different name in common with $\varphi_{\llbracket\rrbracket}(u)$; thus, $\varphi_{\llbracket\rrbracket}(u)$ would contain an infinite number of names.

To simplify reading, we shall usually write $\llbracket \cdot \rrbracket$ instead of $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\rrbracket})$, by leaving the renaming policy understood. Moreover, we let S range over processes of the source language (viz., $\mathcal{L}_1$) and T range over processes of the target language (viz., $\mathcal{L}_2$).

Valid encodings. We shall call *valid* any encoding that satisfies the criteria we are going to present. Notice that, since we aim at a set of criteria suitable for both encodability and separation results, we have to find a compromise between 'minimality' (typical of separation results, where one wants to identify the minimal set of properties that make a separation result provable) and 'maximality' (typical of encodability results, where one wants to show that the encoding satisfies as many properties as possible).

First of all, a translation should be compositional, i.e. the translation of a compound term must be defined in terms of the translation of the subterms, where, in general, the translated subterms can be combined by relying on a context that coordinates their inter-relationships.

A *k-ary context* $C(\__1; \ldots; \__k)$ is a term where k occurrences of $\mathbf{0}$ are linearly replaced by the holes $\{\__1; \ldots; \__k\}$ (every hole must occur once and only once). In defining compositionality, we let the context used to combine the translated subterms depend on the operator that combines the subterms and on the free names (written $\text{FN}(\cdot)$) of the subterms. For example, we could think to have a name handler for every free name in the subterms.

> **Property 1 (Compositionality).** *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* compositional *if, for every k-ary operator* $\mathtt{op}$ *of* $\mathcal{L}_1$ *and for every subset of names N, there exists a k-ary context* $C^N_{\mathtt{op}}(\__1; \ldots; \__k)$ *such that, for all* $S_1, \ldots, S_k$ *with* $\text{FN}(S_1, \ldots, S_k) = N$*, it holds that* $[\![\, \mathtt{op}(S_1, \ldots, S_k) \,]\!] = C^N_{\mathtt{op}}([\![\, S_1 \,]\!]; \ldots; [\![\, S_k \,]\!])$.

Compositionality is a very natural property and, indeed, every encoding we are aware of is defined compositionally. Compositionality with respect to some specific operator has been assumed also to prove some separation result, viz. of synchronous vs asynchronous π-calculus [11] or of persistent fragments of the asynchronous π-calculus [10]. However, for separation results, the most widely accepted criterion is homomorphism of parallel composition [12, 28, 45, 46, 50, 51]; indeed, translating a parallel process by introducing a coordinating context would reduce the degree of distribution and show that $\mathcal{L}_2$ has not enough expressive power to simulate $\mathcal{L}_1$. This point of view has been, however, sometimes criticized and, indeed, there exist encodings that do not translate parallel composition homomorphically [4, 7, 42].

Our definition of compositionality allows two processes that only differ in their free names to have totally different translations: indeed, it could be that $C^N_{\mathtt{op}}(\ldots)$ is very different from $C^M_{\mathtt{op}}(\ldots)$, whenever $N \neq M$. We want to avoid this fact; indeed, a valid translation cannot depend on the particular names involved in the source process, but only on its syntactic structure. In our view, a translation should reflect in the translated term all the renamings carried out in the source term. In what follows, we denote with σ a substitution of names for names, i.e. a function $\sigma : \mathcal{N} \longrightarrow \mathcal{N}$, and we shall usually specify only the non-trivial part of a substitution: for example, $\{^b/_a\}$ denotes the (non-injective) substitution that maps a to b and every other name to itself. Moreover, we shall also extend substitutions to tuples of names in the expected way, i.e. component-wise.

> **Property 2 (Name invariance).** *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* name invariant *if, for every* S *and* σ*, it holds that*
>
> $$[\![\, S\sigma \,]\!] \begin{cases} = [\![\, S \,]\!]\sigma' & \text{if } \sigma \text{ is injective} \\ \asymp_2 [\![\, S \,]\!]\sigma' & \text{otherwise} \end{cases}$$
>
> *where* σ' *is such that* $\varphi_{[\![\,]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![\,]\!]}(a))$ *for every* $a \in \mathcal{N}$.

To understand the distinction between injective and non-injective substitutions, assume that σ maps two (or more) different names to the same name. Then, the set of free names of $S\sigma$ is smaller than the set of free names in S; by compositionality, this fact leads to different translations, in general. For example, if the translation introduces a name handler for every free name, having sets of free names with different cardinality leads to inherently different translations. However, non-injective substitutions are natural in name-passing calculi, where language contexts can induce them. In this case, the formulation with '$=$' is too demanding and the weaker formulation (with '$\asymp_2$') is needed. Thus, this formulation implies that two name handlers for the same name are behaviourally equivalent to one handler for that name; this seems us a very reasonable requirement. Notice that our definition of name invariance is definitely more complex than those, e.g., of [12, 45, 50, 51], where it is required that $[\![\, S\sigma \,]\!] = [\![\, S \,]\!]\theta$ for some (not better specified) substitution θ. However, we do not think that our formulation is more demanding; it is just more detailed and we consider this fact a further

contribution of our paper. Finally, notice that we are not aware of any encoding that satisfies Property 2 in the weaker formulation, i.e. with '$\asymp_2$' in place of '='. Nevertheless, since we are defining a general theory, we do not see anything wrong with the weaker formulation and still consider it as part of the definition of valid encodings.

Up to now, we have presented and discussed properties dealing with the way in which a translation is defined; we are still left with the more crucial part of the criteria. We want to focus our attention on the possible computations (i.e., sequences of reductions) of a process; thus, we require that the source and the target language have the same computations. A widely accepted way to formalize this idea is via operational correspondence that, intuitively, ensures two crucial aspects: (*i*) every computation of a source term can be mimicked by its translation (thus, the translation does not reduce the behaviours of the source term); and (*ii*) every computation of a translated term corresponds to some computation of its source term (thus, the translation does not introduce new behaviours).

Property 3 (Operational correspondence). *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* operationally corresponding *if it is*

Complete: *for all* $S \Longmapsto_1 S'$*, it holds that* $[\![S]\!] \Longmapsto_2 \asymp_2 [\![S']\!]$*;*
Sound: *for all* $[\![S]\!] \Longmapsto_2 T$*, there exists an* S' *such that* $S \Longmapsto_1 S'$ *and* $T \Longmapsto_2 \asymp_2 [\![S']\!]$.

Notice that operational correspondence is very often used for assessing the quality of an encoding; thus, we took it into account for having a set of criteria that work well both for encodability and for separation results. Nothing related to this property has ever been assumed for separation results, except in [23, 25] where, however, it was formulated in a too demanding way. Also notice that the original formulation of operational correspondence put forward in [44] does not use '$\asymp_2$'; for this reason, it is too demanding and, indeed, several encodings (including those in *loc. cit.*) do not enjoy it. The problem is that usually encodings leave some 'junk' process after having mimicked some source language reduction; such a process invalidates the 'exact' formulation of this property. The use of '$\asymp_2$' is justified to get rid of potential irrelevant junk processes.

Another important semantic issue, borrowed from [12, 17, 30, 42], is that a translation should not introduce infinite computations, written $\longmapsto^{\omega}$.

Property 4 (Divergence reflection). *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ reflects divergence *if, for every* S *such that* $[\![S]\!] \longmapsto_2^{\omega}$*, it holds that* $S \longmapsto_1^{\omega}$.

One may argue that divergence can be ignored if it arises with negligible probability or in unfair computations. However, suppose that every translation of $\mathcal{L}_1$ into $\mathcal{L}_2$ introduces some kind of divergence; this means that, to preserve all the functionalities of a terminating source term, every translation has to add infinite computations in the translation of the term. This fact makes $\mathcal{L}_2$ not powerful enough to encode $\mathcal{L}_1$ and is fundamental to proving several separation results (e.g., that the `test-and-set` primitive cannot be encoded via any combination of `read` and `write` – see [30]).

It is interesting to notice that, with all the properties listed up to now, one can accept the translation that maps every source term into $\mathbf{0}$. Of course, this translation is "wrong" because it does not distinguish processes with different interaction capabilities. In process calculi, interaction capabilities are usually described either by the *barbs* that a process exhibits [41] or by the set of *tests* that a process successfully passes [18, 55]. Barbs are often defined in a very ad hoc way, they are chosen as the simplest predicates that induce meaningful congruences and they strictly depend on their language (even though in [55] there is a preliminary attempt at a 'canonical' definition of barbs); for this reason, we found it difficult to work out a satisfactory semantic property relying on barbs for encodings that translate a source language into

a very different target language (notice that barb correspondence is by contrast very natural in, e.g., [12, 28, 45] where similar languages are studied). On the contrary, the testing approach is more uniform: it identifies a binary predicate $P \Downarrow O$ of *successful computation* for a process P in a parallel context O (usually called *observer*, that is a normal process containing occurrences of a distinguished success action), and, by varying O, it describes the interactions P can be engaged in. Moreover, the testing approach is at the same time more general and more elementary than barbs: the latter ones can be identified via elementary tests, and test passing is the basic mechanism for the 'canonical' definition of barbs in [55].

By following [3, 10, 11], we shall require that the source and the translated term behave in the same way with respect to success. However, a formulation like "$\forall P \forall O.P \Downarrow O$ iff $[\![P]\!] \Downarrow [\![O]\!]$" is not adequate in our setting: indeed, it is possible to have a successful computation for $P|O$ but not for $[\![P]\!] \mid [\![O]\!]$ since, because of compositionality, a successful computation in the target would be possible only with the aid of the coordinating context used to compositionally translate the parallel composition. Thus, we have to define $\Downarrow$ as a unary predicate and require that "$\forall P \forall O.P|O \Downarrow$ iff $[\![P|O]\!] \Downarrow$". For our aims, it is not necessary to distinguish between processes and observers. Moreover, to formulate our property in a simpler way, we assume that all the languages contain the same success process $\surd$ and that $\Downarrow$ means reducibility (in some modality, e.g. may/must/...) to a process containing a top-level unguarded occurrence of $\surd$. This is similar to [28, 45], where $\surd$ is an output over a reserved channel and $\Downarrow$ is defined in terms of may and must, respectively. Clearly, different modalities in general lead to different results; in this paper, proof will be carried out in a 'may' modality, but all our results could be adapted to other modalities. So, formally, $P \Downarrow$ if there exists P' such that $P \Longrightarrow P'$ and $P' \equiv P'' \mid \surd$, for some P''. Finally, for the sake of coherence, we require the notion of success be caught by the semantic theory underlying the calculi, viz. $\asymp$; in particular, we assume that $\asymp$ never relates two processes P and Q such that $P \Downarrow$ and $Q \not\Downarrow$.

> **Property 5 (Success sensitiveness).** *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* success sensitive *if, for every* S*, it holds that* $S \Downarrow$ *if and only if* $[\![S]\!] \Downarrow$.

Notice that Property 5 does not necessarily imply that $[\![\surd]\!] = \surd$ for any valid encoding, even though it is something very natural to have. In general, since $\surd$ is a 0-ary operator, we just have that, by Property 1, $[\![\surd]\!] = C^{\emptyset}_{\surd}()$; but a 0-ary context is simply a constant target process.

3. Some Sample Process Calculi

We now very briefly present the syntax and the operational semantics of the languages we will use in the remainder of this paper; for more details, the interested reader can refer to [6, 12, 16, 37]. All the languages have a common syntax given by

$$P ::= \mathbf{0} \;\mid\; (\nu n)P \;\mid\; P_1|P_2 \;\mid\; !P \;\mid\; \surd$$

As usual, $\mathbf{0}$ is the terminated process, whereas $\surd$ denotes success (see the discussion on Property 5); $P_1|P_2$ denote the parallel composition of two processes; $(\nu n)P$ restricts to P the visibility of n and binds n in P; finally, $!P$ denotes the replication of process P. We have assumed here a very simple way of modeling recursive processes; all what we are going to prove does not rely on this choice and can be rephrased under different forms of recursion.

In all calculi we are going to consider, (νn) is a binder for n in the continuation. In π-derived calculi and in MA, there is also another binder: the input prefix, that binds the input variable in the continuation. Free and bound names, written $\mathrm{Fn}(\cdot)$ and $\mathrm{Bn}(\cdot)$, are defined accordingly.

Terms of this syntax are equated up-to structural congruence, that is the least binary congruence closed under alpha-renaming of bound names and under the following axioms, that are the 'classical' structural laws taken from [38, 39]:

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

$$(\nu n)\mathbf{0} \equiv \mathbf{0} \qquad (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P \qquad P \mid (\nu n)Q \equiv (\nu n)(P \mid Q) \text{ if } n \notin \text{FN}(P)$$

The inference rules that define the operational semantics of processes are:

$$\frac{P \longmapsto P'}{\mathcal{E}(P) \longmapsto \mathcal{E}(P')} \qquad \frac{P \equiv P' \quad P' \longmapsto Q' \quad Q' \equiv Q}{P \longmapsto Q}$$

where $\mathcal{E}(\cdot)$ denotes an *evaluation context*, defined as

$$\mathcal{E}(\cdot) \;::=\; \cdot \;\mid\; \mathcal{E}(\cdot) \mid P \;\mid\; P \mid \mathcal{E}(\cdot) \;\mid\; (\nu n)\mathcal{E}(\cdot)$$

Of course, the operational axioms are specific to every language and are given below.

CCS: it is obtained from the common syntax as follows:

$$P \;::=\; \ldots \;\mid\; \Sigma_{i=1}^{n}\pi_i.P_i \qquad\qquad \pi \;::=\; a \;\mid\; \bar{a}$$

where $\Sigma_{i=1}^{n}\pi_i.P_i$ is the non-deterministic choice between the prefixed processes $\pi_i.P_i$. In CCS, prefixes are just names (ranged over by a) or co-names (ranged over by $\bar{a}$). To fully define the operational semantics, it suffices to consider the following axiom:

$$(\ldots + a.P + \ldots) \mid (\ldots + \bar{a}.Q + \ldots) \longmapsto P \mid Q$$

Moreover, structural equivalence also includes the monoidal laws for sum.

π_a: the *asynchronous π-calculus* is obtained from the common syntax as follows:

$$P \;::=\; \ldots \;\mid\; \overline{a}\langle b\rangle \;\mid\; a(x).P \;\mid\; [a = b]P$$

Here, $\overline{a}\langle b\rangle$ denotes the emission of name b along channel a; $a(x).P$ is an input prefixed process that waits for some name from channel a that will replace x in the continuation P (and is a binder for x in P); finally, $[a = b]P$ is a test for equality of a and b (if the test is passed, then P is activated, otherwise P is blocked for ever). The only reduction axiom is

$$a(x).P \mid \overline{a}\langle b\rangle \longmapsto P\{^{b}/_{x}\}$$

Moreover, structural congruence is extended to handle name matching:

$$[a = a]P \equiv P$$

π_{mix}: the *mixed choice π-calculus* is defined similarly to CCS, but with the possibility of passing/receiving names during a communication and of checking name equality:

$$P \;::=\; \ldots \;\mid\; [a = b]P \;\mid\; \Sigma_{i=1}^{n}\pi_i.P_i \qquad\qquad \pi \;::=\; a(x) \;\mid\; \overline{a}\langle b\rangle$$

Apart from the presence of choices, the only difference with π_a is that in π_{mix} also output actions are prefixes: they block the continuation process until a communication happens. The operational semantics is obtained from the following axiom:

$$(\ldots + a(x).P + \ldots) \mid (\ldots + \overline{a}\langle b\rangle.Q + \ldots) \longmapsto P\{^{b}/_{x}\} \mid Q$$

Moreover, structural equivalence also includes the monoidal laws for sum and the structural axiom given for π_a to handle name matching.

π_{sep}: the *separate choice π-calculus* is the sub-calculus of π_{mix} where every choice contains prefixes of the same kind. It is obtained from the common syntax as follows:

$$P \ ::= \ \ldots \ \mid \ [a = b]P \ \mid \ \Sigma_{i=1}^{n} a_i(x_i).P_i \ \mid \ \Sigma_{i=1}^{n} \overline{a_i}\langle b_i\rangle.P_i$$

The operational and structural axioms are formally identical to the ones for π_{mix}.

π^n and e^π: the *π-calculus with polyadic synchronizations* is defined similarly to π_{mix} but, instead of specifying a single channel name, a tuple of names (of length at most n in π^n or of unbounded length in e^π) is exploited. Formally, π^n and e^π are defined like π_{mix} with prefixes defined as follows:

$$\pi \ ::= \ a_1 \cdot \ldots \cdot a_k(x) \ \mid \ \overline{a_1 \cdot \ldots \cdot a_k}\langle b\rangle \qquad \text{for every } k \leq n$$

$$\pi \ ::= \ a_1 \cdot \ldots \cdot a_k(x) \ \mid \ \overline{a_1 \cdot \ldots \cdot a_k}\langle b\rangle \qquad \text{for every } k$$

The operational axiom is the one of π_{mix}, tailored to polyadic synchronizations:

$$(\ldots + a_1 \cdot \ldots \cdot a_k(x).P + \ldots) \mid (\ldots + \overline{a_1 \cdot \ldots \cdot a_k}\langle b\rangle.Q + \ldots) \longmapsto P\{^b/_x\} \mid Q$$

Again, structural equivalence also includes the monoidal laws for sum and name matching.

MA: the *mobile ambient calculus* is a calculus for modeling mobile and hierarchically distributed processes; it can be obtained from the common syntax as follows:

$$P \ ::= \ \ldots \ \mid \ a[P] \ \mid \ M.P \ \mid \ \langle M\rangle \ \mid \ (x).P$$

$$M \ ::= \ n \ \mid \ in\_a \ \mid \ out\_a \ \mid \ open\_a \ \mid \ M.M$$

The term $a[P]$ denotes a process P located within an ambient named a; of course, P can have as sub-terms other ambients, that are then nested in a. In MA entire ambients can move: an ambient n can enter into another ambient m via the in_m action or exit from another ambient m via the out_m action; moreover, an ambient n can be opened via the $open\_n$ action. Communication is anonymous (no channel name is specified for input/output), can only happen between co-located processes and can exchange sequences of actions, apart from raw names. Formally, the operational semantics is obtained from the following axioms:

$$n[in\_m.P_1 | P_2] \mid m[P_3] \longmapsto m[P_3 \mid n[P_1 | P_2]]$$

$$m[n[out\_m.P_1 | P_2] \mid P_3] \longmapsto n[P_1 | P_2] \mid m[P_3]$$

$$open\_n.P_1 \mid n[P_2] \longmapsto P_1 \mid P_2$$

$$\langle M\rangle \mid (x).P \longmapsto P\{^M/_x\}$$

Moreover, structural congruence also includes the following axioms:

$$(M.M').P \equiv M.(M'.P) \qquad\qquad m[(\nu n)P] \equiv (\nu n)m[P] \ \text{ if } n \neq m$$

and evaluation contexts also include ambient encapsulation:

$$\mathcal{E}(\cdot) \ ::= \ \ldots \ \mid \ a[\mathcal{E}(\cdot)]$$

MA strongly relies on a type system to avoid inconsistent processes like, e.g., $n.P$ or $in\_n[P]$; these two processes can arise after the (ill-typed) communications $(x).x.P \mid \langle n\rangle$ and $(x).x[P] \mid \langle in\_n\rangle$. For MA we only consider the sub-language formed by all the well-typed processes, as defined in [14].

4. Our Criteria for Encodability Results

We now give evidence for supporting our combination of criteria, for proving both encodability and separation results. We start with the former ones: we show that several well known encodings in the literature satisfy them (thus, our criteria accord with the community's common sense), but there are still examples of encodings that do not meet all our criteria (so, our proposal is not trivial).

4.1. Our criteria accord with common sense

Most of our criteria are not new; the only exception is Property 5. Moreover, also Property 2 has some novelty: to the best of our knowledge, it is the first time it has been codified in such a precise and formal way. Nevertheless, the *combination* of these five properties is new and constitutes our proposal. In order to show that it fits well with the common understanding of language expressiveness, we now quickly discuss the most important encodability results for process calculi we are aware of.

First, we want to mention [39], where the *polyadic* (synchronous) π-calculus is encoded into its monadic fragment. The encoding acts homomorphically on all operators, except for

$$\begin{array}{lcl} [\![\, \overline{a}\langle b_1, \ldots, b_k\rangle.P \,]\!] & \triangleq & (\nu c)\overline{a}\langle c\rangle.\overline{c}\langle b_1\rangle.\ldots.\overline{c}\langle b_k\rangle.[\![\, P \,]\!] \\ [\![\, a(x_1, \ldots, x_k).P \,]\!] & \triangleq & a(y).y(x_1).\ldots.y(x_k).[\![\, P \,]\!] \end{array}$$

Clearly, Properties 1–5 are all satisfied; however, as we have explicitly proved in [23, 26], this holds only for the *well-typed* fragment of the calculus (i.e., the set of processes where no arity mismatch between an input and an output over the same channel can arise at runtime).

Another well-established encodability result is from the synchronous (choice-free) π-calculus into π_a; this result appears as two different encodings in [6, 31]. Boudol's encoding is a homomorphism for all operators, except for:

$$\begin{array}{lcl} [\![\, \overline{a}\langle b\rangle.P \,]\!] & \triangleq & (\nu c)(\overline{a}\langle c\rangle \mid c(z).(\overline{z}\langle b\rangle \mid [\![\, P \,]\!])) \\ [\![\, a(x).P \,]\!] & \triangleq & a(y).(\nu d)(\overline{y}\langle d\rangle \mid d(x).[\![\, P \,]\!]) \end{array}$$

whereas Honda and Tokoro's is even simpler:

$$\begin{array}{lcl} [\![\, \overline{a}\langle b\rangle.P \,]\!] & \triangleq & a(z).(\overline{z}\langle b\rangle \mid [\![\, P \,]\!]) \\ [\![\, a(x).P \,]\!] & \triangleq & (\nu d)(\overline{a}\langle d\rangle \mid d(x).[\![\, P \,]\!]) \end{array}$$

These three encodings, that are maybe the best known in process calculi, are all valid, but they all suffer from the fact that their soundness has never been fully established, since the reference criterion was *full abstraction* w.r.t. (weak) bisimilarity. Boudol only proved one direction (viz., that $[\![\, P \,]\!] \asymp_2 [\![\, Q \,]\!]$ implies $P \asymp_1 Q$, where $\asymp$ denotes a Morris-like preorder), whereas [53, 54, 59] showed that full abstraction w.r.t. (weak) bisimilarity does not hold. Possible ways to remedy this lack are weakening the notion of equivalence; two possibilities are barbed congruence closed under *translated contexts* [5, 46] or *typed contexts* [53, 54, 59]. However, by changing the reference equivalence, an encodability result can turn into a separation result: this is what happens to the possibility of encoding the synchronous π-calculus into π_a [11], if the reference semantic theory is must testing [18].

As we have said in the introduction, full abstraction is an orthogonal criterion. Indeed, it is possible to have encodings that enjoy full abstraction but not our criteria. A notable example is the $\mathcal{D}[\![\,\cdot\,]\!]$ encoding of input-guarded choices into the asynchronous π-calculus given in [44]: it is fully abstract w.r.t. weak asynchronous bisimilarity but it introduces divergence. And, as we have just said, there exist valid encodings that are not fully abstract w.r.t. the

expectable notions of equivalences (e.g., strong/weak bisimilarity): the three ones described above are good examples.

Other issues that raised interesting encodability results in process calculi are: external vs internal mobility [5], locality of received names [19, 36], higher-order vs first-order communications [56], depth of prefix nesting [33, 48] and (different forms of) guarded choice [42, 44]. All these encodings have been proved sound by relying on different criteria, ranging from full abstraction (w.r.t. different equivalences) to sensitiveness to different semantic notions (e.g., divergence, deadlock, liveness, ...). On the other hand, they all satisfy our criteria, with the only exception of compositionality that is sometimes weakened by assuming a notion of *two-level* encoding (for a discussion on this issue, see Section 6).

Another aspect related to compositionality emerges from the standard encoding of replication with process definitions:

$$[\![\, !P \,]\!] \triangleq A_P \qquad \text{where} \quad A_P \stackrel{\text{def}}{=} [\![\, P \,]\!] \mid A_P$$

under the assumption that there exists a process constant A_P for every process P. This encoding does not satisfy compositionality, as defined in Property 1: the encoding of $!P$ is not obtained by filling a unary context with $[\![\, P \,]\!]$. This should not be surprising, since process definitions entail two worlds: processes and process definitions. This calls for a refined notion of compositionality (where the two worlds are somehow combined into a single one, e.g. by letting process constant be contexts, with the understanding that they represent their unfolding) or, as we have done in this paper, by exploiting a modeling of recursive processes where everything lives in a single world. For example, recursion would be another possible candidate, since process definitions belong to the language of processes. Indeed, we could model replication via recursion as

$$[\![\, !P \,]\!] \triangleq \mathbf{rec}\, X.([\![\, P \,]\!] \mid X)$$

and this is compositional, using the unary context $\mathbf{rec}\, X.(\_ \mid X)$. Nevertheless, a valid encoding of replication with process definitions can be obtained by letting

$$[\![\, !P \,]\!] \triangleq [\![\, P \,]\!] \mid A_P$$

with A_P defined as above.

To conclude, we want to mention other very well-known encodings in the world of process calculi: the translation of the λ-calculus into the π-calculus [38, 57, 58]. The translations proposed do not fit well with the criteria we have presented because they are *parametric* on an auxiliary name. We shall discuss in Section 6 the problems that we met when trying to extend our framework for accepting also these kinds of translations as valid encodings.

4.2. Our criteria are not trivial

After the previous discussion, one could argue that all the encodings proposed in the literature so far satisfy our criteria. Thus, one could think that it is easy to write valid encodings and so our proposal is not adequate for evaluating encodability results. Actually, this is *not* the case, and a notable example is encoding π_a into MA: this is a non-trivial task, if we want to satisfy all the properties in Section 2. Indeed, in several papers [13, 15, 16] there are attempts to encode π_a into MA, but none of them satisfies operational soundness and divergence reflection.

For example, let us consider the simplest of all the encodings proposed in [13, 15, 16], i.e. the one in [16]. There, the idea is to let

$$\begin{aligned} [\![\, n(x).P \mid \overline{n}\langle m \rangle \,]\!] \triangleq \quad & (\nu p)(\mathtt{io}[in\_n.(x).p[out\_n.[\![\, P \,]\!]]] \mid open\_n) \\ & \mid \mathtt{io}[in\_n.\langle m \rangle] \mid n[!open\_\mathtt{io}] \end{aligned}$$

However, this encoding suffers from two problems (and, indeed, it is not valid according to our definition):

1. *it introduces divergence* Consider for example $!n(x) \mid \overline{n}\langle m\rangle$: it does not diverge but its encoding diverges (infinitely many ambients named `io` originating from the encoding of the replicated input can repeatedly enter ambient n and be opened therein).
2. *it violates operational soundness* Consider for example $n(x) \mid \overline{n}\langle m\rangle \mid \overline{n}\langle m'\rangle$: it can only reduce to either $\overline{n}\langle m\rangle$ or $\overline{n}\langle m'\rangle$ whereas

$$[\![\, n(x) \mid \overline{n}\langle m\rangle \mid \overline{n}\langle m'\rangle \,]\!] \longmapsto^7 \texttt{io}[in\_n.\langle m\rangle] \;\mid\; n[!open\_\texttt{io}] \longmapsto^2 n[!open\_\texttt{io} \mid \langle m\rangle]$$

   where $n[!open\_\texttt{io} \mid \langle m\rangle]$ cannot reduce and it is not equivalent (w.r.t. any 'reasonably defined' notion of equivalence for MA) to $[\![\, \overline{n}\langle m\rangle \,]\!]$ nor to $[\![\, \overline{n}\langle m'\rangle \,]\!]$.

To conclude, notice that the encodings proposed in [13, 15] suffer from the same problems, but are more complicated for typing reasons.

To the best of our knowledge, the encoding we are going to present now (already appeared in [24]) is the first one that fully satisfies operational correspondence without introducing divergence. Moreover, our encoding shows how difficult could be writing valid encodings. The encoding relies on a renaming policy that maps every name a to a triple of pairwise different names (a_1, a_2, a_3); it is a homomorphism w.r.t. all the operators, except for restrictions, inputs and outputs, that are translated as follows:

$$\begin{aligned}
[\![\, (\nu a)P \,]\!] &\triangleq (\nu\, a_1, a_2, a_3)[\![\, P \,]\!] \\
[\![\, \overline{a}\langle b\rangle \,]\!] &\triangleq a_1[a_2[open_a_3.\langle b_1, b_2, b_3\rangle]] \\
[\![\, a(x).P \,]\!] &\triangleq open_a_1.(\nu p, q)(open_p \;\mid\; a_3[in_a_2.open_\texttt{rest} \;\mid\; (x_1, x_2, x_3).in_q.p[out_q.[\![\, P \,]\!]]] \\
&\qquad\qquad \mid\; q[open_a_2.\texttt{rest}[!\,\texttt{rest}[in_a_3.out_q.in_a_2.open_\texttt{rest}]]]) \\
&\quad \text{for } p, q \notin \textsc{Fn}([\![\, P \,]\!]) \cup \{\texttt{rest}, \texttt{poly}, a_1, a_2, a_3, x_1, x_2, x_3\}
\end{aligned}$$

where (x_1, x_2, x_3) is a shortcut for $(x_1).open\_\texttt{poly}.(x_2).open\_\texttt{poly}.(x_3)$ and $\langle b_1, b_2, b_3\rangle$ is a shortcut for $\langle b_1\rangle \mid \texttt{poly}[\langle b_2\rangle \mid \texttt{poly}[\langle b_3\rangle]]$, with `poly` a reserved name.

Our encoding works as follows. For every communication along a, the ambient named a_3 is used as a 'pilot' ambient to enter a_2 and consume the datum associated to b. To reflect the fact that an output along a can be consumed only once, we exploit the outer ambient a_1 and the corresponding $open\_a_1$ action. To avoid interferences that can arise from independent communications along channel a, only one a_3-ambient will be opened within a_2; the (possible) other ones must be rolled back, i.e. reappear at top-level, ready to enter another ambient a_2. This is done by opening a_2 in a restricted ambient q and by leading all the not consumed a_3-ambients out from q via the reserved ambient `rest`, that also restores the in_a_2 capability consumed.

The encoding just presented is valid, i.e. it satisfies all the properties of Section 2, with operational correspondence formulated up-to strong barbed equivalence. All the properties are easy to prove, except for operational soundness and divergence reflection. To carry out the proofs, we found it useful to assign a number to the reductions that the encoding of a communication performs, to easily refer them later on. In what follows, to ease reading, we

enlight the parts of the process that are involved in the generation of the next transition.

$$
\begin{array}{ll}
& [\![\, \overline{a}\langle b\rangle \mid a(x).P \,]\!] \\
\triangleq & \quad a_1\,[a_2[open\_a_3.\langle b_1,b_2,b_3\rangle]] \\
& \quad \mid\ open\_a_1\,.(\nu p,q)(open\_p \mid a_3[in\_a_2.open\_\mathtt{rest} \mid (x_1,x_2,x_3).in\_q.p[out\_q.[\![\, P \,]\!]]] \\
& \qquad\qquad q[open\_a_2.\mathtt{rest}[!\,\mathtt{rest}[in\_a_3.out\_q.in\_a_2.open\_\mathtt{rest}]]]) \\
\longmapsto_{\boxed{1}} & \mathrm{Pr}_a^{\boxed{1}} \triangleq a_2\,[open\_a_3.\langle b_1,b_2,b_3\rangle] \\
& \quad \mid (\nu p,q)(open\_p \mid a_3[\,in\_a_2\,.open\_\mathtt{rest} \mid (x_1,x_2,x_3).\cdots] \mid q[\cdots]) \\
\longmapsto_{\boxed{2}} & \mathrm{Pr}_a^{\boxed{2}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,open\_a_3\,.\langle b_1,b_2,b_3\rangle \mid a_3\,[open\_\mathtt{rest} \mid (x_1,x_2,x_3).\cdots]]) \\
\longmapsto_{\boxed{3}} & \mathrm{Pr}_a^{\boxed{3}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,\langle b_1\rangle \mid \mathtt{poly}[\langle b_2\rangle \mid \mathtt{poly}[\langle b_3\rangle]] \mid open\_\mathtt{rest} \\
& \qquad\quad \mid (x_1)\,.open\_\mathtt{poly}.(x_2).open\_\mathtt{poly}.(x_3).in\_q.p[out\_q.[\![\, P \,]\!]]) \\
\longmapsto_{\boxed{4}} & \mathrm{Pr}_a^{\boxed{4}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,\mathtt{poly}\,[\langle b_2\rangle \mid \mathtt{poly}[\langle b_3\rangle]] \mid open\_\mathtt{rest} \\
& \qquad\quad \mid open\_\mathtt{poly}\,.(x_2).open\_\mathtt{poly}.(x_3).in\_q.p[out\_q.[\![\, P \,]\!]\{^{b_1}/_{x_1}\}]]) \\
\longmapsto_{\boxed{5}} & \mathrm{Pr}_a^{\boxed{5}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,\langle b_2\rangle \mid \mathtt{poly}[\langle b_3\rangle] \mid open\_\mathtt{rest} \\
& \qquad\quad \mid (x_2)\,.open\_\mathtt{poly}.(x_3).in\_q.p[out\_q.[\![\, P \,]\!]\{^{b_1}/_{x_1}\}]]) \\
\longmapsto_{\boxed{6}} & \mathrm{Pr}_a^{\boxed{6}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,\mathtt{poly}\,[\langle b_3\rangle] \mid open\_\mathtt{rest} \\
& \qquad\quad \mid open\_\mathtt{poly}\,.(x_3).in\_q.p[out\_q.[\![\, P \,]\!]\{^{b_1}/_{x_1},\,^{b_2}/_{x_2}\}]]) \\
\longmapsto_{\boxed{7}} & \mathrm{Pr}_a^{\boxed{7}} \triangleq (\nu p,q)(open\_p \mid q[\cdots] \\
& \qquad \mid a_2[\,\langle b_3\rangle \mid open\_\mathtt{rest} \mid (x_3)\,.in\_q.p[out\_q.[\![\, P \,]\!]\{^{b_1}/_{x_1},\,^{b_2}/_{x_2}\}]]) \\
\longmapsto_{\boxed{8}} & \mathrm{Pr}_a^{\boxed{8}} \triangleq (\nu p,q)(open\_p \mid q\,[\cdots] \\
& \qquad \mid a_2[open\_\mathtt{rest} \mid in\_q\,.p[out\_q.[\![\, P\{^b/_x\} \,]\!]]) \\
\longmapsto_{\boxed{9}} & \mathrm{Pr}_a^{\boxed{9}} \triangleq (\nu p,q)(open\_p \mid q[\,open\_a_2\,.\mathtt{rest}[!\,\mathtt{rest}[\cdots]] \mid a_2\,[open\_\mathtt{rest} \mid p[out\_q.[\![\, P\{^b/_x\} \,]\!]]]]) \\
\longmapsto_{\boxed{10}} & \mathrm{Pr}_a^{\boxed{10}} \triangleq (\nu p,q)(open\_p \mid q[\,\mathtt{rest}\,[!\,\mathtt{rest}[\cdots]] \mid open\_\mathtt{rest} \mid p[out\_q.[\![\, P\{^b/_x\} \,]\!]]) \\
\longmapsto_{\boxed{11}} & \mathrm{Pr}_a^{\boxed{11}} \triangleq (\nu p,q)(open\_p \mid q\,[!\,\mathtt{rest}[\cdots] \mid p[\,out\_q\,.[\![\, P\{^b/_x\} \,]\!]]) \\
\longmapsto_{\boxed{12}} & \mathrm{Pr}_a^{\boxed{12}} \triangleq (\nu p,q)(\,open\_p \mid q[!\,\mathtt{rest}[\cdots]] \mid p\,[[\![\, P\{^b/_x\} \,]\!]) \\
\longmapsto_{\boxed{13}} & (\nu q)q[!\,\mathtt{rest}[in\_a_3.out\_q.in\_a_2.open\_\mathtt{rest}]] \mid [\![\, P\{^b/_x\} \,]\!]
\end{array}
$$

What we have just shown immediately yields a proof of operational completeness.

Proposition 4.1. *If $P \longmapsto_1 P'$ then $[\![\, P \,]\!] \Longmapsto_{\asymp_2} [\![\, P' \,]\!]$, where $\asymp_2$ denotes strong barbed equivalence for MA.*

Proof: It suffices to observe that $(\nu q)q[!\,\mathtt{rest}[in\_a_3.out\_q.in\_a_2.open\_\mathtt{rest}]] \asymp_2 \mathbf{0}$. □

By contrast, to prove operational soundness and divergence reflection we have to take care of the possible interferences between the encoding of different communications along the same channel. In such a case, some new reductions (viz, those arising from the replicated copies of ambient `rest`) are needed to restore the interfering a_3 ambients at top-level, ready

to complete their task. However, such reductions are *spurious*, in the sense that they do not correspond to original reductions in π_a and are only performed to remedy some interference.

Formally, a reduction arising from the encoding of a π_a process is called *spurious* if

- it is of kind 2 , but leads an a_3 ambient within an a_2 ambient that has already been entered by (at least) another a_3 ambient. Of course, such an ambient can still be there (i.e., its has still not been opened), or it has been opened, it is within a communication or it has finished communicating. Formally:

$$\begin{array}{l} a_3[\ in\_a_2\ .open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots] \mid \ a_2\ [\cdots \mid a_3[\cdots]] \\ \quad \longmapsto_{2s} a_2[\cdots \mid a_3[\cdots] \mid a_3[open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots]] \end{array}$$

$$\begin{array}{l} a_3[\ in\_a_2\ .open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots] \mid \ a_2\ [\cdots \mid (x_i).\cdots] \\ \quad \longmapsto_{2s} a_2[\cdots \mid (x_i).\cdots \mid a_3[open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots]] \end{array}$$

$$\begin{array}{l} a_3[\ in\_a_2\ .open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots] \mid \ a_2\ [\cdots \mid open\_\texttt{poly}.\cdots] \\ \quad \longmapsto_{2s} a_2[\cdots \mid open\_\texttt{poly}.\cdots \mid a_3[open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots]] \end{array}$$

$$\begin{array}{l} a_3[\ in\_a_2\ .open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots] \mid \ a_2\ [\cdots \mid in\_q.\cdots] \\ \quad \longmapsto_{2s} a_2[\cdots \mid in\_q.\cdots \mid a_3[open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots]] \end{array}$$

  In these cases, we denote the step with 2s to emphasize their spurious nature and distinguish them from a step performed to mimic a reduction in π_a.

- it arises from the content of the a replicated copy of ambient `rest`:

$$\begin{array}{l} q[!\,\texttt{rest}[\ in\_a_3\ .out\_q.in\_a_2.open\_\texttt{rest}] \mid \ a_3\ [open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots]] \\ \quad \longmapsto_{14} q[!\,\texttt{rest}[\cdots] \mid a_3[\ open\_\texttt{rest}\ \mid (x_1,x_2,x_3).\cdots \mid \ \texttt{rest}\ [out\_q.in\_a_2.open\_\texttt{rest}]]] \\ \quad \longmapsto_{15} \ q\ [!\,\texttt{rest}[\cdots] \mid a_3[(x_1,x_2,x_3).\cdots \mid \ out\_q\ .in\_a_2.open\_\texttt{rest}]] \\ \quad \longmapsto_{16} q[!\,\texttt{rest}[\cdots]] \mid a_3[in\_a_2.open\_\texttt{rest} \mid (x_1,x_2,x_3).\cdots] \end{array}$$

  Notice that, after reduction 16 , the content of a_3 is exactly the same as the content of a_3 before performing the 2s reduction: this allows a_3 to complete its communication, by entering *another* a_2 ambient. Indeed, reductions 14 /15 /16 can only happen after reduction 10 ; so, the ambient a_2 previously entered by a_3 has already been dissolved when a_3 is restored at top-level by 16 . This fact intuitively ensures us that no divergence is introduced by the encoding (a formal proof will be given in a few moments).

Let us use metavariable ℓ to range over $\{1,\ldots,16,2s\}$. Then, the numbered reductions defined so far are closed under evaluation contexts and structural congruence:

$$\frac{P \longmapsto_{\ell} P'}{\mathcal{E}(P) \longmapsto_{\ell} \mathcal{E}(P')} \qquad \frac{P \equiv Q \longmapsto_{\ell} Q' \equiv P'}{P \longmapsto_{\ell} P'}$$

Here and in what follows, we denote with n^a_{ℓ} the number of reductions of kind ℓ originated from the encoding of a communication along a in a given sequence of n reductions; n_{ℓ} stands for $\sum_{a\in\mathcal{N}} n^a_{\ell}$.

Theorem 4.2 (Operational soundness). *Let P be a π_a process and Q be a MA process such that $[\![P]\!] \longmapsto^n Q$. Then, $P \longmapsto^{n_{\square}} P'$, for some π_a process P' such that $Q \Longmapsto \asymp_2 [\![P']\!]$, where '$\asymp_2$' denotes strong barbed equivalence in MA.*

Proof: See the Appendix. □

We now exploit the previous result to prove that the encoding does not introduce divergence. To this aim, we first need a preliminary result that relates the number of spurious reductions with the number of initial reductions (i.e., reductions of kind [1]), since only spurious reductions can introduce divergence. It turns out that there are at most polynomially many spurious reductions, and this easily leads us to divergence freedom.

Lemma 4.3. *Let* $[\![P]\!] \longmapsto^n$*; then the number of spurious reductions (i.e.,* $n_{[2s]}+n_{[14]}+n_{[15]}+n_{[16]}$*) is at most* $2 \cdot (n_{[1]})^2 - 2 \cdot n_{[1]}$.

Proof: The worst case is when all the $n_{[1]}$ reductions are on the same channel, say a, and can be obtained as follows. Put all the $n_{[1]}$ a_3 ambients in the same a_2 ambient; this introduces $n_{[1]} - 1$ spurious reductions of kind [2s] and the corresponding $3 \cdot (n_{[1]} - 1)$ reductions (of kind [14] , [15] and [16]) to remedy this choice. Then, put all the remaining $n_{[1]} - 1$ a_3 ambients in the same a_2 ambient; this introduces $4 \cdot (n_{[1]} - 2)$ spurious reductions. And so on. Thus, the overall number of spurious reductions is at most

$$\sum_{k=1}^{n_{[1]}} 4 \cdot (k-1) = 4 \cdot \left(\frac{n_{[1]} \cdot (n_{[1]} + 1)}{2} - n_{[1]} \right) = 2 \cdot (n_{[1]})^2 - 2 \cdot n_{[1]}$$

□

Theorem 4.4 (Divergence reflection). *If* $[\![P]\!] \longmapsto^\omega$*, then* $P \longmapsto^\omega$*.*

Proof: Let $[\![P]\!] \longmapsto^n$ and observe that $n > 0$ implies that $n_{[1]} > 0$. Moreover, for every $k \in \{2, \ldots, 13\}$, it holds that $n_{[k]} \leq n_{[1]}$; indeed, by construction of the encoding, it is not possible to produce a reduction of kind [k] without having produced a corresponding reduction of kind [1] . By Lemma 4.3, $n \to \infty$ implies that $n_{[1]} \to \infty$; by Theorem 4.2, we easily conclude. □

5. Our Criteria for Separation Results

We now show that our criteria allow us to revisit several separation results appearing in the literature, and to prove new separation results.

In the following proofs, we shall sometimes use a labeled transition system (LTS, for short) to describe the possible interactions between two parallel components and the way in which they cooperate to yield a reduction. Traditionally, LTSs have been used to give the operational semantics of a process and to define several different behavioural equivalences for them [20, 21]. Here, we do not need all the (sometimes sophisticated) features of the LTSs for the process calculi we are going to use and only use labeled transitions in an informal and intuitive way. For example, we let $P \xrightarrow{\mu}$ mean that P is able to perform action μ; moreover, as usual, we let $\overline{\mu}$ stand for the complementary action of μ, i.e. an action such that, if $P \xrightarrow{\mu}$ and $P' \xrightarrow{\overline{\mu}}$, then $P \,|\, P' \longmapsto$. For full definitions and discussions, we refer the interested reader to the standard references for the various languages (viz. [57] for all the variants of π-calculi that we consider, [12] for π^e and π^m, and [35] for MA). The only (standard) thing that we assume is the presence of a silent action τ and that every τ action corresponds to a reduction.

5.1. Proving known separation results

Let us start with the separation results in [28], i.e. between π_a/MA and CCS. In *loc. cit.*, it is assumed (a form of) success sensitiveness, homomorphism of '|' and name invariance under any renaming policy that maps every name into a single name. The last two properties, mainly

the last one, are quite demanding. We now prove such results by removing any assumption on the renaming policy and by allowing parallel composition be translated by introducing a centralized coordination process. Thus, we assume that, for every $N \subseteq \mathcal{N}$, there exist $\widetilde{n}$ and R such that $C_{|}^{N}(\__1 \,;\, \__2) = (\nu\widetilde{n})(\__1 \mid \__2 \mid R)$.

Theorem 5.1. *There exists no valid encoding of π_a into CCS.*

Proof: By contradiction, assume that there exists a valid encoding $[\![\cdot]\!]$. Let a, b, c and d be pairwise distinct names and define $S \triangleq [x = b][c = c][d = d]\surd$. Since $(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle b\rangle \longmapsto_1 \surd$, Property 3 implies that $[\![(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle b\rangle]\!] \Longrightarrow_2 T \asymp_2 [\![\surd]\!]$; moreover, by Property 5, $[\![\surd]\!]$ reports success and so does T, since $\asymp_2$ is assumed to be sensitive to success. By Property 1, $[\![(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle b\rangle]\!] \triangleq C_{|}^{\{a,b,c,d\}}([\![a(x).S \mid \mathbf{0}]\!]; [\![\overline{a}\langle b\rangle]\!])$, with $C_{|}^{\{a,b,c,d\}}(\__1 \,;\, \__2) \triangleq (\nu\widetilde{n})(\__1 \mid \__2 \mid T')$, for some $\widetilde{n}$ and T'.

By definition of the operational semantics of CCS, every reduction of a generic CCS process $P \mid Q$ can be: (i) a reduction of P, or (ii) a reduction of Q, or (iii) a synchronization between P and Q, with $P \xrightarrow{a}$ and $Q \xrightarrow{\overline{a}}$ or vice versa. Here, $P \xrightarrow{\mu}$, for $\mu \in \{a, \overline{a}\}$, means that P has a top-level sum containing a summand prefixed by action μ and that a is not restricted in P. Thus, the sequence of reductions $[\![(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle b\rangle]\!] \Longrightarrow_2 T$ is generated by $[\![a(x).S \mid \mathbf{0}]\!]$ and $[\![\overline{a}\langle b\rangle]\!] \mid T'$ by either reducing in isolation or by synchronizing, i.e. $[\![a(x).S \mid \mathbf{0}]\!] \xRightarrow{\mu_1 \ldots \mu_k} T_1$ and $[\![\overline{a}\langle b\rangle]\!] \mid T' \xRightarrow{\overline{\mu}_1 \ldots \overline{\mu}_k} T_2$, for $(\nu\widetilde{n})(T_1 \mid T_2) \equiv T$. where, for every $i \in \{1, \ldots, k\}$, there exists m_i such that $\mu_i \in \{m_i, \overline{m}_i\}$. More explicitly, we can write

$$[\![a(x).S \mid \mathbf{0}]\!] \longmapsto_2^{h_0} H_0 \xrightarrow{\mu_1} K_1 \longmapsto_2^{h_1} H_1 \xrightarrow{\mu_2} K_2 \ldots \longmapsto_2^{h_{k-1}} H_{k-1} \xrightarrow{\mu_k} K_k \longmapsto_2^{h_k} T_1$$

$$[\![\overline{a}\langle b\rangle]\!] \mid T' \longmapsto_2^{h'_0} H'_0 \xrightarrow{\overline{\mu}_1} K'_1 \longmapsto_2^{h'_1} H'_1 \xrightarrow{\overline{\mu}_2} K'_2 \ldots \longmapsto_2^{h'_{k-1}} H'_{k-1} \xrightarrow{\overline{\mu}_k} K'_k \longmapsto_2^{h'_k} T_2$$

Moreover, notice that $\{m_1, \ldots, m_k\} \cap \widetilde{n} = \emptyset$: indeed, $[\![a(x).S \mid \mathbf{0}]\!] \triangleq C_{|}^{\{a,b,c,d\}}([\![a(x).S]\!]; [\![\mathbf{0}]\!]) = (\nu\widetilde{n})([\![a(x).S]\!] \mid [\![\mathbf{0}]\!] \mid T')$ and $[\![a(x).S \mid \mathbf{0}]\!] \xRightarrow{\mu_1 \ldots \mu_k}$ implies that the name occurring in μ_i (viz., m_i) does not belong to $\widetilde{n}$, for every i.

Let σ be the permutation that swaps a with c and b with d; also let σ' denote the permutation of names induced by σ, as defined in Property 2. By Property 2,

$$[\![c(x).S\sigma \mid \mathbf{0}]\!] \longmapsto_2^{h_0} H_0\sigma' \xrightarrow{\mu_1\sigma'} K_1\sigma' \longmapsto_2^{h_1} H_1\sigma' \xrightarrow{\mu_2\sigma'} K_2\sigma' \ldots \longmapsto_2^{h_{k-1}} H_{k-1}\sigma' \xrightarrow{\mu_k\sigma'} K_k\sigma' \longmapsto_2^{h_k} T_1\sigma'$$

$$[\![\overline{c}\langle d\rangle]\!] \mid T' \longmapsto_2^{h'_0} H'_0\sigma' \xrightarrow{\overline{\mu}_1\sigma'} K'_1\sigma' \longmapsto_2^{h'_1} H'_1\sigma' \xrightarrow{\overline{\mu}_2\sigma'} K'_2\sigma' \ldots \longmapsto_2^{h'_{k-1}} H'_{k-1}\sigma' \xrightarrow{\overline{\mu}_k\sigma'} K'_k\sigma' \longmapsto_2^{h'_k} T_2\sigma'$$

Like before, it must be that $\{\sigma'(m_1), \ldots, \sigma'(m_k)\} \cap \widetilde{n} = \emptyset$.

Now, consider $Q \triangleq ((a(x).P \mid \mathbf{0}) \mid \overline{a}\langle d\rangle) \mid ((c(x).P\sigma \mid \mathbf{0}) \mid \overline{c}\langle b\rangle)$. Trivially, $Q \not\Downarrow$ whereas, as we shall see, $[\![Q]\!] \Downarrow$; this yields the desired contradiction. By compositionality,

$$\begin{aligned}
[\![Q]\!] &\triangleq C_{|}^{\{a,b,c,d\}}([\![(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle d\rangle]\!] \,;\, [\![(c(x).S\sigma \mid \mathbf{0}) \mid \overline{c}\langle d\rangle]\!]) \\
&= (\nu\widetilde{n})([\![(a(x).S \mid \mathbf{0}) \mid \overline{a}\langle d\rangle]\!] \mid [\![(c(x).S\sigma \mid \mathbf{0}) \mid \overline{c}\langle d\rangle]\!] \mid T') \\
&\triangleq (\nu\widetilde{n})(C_{|}^{\{a,b,c,d\}}([\![a(x).S \mid \mathbf{0}]\!]; [\![\overline{a}\langle d\rangle]\!]) \mid C_{|}^{\{a,b,c,d\}}([\![c(x).S\sigma \mid \mathbf{0}]\!]; [\![\overline{c}\langle d\rangle]\!]) \mid T') \\
&= (\nu\widetilde{n})((\nu\widetilde{n})([\![a(x).S \mid \mathbf{0}]\!] \mid [\![\overline{a}\langle d\rangle]\!] \mid T') \mid (\nu\widetilde{n})([\![c(x).S\sigma \mid \mathbf{0}]\!] \mid [\![\overline{c}\langle b\rangle]\!] \mid T') \mid T')
\end{aligned}$$

Then, consider

$$
\begin{array}{lll}
[\![Q]\!] & \longmapsto_2^{2(h_0+h'_0)} & (\nu\widetilde{n})((\nu\widetilde{n})(H_0 \mid H'_0) \mid (\nu\widetilde{n})(H_0\sigma' \mid H'_0\sigma') \mid T') \\
& \longmapsto_2 \longmapsto_2 & (\nu\widetilde{n})((\nu\widetilde{n})(K_1 \mid K'_1) \mid (\nu\widetilde{n})(K_1\sigma' \mid K'_1\sigma') \mid T') \qquad (1) \\
& \longmapsto_2^{2(h_1+h'_1)} & (\nu\widetilde{n})((\nu\widetilde{n})(H_1 \mid H'_1) \mid (\nu\widetilde{n})(H_1\sigma' \mid H'_1\sigma') \mid T') \\
& \longmapsto_2 \longmapsto_2 & (\nu\widetilde{n})((\nu\widetilde{n})(K_2 \mid K'_2) \mid (\nu\widetilde{n})(K_2\sigma' \mid K'_2\sigma') \mid T') \qquad (2) \\
& \cdots & \\
& \longmapsto_2^{2(h_{k-1}+h'_{k-1})} & (\nu\widetilde{n})((\nu\widetilde{n})(H_{k-1} \mid H'_{k-1}) \mid (\nu\widetilde{n})(H_{k-1}\sigma' \mid H'_{k-1}\sigma') \mid T') \\
& \longmapsto_2 \longmapsto_2 & (\nu\widetilde{n})((\nu\widetilde{n})(K_k \mid K'_k) \mid (\nu\widetilde{n})(K_k\sigma' \mid K'_k\sigma') \mid T') \qquad (k) \\
& \longmapsto_2^{2(h_k+h'_k)} & (\nu\widetilde{n})((\nu\widetilde{n})(T_1 \mid T_2) \mid (\nu\widetilde{n})(T_1\sigma' \mid T_2\sigma') \mid T') \\
& \equiv & (\nu\widetilde{n})(T \mid (\nu\widetilde{n})(T_1\sigma' \mid T_2\sigma') \mid T')
\end{array}
$$

where the two reductions labeled with (i) are obtained by synchronizing

- μ_i produced by H_{i-1} with $\overline{\mu}_i$ produced by H'_{i-1}, and $\mu_i\sigma'$ produced by $H_{i-1}\sigma'$ with $\overline{\mu'_i}\sigma'$ produced by $H'_{i-1}\sigma'$, if $m_i \notin \varphi_{[\![ ]\!]}(b)$;
- μ_i produced by H_{i-1} with $\overline{\mu}_i$ produced by $H'_{i-1}\sigma'$, and $\mu_i\sigma'$ produced by $H_{i-1}\sigma'$ with $\overline{\mu'_i}\sigma'$ produced by H'_{i-1}, otherwise.

Thus, we have proved that $[\![Q]\!] \Downarrow$, since $(\nu\widetilde{n})(T \mid (\nu\widetilde{n})(T_1\sigma' \mid T_2\sigma') \mid T')$ is successful because T is; by contrast, $Q \not\Downarrow$, in contradiction with the validity of $[\![\cdot]\!]$. □

The previous proof can be adapted to MA: indeed, we can exploit the encoding of channel based communications of π_a into MA given in Section 4.2 and the encoding of name matching in MA provided in [50]. Thus, process $(a(x).[x = b][c = c][d = d]\surd \mid \mathbf{0}) \mid \overline{a}\langle b\rangle$ can be written in MA and the proof then proceeds like above.

Theorem 5.2. *There exists no valid encoding of MA into CCS.*

We now aim at proving other separation results, viz. those in [12, 45, 50, 51], in a more uniform and abstract setting. Our aim is to characterize the features of a language that make an encoding impossible *without committing at any concrete language*, i.e. by only relying on properties of reductions and of success. For example, the separation result in Theorem 5.1 only holds for CCS and π_a; by changing the languages the proof is meaningless. Instead, the proof-techniques put forward by Theorems 5.3, 5.8 and 5.9 are results that do not involve any specific language and can be instantiated (as trivial corollaries) to concrete cases.

To this aim, however, we must leave the ideal framework presented in Section 2 and make it slightly more concrete; carrying out proofs at the abstract level is a challenging open problem. Mainly, we have to make some assumptions on the semantic theory of the target language, viz. '$\asymp_2$'. We propose three possible instantiations that allow us to develop proofs.

5.1.1. First Setting

Definition 5.1. *We say that $\asymp$ is* exact *whenever $P \asymp P'$ and $P \xrightarrow{\mu}$ imply that $P' \stackrel{\mu}{\Rightarrow}$, for every $\mu \neq \tau$.*

Examples of exact equivalences are (the different kinds of) synchronous bisimilarity and synchronous trace equivalence. Regretfully, under this assumption, we are able to develop proofs only under a restricted formulation of Property 1.

Definition 5.2. *We say that* $[\![\cdot]\!]$ *is* homomorphic (w.r.t. '|') *whenever* $C^N_{|}(\_1;\_2)$*, the context used to compositionally translate the parallel composition of two processes with free names in N, is* $\_1 \mid \_2$*, for every set of names N.*

This requirement is similar to the one in [12, 28, 45, 46, 50, 51], where only encodings that translate '|' homomorphically are considered. In such works, this requirement is defended by saying that only encodings that do not reduce the degree of parallelism of any source process can be 'valid' means for language comparison. Reducing the degree of parallelism is seen as a weakness of encodings in the setting of process calculi: the target language is not able to mimic the source one without reducing the degree of distribution. We agree with this argument and, hence, we believe that working with homomorphic encodings is not a too high price to be payed.

We are now ready to prove our first proof-technique.

Theorem 5.3. *Assume that there is a* $\mathcal{L}_1$*-process* S *such that* $S \not\longmapsto_1$*,* $S \not\Downarrow$ *and* $S \mid S \Downarrow$*; moreover, assume that every* $\mathcal{L}_2$*-process* T *that does not reduce is such that* $T \mid T \not\longmapsto_2$*. If* $\asymp_2$ *is exact, there cannot exist any valid and homomorphic encoding* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$

Proof: We work by contradiction. First, let us fix, for every $\mathcal{L}_1$-process S that does not reduce, a $\mathcal{L}_2$-process $f([\![S]\!])$ such that $[\![S]\!] \Longmapsto_2 f([\![S]\!]) \not\longmapsto_2$; such a process always exists because of Property 4 (when $[\![S]\!]$ does not reduce, we can always let $f([\![S]\!]) = [\![S]\!]$). Now, consider the auxiliary encoding $(\!| \cdot |\!) : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ such that:

$$(\!| S |\!) \triangleq \begin{cases} f([\![S]\!]) & \text{if } S \not\longmapsto_1 \\ (\!| S_1 |\!) \mid (\!| S_2 |\!) & \text{if } S = S_1 \mid S_2 \longmapsto_1 \\ [\![S]\!] & \text{otherwise} \end{cases}$$

Such an encoding satisfies the following two properties:

$$\text{A. if } S \not\longmapsto_1 \text{ then } (\!| S |\!) \not\longmapsto_2 \qquad\qquad \text{B. } (\!| S |\!) \asymp_2 [\![S]\!]$$

Property A follows by construction of $(\!| \cdot |\!)$; let us prove Property B, by induction on the structure of S. If $S \not\longmapsto_1$ (base step and first sub-case of the inductive step), then, by operational completeness (that is part of Property 3), we have that $[\![S]\!] \Longmapsto_2 f([\![S]\!])$ implies the existence of a S' such that $S \Longmapsto_1 S'$ and $f([\![S]\!]) \Longmapsto_2 \asymp_2 [\![S']\!]$. Since $S \not\longmapsto_1$, we have that S' can only be S itself; moreover, the fact that $f([\![S]\!]) \not\longmapsto_1$ implies that $(\!| S |\!) \asymp_2 [\![S]\!]$, as desired. If $S = S_1 \mid S_2 \longmapsto_1$ then, by structural induction, $(\!| S_1 |\!) \asymp_2 [\![S_1]\!]$ and $(\!| S_2 |\!) \asymp_2 [\![S_2]\!]$; we easily conclude by congruence of $\asymp_2$ with respect to parallel composition. The third sub-case is trivial, by reflexivity of $\asymp_2$.

Now, let us take a $\mathcal{L}_1$-process S such that $S \not\longmapsto_1$, $S \not\Downarrow$ and $S \mid S \Downarrow$; by Property 5 and homomorphism, we have that $[\![S]\!] \not\Downarrow$ and $[\![S \mid S]\!] \triangleq [\![S]\!] \mid [\![S]\!] \Downarrow$. This implies that $[\![S]\!] \mid [\![S]\!] \longmapsto_2$, with $[\![S]\!] \xrightarrow{\mu}$ and $[\![S]\!] \xrightarrow{\overline{\mu}}$, for some pair of complementary actions μ and $\overline{\mu}$ (here we are assuming binary synchronizations, as often happens in process calculi). Since $\asymp_2$ is 'exact', we can use property B to obtain that $(\!| S |\!) \xrightarrow{\mu}$ and $(\!| S |\!) \xrightarrow{\overline{\mu}}$; thus, $(\!| S |\!) \mid (\!| S |\!) \longmapsto_2$ whereas, by $S \not\longmapsto_1$ and property A, $(\!| S |\!) \not\longmapsto_2$, in contradiction with the hypothesis. □

Corollary 5.4. *There exists no valid and homomorphic encoding of* π_{mix}*, CCS and MA into* π_{sep} *and into* π_a*.*

Proof: Take any exact behavioural theory for π_{sep} (e.g., strong/branching/weak bisimilarity, both in their early/late/open form, or may/must/fair testing, just to mention some possibilities). On one hand, notice that, if T is a π_{sep}-process such that $T \mid T \longmapsto_2$, then

$T \equiv (\nu\widetilde{n})(\Sigma_{i=1}^{m} a_i(x_i).T_i \mid \Sigma_{j=1}^{n} \overline{a'_j}\langle b_j\rangle.T'_j \mid T'')$ and there exist $i \in \{1,\ldots,m\}$ and $j \in \{1,\ldots,n\}$ such that $a_i = a'_j$. Thus, trivially, $T \longmapsto_2$; hence, every π_{sep}-process T that does not reduce is such that $T \mid T \not\longmapsto_2$.

On the other hand, we can find both in CCS, in π_{mix} and in MA a process S that does not reduce and does not report success, but such that $S \mid S$ reports success: it suffices to let S be $a.\mathbf{0} + \bar{a}.\surd$ in CCS, $a(x).\mathbf{0} + \overline{a}\langle b\rangle.\surd$ in π_{mix} and $(\nu p)(open\_p.\surd \mid n[in\_n.p[out\_n.out\_n.\mathbf{0}]])$ in MA.

The case for π_a is similar: just notice that now if T is a π_a-process such that $T \mid T \longmapsto_2$, then $T \equiv (\nu\widetilde{n})(a(x).T \mid \overline{a}\langle b\rangle \mid T')$. □

5.1.2. Second Setting

Definition 5.3. *We say that* $\asymp$ *is* reduction sensitive *whenever* $P \asymp P'$ *and* $P' \longmapsto$ *imply that* $P \longmapsto$.

Examples of reduction sensitive equivalence/preorders are strong synchronous/asynchronous bisimulation [1, 37] and the expansion preorder [2]. Working with a reduction sensitive $\asymp_2$ has the advantage that we are able to carry out proofs also under translations of '|' more liberal than the homomorphic one.

A Uniform Approach to Separation Results. We now describe the methodological approach we shall follow to prove separation results. The key fact that will enable all our proofs is the following (adapted from [23] and corresponding to property A in the proof of Theorem 5.3). By using the terminology of [44], this proposition implies that in this setting we only consider *prompt* encodings.

Proposition 5.5. *If* $\asymp_2$ *is reduction sensitive and* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ *is a valid encoding, then* $S \not\longmapsto_1$ *implies that* $[\![S]\!] \not\longmapsto_2$, *for every* S.

Proof: By contradiction, assume that $[\![S]\!] \longmapsto_2 T$, for some $S \not\longmapsto_1$. By operational soundness, there exists a S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 T' \asymp_2 [\![S']\!]$; but the only such S' is S itself. Since $\asymp_2$ is reduction sensitive and since $[\![S']\!] = [\![S]\!] \longmapsto_2$, then $T' \longmapsto_2 T''$. Again, by operational soundness $T'' \Longmapsto_2 T''' \asymp_2 [\![S]\!]$, and so on; thus, $[\![S]\!] \longmapsto_2 T \Longmapsto_2 T' \longmapsto_2 T'' \Longmapsto_2 T''' \longmapsto_2 \ldots$, in contradiction with Property 4 (since $S \not\longmapsto_1$ implies that S does not diverge). □

Another crucial consequence of our criteria are the following two propositions. Here and in what follows, let us assume the following notation: $block(S)$ denotes any term S' such that $\text{Fn}(S') = \text{Fn}(S)$, $S' \not\longmapsto_1$, $S' \not\Downarrow_1$ and S' cannot interact with any other $\mathcal{L}_1$-process. It is easy to build such a S': it suffices to prefix S with any blocking action involving a new restricted name (for example, in CCS and in any of the π-calculi we can prefix S with an input from a new restricted channel; in MA with an *open* of a new restricted ambient; and so on).

Proposition 5.6. *Let* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ *be a valid encoding and* $\asymp_2$ *be reduction sensitive. Then, for every set of names N, it holds that* $C_{|}^{N}(\__1;\__2)$ *has both its holes at top-level.*

Proof: Let us fix a set of names N and a $\mathcal{L}_1$-process S with $\text{Fn}(S) = N$. Let us now consider $S' \triangleq \surd \mid block(S)$. By Proposition 5.5, it must be that $[\![S']\!] \not\longmapsto_2$, since $S' \not\longmapsto_1$. By Property 1, we have that $[\![S']\!] \triangleq C_{|}^{N}([\![\surd]\!]; [\![block(S)]\!])$. By Property 5, it must be that $[\![S']\!] \Downarrow_2$, since $S' \Downarrow_1$. All these facts entail that the top-level occurrence of $\surd$ in $[\![S']\!]$ is exhibited

- either by the translating context, and so $C_{|}^{N}(\__1;\__2) \Downarrow_2$,

- or by $[\![\surd]\!]$, but this implies that $C_{|}^{N}(\__1;\__2)$ has $\__1$ at top-level.

Indeed, it is not possible that $\surd$ is exhibited by $[\![block(S)]\!]$, since $block(S) \not\Downarrow_1$. However, the first case is not possible, otherwise $[\![block(S) \,|\, block(S)]\!] \Downarrow_2$, whereas $block(S) \,|\, block(S) \not\Downarrow_1$. To show that also the occurrence of $\__2$ in $C_{|}^{N}(\__1;\__2)$ is at top-level, it suffices to reason in the very same way, but using $S' \triangleq block(S) \mid \surd$. □

Proposition 5.7. *Let $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ be a valid encoding and $\asymp_2$ be reduction sensitive. If there exist two $\mathcal{L}_1$-terms S_1 and S_2 such that $S_1 \mid S_2 \Downarrow$, with $S_i \not\Downarrow$ and $S_i \not\mapsto_1$ for $i = 1, 2$, then $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto_2$.*

Proof: By Property 5, $[\![S_1 \mid S_2]\!] \Downarrow_2$ and, by Proposition 5.6, it has both $[\![S_1]\!]$ and $[\![S_2]\!]$ at top-level. However, since none of $[\![S_1]\!]$, $[\![S_2]\!]$ and $[\![block(S_1) \mid block(S_2)]\!]$ can report success, it must be that $[\![S_1 \mid S_2]\!] \longmapsto_2$. This can only happen by synchronizing $[\![S_1]\!]$ and $[\![S_2]\!]$. If this was not the case, we would have that $[\![S_1 \mid block(S_2)]\!] \longmapsto_2$ or $[\![block(S_1) \mid S_2]\!] \longmapsto_2$ or $[\![block(S_1) \mid block(S_2)]\!] \longmapsto_2$, in violation of Proposition 5.5: indeed, $S_1 \mid block(S_2) \not\mapsto_1$ because $S_1 \not\mapsto_1$, $block(S_2) \not\mapsto_1$ and $block(S_2)$ cannot interact with S_1; a similar reasoning holds for $block(S_1) \mid S_2$ and $block(S_1) \mid block(S_2)$. □

In this framework, the way in which we prove a separation result between $\mathcal{L}_1$ and $\mathcal{L}_2$ is the following:

(*a*) by contradiction, suppose that there exists a valid encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$;
(*b*) find a pair of $\mathcal{L}_1$-processes S_1 and S_2 that satisfy the hypothesis of Proposition 5.7; by such a result, $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto_2$;
(*c*) from S_2 obtain a process S_2' with the same free names as S_2 but such that $S_1 \mid S_2' \not\mapsto_1$ and $[\![S_1]\!] \mid [\![S_2']\!] \longmapsto_2$;
(*d*) by Property 1, this implies that $[\![S_1 \mid S_2']\!] \longmapsto_2$, in contradiction with Proposition 5.5.

Notice that the identification of S_1 and S_2 (point (*b*) above) is usually very simple: they are directly obtained from the constructs of $\mathcal{L}_1$ that one believes not to be encodable into $\mathcal{L}_2$. This is different from [12, 28, 45, 50, 51] where, by contrast, a lot of effort must be spent to define a programming scenario that can be properly implemented in the source language but not in the target one. Point (*c*) is the only part that requires some ingenuity (it can be easy or quite difficult): usually, it strongly relies on Property 2 (sometimes also on compositionality) to slightly modify S_2 in order to obtain the new process S_2'.

A Simpler Proof of Known Separation Results. First, we reformulate Theorem 5.3 by changing the hypothesis on $\asymp_2$; this modification will allow us to obtain Corollary 5.4 under a different choice of semantic theories for π_{sep}.

Theorem 5.8. *Assume that there is a $\mathcal{L}_1$-process S such that $S \not\mapsto_1$, $S \not\Downarrow$ and $S \mid S \Downarrow$; moreover, assume that every $\mathcal{L}_2$-process T that does not reduce is such that $T \mid T \not\mapsto_2$. Also assume that $\asymp_2$ is reduction sensitive. Then, there cannot exist any valid encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.*

Proof: By contradiction. Let S be such that $S \not\mapsto_1$, $S \not\Downarrow$ and $S \mid S \Downarrow$; by Proposition 5.7, $[\![S]\!] \mid [\![S]\!] \longmapsto_2$ that, by hypothesis, implies that $[\![S]\!] \longmapsto_2$, in contradiction with Proposition 5.5. □

We now give a second proof-technique that allows us to obtain the hierarchy for polyadic synchronizations in [12] and to adapt the results in [23, 25] to the present setting. To this aim, let us define the *matching degree* of a language $\mathcal{L}$, written $\textsc{Md}(\mathcal{L})$, as the least upper

bound on the number of names that must be matched to yield a reduction in $\mathcal{L}$. For example, the matching degree of CCS [37], of the π-calculus [37] and of Mobile Ambients [16] is 1; the matching degree of Dπ [29] is 2; the matching degree of π^n (the π-calculus with n-ary polyadic synchronizations [12]) is n; the matching degree of e^π (the π-calculus with arbitrary polyadic synchronizations [12]) is ∞. Indeed, as a representative sample, the π-calculus process $a(x).P \mid \overline{a}\langle b\rangle.Q$ can reduce because of the successful matching between the channel name specified for input and for output (a here).[2]

Theorem 5.9. *If* $\text{MD}(\mathcal{L}_1) > \text{MD}(\mathcal{L}_2)$ *and* $\asymp_2$ *is reduction sensitive, then there exists no valid encoding* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.

Proof: By contradiction assume the existence of a valid encoding $[\![\cdot]\!]$. Pick up two $\mathcal{L}_1$-processes S_1 and S_2 that satisfy the hypothesis of Proposition 5.7 and that synchronize only once (before reporting success) by matching exactly $k = \text{MD}(\mathcal{L}_1)$ names, viz. $\{n_1, \ldots, n_k\}$. By Proposition 5.7, their encodings must synchronize: i.e., $[\![S_1]\!] \xrightarrow{\mu}$ and $[\![S_2]\!] \xrightarrow{\overline{\mu}}$. Since $\text{MD}(\mathcal{L}_1) > \text{MD}(\mathcal{L}_2)$, it must be that the names in $\text{FN}(\mu) \cap \text{FN}(\overline{\mu})$ matched when synchronizing μ and $\overline{\mu}$ (say, $\{m_1, \ldots, m_h\}$) are less than k; this implies the existence of an n_i such that $\varphi_{[\![\,]\!]}(n_i) \cap \{m_1, \ldots, m_h\} = \emptyset$. Let us choose a new name m (i.e., $m \notin \text{FN}(S_1) \cup \text{FN}(S_2) \cup \text{FN}(\mu) \cup \text{FN}(\overline{\mu})$) and consider the substitution σ that swaps m and n_i. Trivially, $S_1 \mid S_2\sigma \not\mapsto_1$ because, by construction, S_1 and S_2 can only synchronize by matching k names; thus, also S_1 and $S_2\sigma$ can only synchronize by matching k names and the match on the i-th name fails, since S_1 contains n_i and $S_2\sigma$ contains m. By Property 2, $[\![S_2\sigma]\!] = [\![S_2]\!]\sigma'$ and $[\![S_2]\!]\sigma' \xrightarrow{\overline{\mu}\sigma'}$. Now, notice that $\overline{\mu}\sigma'$ is still synchronizable with μ because σ' swaps component-wise $\varphi_{[\![\,]\!]}(n_i)$ and $\varphi_{[\![\,]\!]}(m)$, and so it does not touch $\{m_1, \ldots, m_h\}$; so, $[\![S_1]\!] \mid [\![S_2\sigma]\!] \longmapsto_2$. By Proposition 5.6, $[\![S_1 \mid S_2\sigma]\!] \longmapsto_2$ whereas $S_1 \mid S_2\sigma \not\mapsto_1$, in contradiction with Proposition 5.5. □

Corollary 5.10. *There exists no valid encoding from* e^π *into* π^m*, for every* m*, and from* π^m *into* π^n*, whenever* $m > n$.

Proof: Observe that $\text{MD}(e^\pi) = \infty$ and that $\text{MD}(\pi^m) = m$; then apply Theorem 5.9. □

5.1.3. Third Setting

The setting presented in Section 5.1.2 relies on the assumption that $\asymp_2$ is reduction sensitive. This restriction seems us not too severe, since most of the operational correspondence results appearing in the literature are formulated up to such semantic theories; the only notable exception we are aware of is [42, 44], where weak (asynchronous) bisimilarity [1] is exploited. We now sketch a weaker setting, that covers all the separation results we are aware of (including [42, 44]) without breaking the elegant and powerful proof-techniques developed in Section 5.1.2.

We have said that the aim of formulating operational correspondence up to $\asymp_2$ is to get rid of junk processes possibly arising from the encoding. We can make this intuition explicit by formulating operational correspondence as follows:

- for all $S \Longmapsto_1 S'$, there exist $\widetilde{n}$ and T' such that $[\![S]\!] \Longmapsto_2 (\nu\widetilde{n})([\![S']\!] \mid T') \asymp_2 [\![S']\!]$;
- for all $[\![S]\!] \Longmapsto_2 T$, there exist S', $\widetilde{n}$ and T' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 (\nu\widetilde{n})([\![S']\!] \mid T') \asymp_2 [\![S']\!]$.

[2] Incidentally, the early-style LTS for the π-calculus also verifies that $\xrightarrow{\overline{a}b}$ synchronizes with $\xrightarrow{ab}$. However, this does not imply that the matching degree of the π-calculus is 2. Indeed, the process that generates label ab can generate label ac, for every name c; thus, the only name that is matched is the name of the communication channel (a in this case), whereas the second name (viz. b) is only a parameter exchanged.

Maybe, such a formulation can be criticized by saying that it is too 'syntactic', but in practice we are not aware of any encoding that does not satisfy it. Moreover, encodings work at a syntactic level on processes; thus, in the same vein as Properties 1 and 2, we think that it is acceptable to also have syntactic conditions on operational correspondence.

Restricting $\asymp_2$ to pairs of kind $((\widetilde{\nu n})(T \mid T'), T)$, for $(\widetilde{\nu n})(T \mid T') \asymp_2 T$, yields a reduction sensitive relation, for any $\asymp_2$; thus, Propositions 5.5, 5.6 and 5.7 (and, consequently, all the results proved in Section 5.1.2) hold also in this setting without requiring reduction sensitiveness of $\asymp_2$.

5.2. Proving new separation results

We now prove in the settings of Sections 5.1.2 and 5.1.3 that MA cannot be encoded into π_{mix} (actually, the proof scales well to the general π-calculus, as presented in [40]) and that it cannot encode CCS (actually, its mixed choice operator).

The proofs will be carried out by following the general methodology presented in Section 5.1.2. A challenging issue for future research is the development of analogous proofs to the setting of Section 5.1.1 or, even better, without making any assumption on '$\asymp_2$'. However, when the target language is MA, the first setting seems not very adequate, since the bisimulation equivalences for MA [35] are *not* 'exact'.

Theorem 5.11. *Let '$\asymp_2$' satisfy the assumptions in Section 5.1.2 or 5.1.3. Then, there exists no valid encoding of MA into π_{mix}.*

Proof: Consider the processes $S_1 \triangleq n[in\_m.p[out\_n.out\_m.\surd]]$ and $S_2 \triangleq m[\mathbf{0}] \mid open\_p$, where $S_1|S_2 \Downarrow$, $S_1 \not\Downarrow$ and $S_2 \not\Downarrow$; by Proposition 5.7, it must be that $[\![S_1|S_2]\!] \longmapsto_2$ with the contribution of both $[\![S_1]\!]$ and $[\![S_2]\!]$.

We first prove that in π_{mix} we have that $[\![S_1|S_2]\!] \equiv (\widetilde{\nu n})(([\![S_1]\!] + \ldots) \mid ([\![S_2]\!] + \ldots) \mid T)$, with $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto_2$. By Proposition 5.6, $[\![S_1|S_2]\!] = C_{\mid}^{\mathrm{FN}(S_1,S_2)}([\![S_1]\!]; [\![S_2]\!])$, with both $[\![S_1]\!]$ and $[\![S_2]\!]$ occurring at top-level. So, for $i \in \{1,2\}$, we have that $\__i$ is underneath some restriction, or it is replicated, or, if the outermost operator of $[\![S]\!]$ is a prefix for every S, then $\__i$ can also be a summand of some mixed sum. In all these cases, we easily conclude.

Because $[\![S_1]\!]$ and $[\![S_2]\!]$ must synchronize, we have that $[\![S_1]\!]$ performs an output over some channel a and $[\![S_2]\!]$ performs a corresponding input from a (the case in which the input and the output are swapped is identical). Because of compositionality, $[\![S_1]\!] \triangleq C_{n[\,]}^{\{n,m,p\}}([\![in\_m.p[out\_n.out\_m.\surd]]\!])$; thus, the output over a can be either performed by $C_{n[\,]}^{\{n,m,p\}}(\cdot)$ or by $[\![in\_m.p[out\_n.out\_m.\surd]]\!]$. In both cases we could violate Proposition 5.5: in the first case $[\![n[\langle n\rangle \mid \langle m\rangle \mid \langle p\rangle] \mid S_2]\!] \longmapsto_2$, in the second case $[\![in\_m.p[out\_n.out\_m.\surd] \mid S_2]\!] \longmapsto_2$. □

Theorem 5.12. *Let '$\asymp_2$' satisfy the assumptions in Section 5.1.2 or 5.1.3. Then, there exists no valid encoding of CCS into MA.*

Proof: Let $P \triangleq \bar{x}.a.b.c + y.\surd$ for a, b, c pairwise distinct; let P_{mn} denote $\bar{m}.a.b.c + n.\surd$, i.e. the process $P\sigma$, where σ is the substitution that maps x in m and y in n. By construction, $P_{ab} \mid P_{bc} \Downarrow$ whereas $P_{mn} \not\Downarrow$ for every m and n; by Proposition 5.6, $[\![P_{ab} \mid P_{bc}]\!] = (\widetilde{\nu n})([\![P_{ab}]\!] \mid [\![P_{bc}]\!] \mid T)$ and, by Proposition 5.7, it must be that $[\![P_{ab}]\!] \mid [\![P_{bc}]\!] \longmapsto_2$. In MA, this can happen in three ways (symmetric cases are omitted because they do not add anything new to the proof):

1. $[\![P_{ab}]\!]$ performs an output and $[\![P_{bc}]\!]$ performs an input;
2. $[\![P_{ab}]\!]$ wants to open an ambient m and $[\![P_{bc}]\!]$ has such a top-level unrestricted ambient;

3. $[\![P_{ab}]\!]$ has a top-level ambient that wants to enter into an ambient m and $[\![P_{bc}]\!]$ has an unrestricted top-level occurrence of ambient m.

We now show that all these cases lead to contradict Proposition 5.5.

1. Consider $[\![P_{ba} \mid P_{bc}]\!]$. By compositionality, we have that $[\![P_{ba} \mid P_{bc}]\!] = C_{|}^{\{a,b,c\}}([\![P_{ba}]\!]; [\![P_{bc}]\!]) = (\nu\widetilde{n})([\![P_{ba}]\!] \mid [\![P_{bc}]\!] \mid T)$; moreover, by name invariance, also $[\![P_{ba}]\!]$ performs an output. Thus, $[\![P_{ba}]\!] \mid [\![P_{bc}]\!] \longmapsto_2$; hence $[\![P_{ba} \mid P_{bc}]\!] \longmapsto_2$, whereas $P_{ba} \mid P_{bc} \not\longmapsto_1$.
2. Because of compositionality, $[\![P_{bc}]\!] = C_{+}^{\{a,b,c\}}([\![\bar{b}.a.b.c]\!]; [\![c.\surd]\!])$; thus, ambient m is exhibited either by $C_{+}^{\{a,b,c\}}(\__1; \__2)$, or by $[\![c.\surd]\!]$, or by $[\![\bar{b}.a.b.c]\!]$. In the first case, we would also have that $[\![P_{ab} \mid P_{cb}]\!] \longmapsto_2$, while in the second case we would also have that $[\![P_{ab} \mid P_{ac}]\!] \longmapsto_2$; hence, the only way to respect Proposition 5.5 is the third possibility. So, $C_{+}^{\{a,b,c\}}(\__1; \__2)$ must have $\__1$ at top-level.

   With a similar reasoning, we can show that $[\![b.\surd]\!]$ performs the $open\_m$ action and that $C_{+}^{\{a,b,c\}}(\__1; \__2)$ has also $\__2$ at top-level.

   Now consider P_{bb}: its encoding is $[\![\bar{b}.a.b.c + b.\surd]\!] \triangleq C_{+}^{\{a,b,c\}}([\![\bar{b}.a.b.c]\!]; [\![b.\surd]\!]) \equiv (\nu\widetilde{m})([\![\bar{b}.a.b.c]\!] \mid [\![b.\surd]\!] \mid T')$, for some $\widetilde{m}$ and T', since $C_{+}^{\{a,b,c\}}(\__1; \__2)$ has both $\__1$ and $\__2$ at top-level. Thus, $[\![P_{bb}]\!] \longmapsto_2$, since $[\![\bar{b}.a.b.c]\!]$ wants to open m and $[\![b.\surd]\!]$ provides an unrestricted occurrence of m at top-level. By contrast, $P_{bb} \not\longmapsto_1$, in violation of Proposition 5.5.
3. Like in the previous case, we can conclude that $[\![\bar{b}.a.b.c]\!]$ exhibits ambient m and that $C_{+}^{\{a,b,c\}}(\__1; \__2)$ has a top-level occurrence of $\__1$. Moreover, either $[\![b.\surd]\!]$ has the ambient aiming at entering into m, or $C_{+}^{\{a,b,c\}}(\__1; \__2)$ is of the form $(\nu\widetilde{h})(h[\__1 \mid Q] \mid \ldots)$ and $[\![b.\surd]\!]$ performs an in_m action. In both cases, we can reason like in case 2 above to obtain that $[\![P_{aa}]\!] \longmapsto_2$ and conclude.

□

Of course, the previous proof scales straightforwardly to π_{mix}, by considering process $P \triangleq \overline{x}\langle x\rangle.a(x_1).b(x_2).c(x_3) \mid y(z).\surd$. Thus, we have also proved that

Theorem 5.13. *Let '$\asymp_2$' satisfy the assumptions in Section 5.1.2 or 5.1.3. Then, there exists no valid encoding of π_{mix} into MA.*

6. On Enhanced Forms of Translations

Before concluding, we would like to briefly discuss if the criteria of Section 2 can be enhanced by mainly showing to what extent the first two properties can be formulated in a more liberal way.

Concerning Property 1, all the encodings we are aware of satisfy some form of compositionality, and it would be very hard to concretely define an encoding that does not satisfy any form of such a property. Indeed, if a translation exists, we should be able to write it down, and this is feasible only if it can be defined inductively (since the syntactic terms can be arbitrary long). However, this fact does not imply that our formulation of compositionality is the only possibility for having an inductively defined translation. For example, as it happens in [4, 7], we can have a 'two-level' encoding: $[\![\cdot]\!]$ is a translation that satisfies Properties 2–5 and it is such that $[\![P]\!] \triangleq C^{\mathrm{FN}(P)}((\!| P |\!))$, where $(\!| \cdot |\!)$ is a compositional translation (this property is called *weak* compositionality in [49]). The proof-techniques presented in Sections 5.1.2 and 5.1.3 can be readily adapted to this enhanced notion of encoding, whereas the proof-technique of Section 5.1.1 cannot (recall that there we had to work with homomorphic translations of parallel composition).

Also name invariance is somehow related to the inductive definition of the translation: such an induction is usually carried out over the syntactic structure of the source term, and this justifies the fact that, for source terms that differ only by an injective name substitution, the encoded terms should have the same syntactic structure. However, our formulation of Property 2 does not scale well to *parametric* translations, that are functions from pairs of the form $(S, \mathit{Parameters})$ to target processes instead of being functions from source processes to target ones (as we have considered throughout this paper). Consider, for example, the encodings in [34, 38, 57, 58]; there, an encoding is a family of translations $[\![\cdot]\!]_\Xi$, where the parameter Ξ is a set of names (in the first one) or a single name (in the last three ones) used as auxiliary parameters in the translation. Moreover, one can also imagine different translation schemata, where Ξ represents, e.g., an upper bound on the free names of the source process. In these cases, our framework is less adequate: indeed, it is difficult to formulate our properties and carry out proofs without knowing what the index represents. For example, which is the initial (i.e., top-level) value of Ξ in $[\![\cdot]\!]_\Xi$? If Ξ are names, are they part of the source or of the target language? The latter question is very delicate when defining Property 2: in the first case, the property should be adapted by requiring that $[\![S\sigma]\!]_{\Xi\sigma}$ is equal/equivalent to $([\![S]\!]_\Xi)\sigma'$; in the second case, we have that $[\![S\sigma]\!]_{\Xi\sigma'}$ must be equal/equivalent to $([\![S]\!]_\Xi)\sigma'$. Moreover, does compositionality have to somehow take into account Ξ or not? Even more, operational correspondence should be formulated by keeping Ξ the same or let it evolve along reductions (i.e., $S \Longmapsto_1 S'$ implies $[\![S]\!]_\Xi \Longmapsto_2 \asymp_2 [\![S']\!]_\Xi$ or $[\![S]\!]_\Xi \Longmapsto_2 \asymp_2 [\![S']\!]_{\Xi'}$, for some – which? – Ξ')?

Thus, even if we believe that parametric forms of encoding are very reasonable, we have problems in defining a *general* framework without specifying anything on the index. On the contrary, we can properly *specialize* our criteria from case to case, according to the rôle of the index in the encoding.

7. Conclusion

We have collected together some criteria that an encoding should satisfy to be considered a valid means for language comparison. We have argued that the resulting set of criteria is a satisfactory notion for assessing the relative expressive power of process calculi by noting that most encodings appearing in the literature satisfy them. Moreover, this notion is not trivial, because there exist known encodings that do not satisfy all the criteria we have proposed: a representative example is given by the encodings of the π-calculus into Mobile Ambients [14, 16].

This paper is mostly methodological, as it describes a new approach both to encodability and to separation results. On one hand, we believe that, for encodability results, we have proposed a valid alternative to full abstraction for comparing languages: our proposal is more focused on expressiveness issues, whereas full abstraction is more appropriate when we look for a tight correspondence between the behavioural equivalences associated with the compared languages. We think that full abstraction is still an interesting notion to investigate when developing an encoding, but it should be considered an "extra-value": if it holds, the encoding is surely more interesting, because it enables not only a comparison of the languages, but also of their associated equivalences. On the other hand, our proposal is also interesting for separation results: as we have shown, several separation results appearing in the literature can be easily formulated and proved in terms of our criteria. In Table 1 we have comparatively listed such results. Roughly speaking, the approach taken in [12, 28, 45, 50, 51] consists in (*i*) identifying a problem that can be solved in the source language but not in the target, and then (*ii*) finding the least set of criteria that an encoding should meet to translate a solution of the problem in the source into a solution of the problem in the target. Concerning point (*ii*), we have already argued that the criteria put forward by our criteria are not more

	Electoral Systems	Matching Systems	Our Criteria		
			1^{st} setting	2^{nd} setting	3^{rd} setting
CCS $\not\rightarrow \pi_{sep}$	[45] (a)	$\times$	✓	✓	✓
$\pi_{mix} \not\rightarrow \pi_{sep}$	[45] (a)	$\times$	✓	✓	✓
MA $\not\rightarrow \pi_{sep}$	[50] (a)	$\times$	✓	✓	✓
$e^{\pi} \not\rightarrow \pi^{m} \not\rightarrow \pi^{n}$ $(m > n)$	$\times$	[12] (c)	?	✓	✓
MA $\not\rightarrow \pi_{mix}$	$\times$	$\times$	?	✓	✓
CCS $\not\rightarrow$ MA	$\times$	$\times$	//	✓	✓
$\pi_{a} \not\rightarrow$ CCS	[45] (b)	[28] (a)	✓		
MA $\not\rightarrow$ CCS	[51] (b)	[28] (a)	✓		

Table 1: Comparison of different separation methodologies. For every result, we list where it appears ('$\times$' if it has never been published, '?' if we believe that it holds but we have not been able to prove it, and '//' if the setting does not apply) and the criteria adopted: (a) stands for homomorphism w.r.t. '|', (a form of) name invariance and (a form of) success sensitiveness; (b) is (a) plus a condition requiring that source processes without shared free names must be translated into target processes without shared free names; (c) is (a) plus divergence reflection.

demanding than those in [12, 28, 45, 50, 51]. Concerning point (i), we are only aware of two kinds of problem: *symmetric electoral systems* [45, 50, 51] and *matching systems* [12, 28]. However, none of them is 'universal', in the sense that different separation results usually require different separation problems (see the '$\times$' in Table 1).

Apart from being very adequate for proving known separation results, our approach can be used to prove several new separation results: in [22, 24], we exploit such criteria to compare the relative expressive power of several calculi for mobility (viz., the asynchronous π-calculus, a distributed π-calculus, and Mobile/Safe/Boxed Ambients together with several of their variants); moreover, the results in [23, 25, 26] can be easily re-formulated under Properties 1–5 (actually, the criteria we assume in this paper generalize the criteria in those papers without compromising the validity of the results appearing therein). Finally, the fact that our criteria are also well-suited for encodability results makes the hierarchies of languages uniform.

Of course, there is still a lot of work to do. For example, with the general formulation of our criteria (see Section 2) we have only been able to prove the last two separation results of Table 1, even though we strongly believe that also the remaining ones hold. It would be nice to prove more separation results in the general framework: in that setting, such results are very strong, since the formulation of our criteria is more liberal and abstract. Moreover, it would be nice to replace the two '?' in Table 1 with a '✓'. Another very challenging direction for future research is to prove existence of a valid encoding without giving a concrete translation. Such 'existence results' are very common in mathematics and physics; to the best of our knowledge, no such result has ever appeared for studying the expressiveness of process calculi. To conclude, the challenge raised in [43] is still open, but we think and hope that our proposal can contribute to its final solution.

Acknowledgments: Part of this work has been carried out while I was visiting the PPS Laboratory (Université Paris Diderot, France); there, I benefited from fruitful and insightful discussions with Daniele Varacca, whose criticism and suggestions radically improved an earlier draft of this work. I am also very grateful to Jesus Aranda and Frank Valencia for their comments and discussions. Thanks also to Thomas Given-Wilson and Barry Jay, that suggested me to include the remark in footnote 1, and to Ruben Monjaraz, that raised the question on compositionality with process definitions (see Section 4.1). I am also very grateful to the anonymous reviewers for their constructive attitude and for several suggestions that

improved the presentation of this work.

Appendix A. Technicalities and Proofs from Section 4.2

To ease reading, let us denote with $\mathtt{PR}_{\bar{a}}$ the encoding of $\overline{a}\langle b\rangle$, for some b, with its enclosing a_1 ambient dissolved, i.e.,

$$\mathtt{PR}_{\bar{a}} \triangleq a_2[open\_a_3.\langle b_1, b_2, b_3\rangle]$$

and with $\mathtt{PR}_a$ the process left by the encoding of some input over a used to remedy to reductions of kind $\boxed{2s}$, i.e.

$$\mathtt{PR}_a \triangleq q[!\,\mathtt{rest}[in\_a_3.out\_q.in\_a_2.open\_\mathtt{rest}]]$$

Moreover, given any ambient process P, we denote with $P^{s_2,s_{14},s_{15}}$ the processes P without the (possible) top-level restrictions on p and q, and that moreover contains:

- s_2 parallel copies of process

  $$\mathtt{P}_{a_3}^{\boxed{2s}} \triangleq a_3[open\_\mathtt{rest} \mid (x_1, x_2, x_3).in\_q'.p'[\cdots]]$$

  within the top-level ambient named a_2, if there is such an ambient in P, or within the top-level ambient named q, otherwise;

- s_{14} parallel copies of process

  $$\mathtt{P}_{a_3}^{\boxed{14}} \triangleq a_3[open\_\mathtt{rest} \mid (x_1, x_2, x_3).in\_q'.p'[\cdots] \mid \mathtt{rest}[out\_q.in\_a_2.open\_\mathtt{rest}]]$$

  and s_{15} parallel copies of process

  $$\mathtt{P}_{a_3}^{\boxed{15}} \triangleq a_3[(x_1, x_2, x_3).in\_q'.p'[\cdots] \mid out\_q.in\_a_2.open\_\mathtt{rest}]$$

  within the top-level ambient named q.

Of course, we shall use notation $P^{s_2,s_{14},s_{15}}$ only for those P's that either have a top-level ambient a_2 or q. Then, working only with such processes allows us to add also the following rule for inferring numbered reductions:

$$\frac{P \longmapsto_{\boxed{\ell}} Q}{P^{s_2,s_{14},s_{15}} \longmapsto_{\boxed{\ell}} Q^{s_2,s_{14},s_{15}}}$$

Before going on, we want to remark that, to be precise, we would have to index processes $\mathtt{PR}_{\bar{a}}$ and $\mathtt{PR}_a^{\boxed{k}, s_2, s_{10}, s_7}$ not only with the name of the channel (viz., a), but also with the communicated name (say, b) in the first case and with the continuation process (say, P) in the second case. However, this would have made our notation much heavier; thus, to ease reading, we prefer to be less precise and rely on the reader's understanding.

We can now prove operational soundness (i.e. the second item of Property 3). To this aim, it suffices to prove the following lemma, where we let $\widehat{n^a_{\boxed{k}}}$ be $n^a_{\boxed{1}} - n^a_{\boxed{2}} - n^a_{\boxed{2s}} + n^a_{\boxed{16}}$, if $k = 1$, and be $n^a_{\boxed{k}} - n^a_{\boxed{k+1}}$, for every $k = 2, \ldots, 12$.

Lemma Appendix A.1. *Let P be a π_a process and Q be an MA process such that $[\![P]\!] \longmapsto^n Q$, for $n = \sum_{\ell\in\{1,\ldots,16,2s\}} n_{\boxed{\ell}}$. Then,*

$$Q \equiv (\nu\widetilde{m}, \widetilde{p}, \widetilde{q})\left([\![R]\!] \mid \prod_{a\in\mathcal{N}} \left(\prod_{k=1}^{n^a_{\boxed{2s}} - n^a_{\boxed{16}}} \mathtt{PR}_{\bar{a}} \mid \prod_{k=1}^{12} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathtt{PR}_a^{\boxed{k}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}} \mid \prod_{i=1}^{n^a_{\boxed{13}}} \mathtt{PR}_a^{s_{2_{13_i}}, s_{14_{13_i}}, s_{15_{13_i}}} \right) \right)$$

where

- $[\![R]\!]$ *has no top-level restrictions;*
- $\widetilde{p}$ *and* $\widetilde{q}$ *have the same length;*
- $s_{14_{k_i}} = s_{15_{k_i}} = 0$*, whenever* $k < 11$*; and*
- $n^a_{\boxed{2s}} = \sum_{k=1}^{13} \sum_{i=1}^{n^a_{\boxed{k}}} s_{2_{k_i}}$, $n^a_{\boxed{14}} = \sum_{k=1}^{13} \sum_{i=1}^{n^a_{\boxed{k}}} s_{14_{k_i}}$ *and* $n^a_{\boxed{15}} = \sum_{k=1}^{13} \sum_{i=1}^{n^a_{\boxed{k}}} s_{15_{k_i}}$.

Proof: By induction on n. The base step is trivial; for the inductive step, let $[\![P]\!] \longmapsto^n Q \longmapsto Q'$. By induction,

$$Q \equiv (\nu \widetilde{m}, \widetilde{p}, \widetilde{q}) \left([\![R]\!] \mid \prod_{a \in \mathcal{N}} \left(\prod_{k=1}^{n^a_{\boxed{2s}} - n^a_{\boxed{16}}} \mathtt{PR}_{\bar{a}} \mid \prod_{k=1}^{12} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathtt{PR}_a^{\boxed{k}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}} \mid \prod_{i=1}^{n^a_{\boxed{13}}} \mathtt{PR}_a^{s_{2_{13_i}}, s_{14_{13_i}}, s_{15_{13_i}}} \right) \right)$$

Let us now consider all the possible cases for the reduction $Q \longmapsto Q'$:

- $[\![R]\!]$ can only evolve in isolation by performing a reduction of kind $\boxed{1}$;
- $\mathtt{PR}_a^{\boxed{1}, s_{2_{1_i}}, s_{14_{1_i}}, s_{15_{1_i}}}$ can evolve in isolation by performing a reduction of kind $\boxed{2}$, or it can interact with some $\mathtt{PR}_{\bar{a}}$ by still performing a reduction of kind $\boxed{2}$ or it can perform a reduction of kind $\boxed{2s}$ by interacting with some $\mathtt{PR}_a^{\boxed{h}, s_{2_{h_j}}, s_{14_{h_j}}, s_{15_{h_j}}}$, for $h \leq 8$;
- $\mathtt{PR}_a^{\boxed{k}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}}$, for $k > 1$, can only evolve in isolation by performing a reduction of kind $\boxed{k+1}$;
- $\mathtt{PR}_a^{s_{2_{13_i}}, s_{14_{13_i}}, s_{15_{13_i}}}$ can only evolve in isolation by performing a reduction of kind $\boxed{14}$ (provided that $s_{2_{13_i}} > 0$), $\boxed{15}$ (provided that $s_{14_{13_i}} > 0$) or $\boxed{16}$ (provided that $s_{15_{13_i}} > 0$).

Let us consider all these cases in isolation.

1. The reduction has been originated by $[\![R]\!]$. In this case, the reduction must be of kind $\boxed{1}$ and, hence, $R \equiv \overline{a}\langle b \rangle \mid a(x).P \mid R'$. The thesis easily follows by noting that, after the $(n+1)$-th reduction, the new value of $\widehat{n^a_{\boxed{1}}}$ is the old value plus one: the new process of the form $\mathtt{PR}_a^{\boxed{1}, s_2, s_{10}, s_7}$ arising from this reduction is $a_2[open\_a_3.\langle b_1, b_2, b_3 \rangle] \mid open\_p \mid a_3[in\_a_2.open\_\mathtt{rest} \mid (x_1, x_2, x_3).in\_q.p[out\_q.[\![P]\!]]]$. Moreover, the restricted p and q can be scope extended and added to $\widetilde{p}$ and $\widetilde{q}$, respectively. Finally, $[\![R']\!]$ has no top-level restrictions, since $[\![R]\!]$ has none.
2. The reduction has been originated by $\mathtt{PR}_a^{\boxed{k}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}}$, for some $k \in \{1, \ldots, 12\}$ and $i \in \{1, \ldots, \widehat{n^a_{\boxed{k}}}\}$. We reason by case analysis.
   (a) The reduction is of kind $\boxed{k+1}$. In this case, $\mathtt{PR}_a^{\boxed{k}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}} \longmapsto_{\boxed{k+1}} \mathtt{PR}_a^{\boxed{k+1}, s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}}$ and we can easily conclude since the new value of $\widehat{n^a_{\boxed{k}}}$ is the old value minus one and the new value of $\widehat{n^a_{\boxed{k+1}}}$ is the old value plus one.

   When $k = 1$, this case also keeps into account the possible reductions arising from an interaction between $\mathtt{PR}_a^{\boxed{1}, s_{2_{1_i}}, s_{14_{1_i}}, s_{15_{1_i}}}$ and one of the $n^a_{\boxed{2s}} - n^a_{\boxed{16}}$ processes $\mathtt{PR}_{\bar{a}}$ of the first product. Indeed, by using structural equivalence, we can swap the process of kind $\mathtt{PR}_{\bar{a}}$ occurring as a parallel component of $\mathtt{PR}_a^{\boxed{1}, s_{2_{1_i}}, s_{14_{1_i}}, s_{15_{1_i}}}$ with the one occurring in the first product; then the new version of $\mathtt{PR}_a^{\boxed{1}, s_{2_{1_i}}, s_{14_{1_i}}, s_{15_{1_i}}}$ can reduce in isolation.

(b) If $k = 1$, the reduction can also be of kind $\boxed{2s}$. In this case, there exist $h \in \{1,\dots,8\}$ and $j \in \{1,\dots,\widehat{n^a_{\boxed{h}}}\}$ such that $\mathrm{PR}_a^{\boxed{1},s_{2_{1_i}},s_{14_{1_i}},s_{15_{1_i}}} \mid \mathrm{PR}_a^{\boxed{h},s_{2_{h_j}},s_{14_{h_j}},s_{15_{h_j}}} \longmapsto_{\boxed{2s}} \mathrm{PR}_{\bar{a}} \mid \mathrm{PR}_a^{\boxed{h},s_{2_{h_j}}+1,s_{14_{h_j}},s_{15_{h_j}}}$. Also in this case we can conclude by letting the new value of $s_{2_{h_j}}$ be the old value plus one, and by noting that the new value of $\widehat{n^a_{\boxed{1}}}$ is the old value minus one and the new value of $n^a_{\boxed{2s}}$ is the old value plus one.

(c) If $k \in \{11,12\}$, it can also be one of the following cases:

- The reduction is of kind $\boxed{14}$. In this case, $\mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}} \longmapsto_{\boxed{14}} \mathrm{PR}_a^{\boxed{k},s_{2_{k_i}}-1,s_{14_{k_i}}+1,s_{15_{k_i}}}$ and we can easily conclude by letting the new value of $s_{2_{k_i}}$ be the old value minus one and the new value of $s_{14_{k_i}}$ be the old value plus one.
- The reduction is of kind $\boxed{15}$. In this case, $\mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}} \longmapsto_{\boxed{15}} \mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}}-1,s_{15_{k_i}}+1}$ and reason similarly to the previous case.
- The reduction is of kind $\boxed{16}$. In this case, $\mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}} \longmapsto_{\boxed{16}} \mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}-1} \mid a_3[\cdots]$. In this case, it suffices to notice that the new value of $n^a_{\boxed{16}}$ is the old value plus one, the new value of $\widehat{n^a_{\boxed{1}}}$ is the old value minus one and that the ambient $a_3[\cdots]$ together with the copy of $\mathrm{PR}_{\bar{a}}$ removed from the first product (arising from the decrement of $n^a_{\boxed{16}}$) form the new $\mathrm{PR}_a^{\boxed{1},0,0,0}$ that is added to the product containing processes of this kind (and this justifies the increase of the value of $\widehat{n^a_{\boxed{1}}}$).

3. The reduction has been originated by $\mathrm{PR}_a^{\boxed{13},s_{2_{13_i}},s_{14_{13_i}},s_{15_{13_i}}}$, for some $i \in \{1,\dots,n^a_{\boxed{13}}\}$. This case is similar to case 2(c) above.

□

Proof of Theorem 4.2.. By Lemma Appendix A.1,

$$Q \equiv (\nu\widetilde{m},\widetilde{p},\widetilde{q})\left([\![R]\!] \mid \prod_{a\in\mathcal{N}} \left(\prod_{k=1}^{n^a_{\boxed{2s}}-n^a_{\boxed{16}}} \mathrm{PR}_{\bar{a}} \mid \prod_{k=1}^{12} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}} \mid \prod_{i=1}^{n^a_{\boxed{13}}} \mathrm{PR}_a^{s_{2_{13_i}},s_{14_{13_i}},s_{15_{13_i}}} \right) \right)$$

Then, reduce every $\mathrm{PR}_a^{\boxed{k},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}}$ such that at least one of $s_{2_{k_i}}, s_{14_{k_i}}, s_{15_{k_i}}$ is different from 0 to $\mathrm{PR}_a^{\boxed{13},s_{2_{k_i}},s_{14_{k_i}},s_{15_{k_i}}}$. We now obtain a process of the form

$$(\nu\widetilde{m},\widetilde{p},\widetilde{q})\left([\![R']\!] \mid \prod_{a\in\mathcal{N}} \left(\prod_{k=1}^{n^a_{\boxed{2s}}-n^a_{\boxed{16}}} \mathrm{PR}_{\bar{a}} \mid \prod_{k=1}^{12} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}'} \mathrm{PR}_a^{\boxed{k},0,0,0} \mid \prod_{i=1}^{n^a_{\boxed{13}}'} \mathrm{PR}_a^{s_{2_{13_i}},s_{14_{13_i}},s_{15_{13_i}}} \right) \right)$$

where $[\![R']\!]$ collects together all the processes of the form $[\![P\{^b/_x\}]\!]$ that are part of the processes $\mathrm{PR}_a^{\boxed{13},s_{2_{k_i}},s_{10_{k_i}},s_{7_{k_i}}}$ obtained after this sequence of reductions.

Now consider all processes of kind $\mathrm{PR}_a^{s_{2_{13_i}},s_{14_{13_i}},s_{15_{13_i}}}$ such that at least one of $s_{2_{13_i}}, s_{14_{13_i}}, s_{15_{13_i}}$ is different from 0. By construction of the encoding, it must always be that $s_{2_{13_i}} \geq s_{14_{13_i}} \geq s_{15_{13_i}}$. Thus, the number of processes of this kind is exactly $n^a_{\boxed{2s}} - n^a_{\boxed{16}}$; furthermore, for every such process, there exists a corresponding $\mathrm{PR}_{\bar{a}}$ process in the first product. We can now perform all the possible actions of kind $\boxed{14}$, $\boxed{15}$ and $\boxed{16}$; this leads to a process of the form

$$(\nu\widetilde{m},\widetilde{p},\widetilde{q})\left([\![R']\!] \mid \prod_{a\in\mathcal{N}} \prod_{k=1}^{12} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}'+n^a_{\boxed{2s}}-n^a_{\boxed{16}}} \mathrm{PR}_a^{\boxed{k},0,0,0} \right)$$

where now all the $a_3[\cdots]$ (produced from some $\mathrm{PR}_a^{s_{2_{13_i}}, s_{14_{13_i}}, s_{15_{13_i}}}$ after all these reductions) together with the corresponding process of kind $\mathrm{PR}_{\tilde{a}}$ now become a new $\mathrm{PR}_a^{\boxed{1},0,0,0}$. Now, let us reduce every $\mathrm{PR}_a^{\boxed{k},0,0,0}$ to $\mathrm{PR}_a^{\boxed{k+1},0,0,0}$ until we obtain a process of the form

$$[\![\, P' \,]\!] \;\mid\; (\nu\widetilde{q}) \prod_{a\in\mathcal{N}} \prod_{k=1}^{12} \widehat{n^a_{\boxed{k}}}' + n^a_{\boxed{25}} - n^a_{\boxed{16}} \prod_{i=1} \mathrm{PR}_a^{0,0,0}$$

that, by what we have observed in the proof of Proposition 4.1, is barbed equivalent to $[\![\, P' \,]\!]$. To conclude, we just note that P' is the process obtained from P by performing the $n_{\boxed{1}}$ reductions associated to the reductions of kind $\boxed{1}$ in $[\![\, P \,]\!] \longmapsto^n Q$. □

References

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

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

[3] M. Baldamus, J. Parrow, and B. Victor. Spi-calculus translated to pi-calculus preserving may-tests. In *Proc. of LICS*, pages 22–31. IEEE Computer Society, 2004.

[4] M. Baldamus, J. Parrow, and B. Victor. A fully abstract encoding of the *i*-calculus with data terms. In *Proc. of ICALP*, volume 3580 of LNCS, pages 1202–1213. Springer, 2005.

[5] M. Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theoretical Computer Science*, 195(2):205–226, 1998.

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

[7] M. Bugliesi and M. Giunti. Secure implementations of typed channel abstractions. In *Proc. of POPL*, pages 251–262. ACM, 2007.

[8] N. Busi, M. Gabbrielli and G. Zavattaro Replication vs. Recursive Definitions in Channel Based Calculi. Proc. of *ICALP'03*, volume 2719 of *LNCS*, pages 133-144. Springer, 2003.

[9] N. Busi, M. Gabbrielli and G. Zavattaro Comparing Recursion, Replication, and Iteration in Process Calculi. Proc. of *ICALP'04*, volume 3142 of *LNCS*, pages 307-319. Springer, 2004.

[10] D. Cacciagrano, F. Corradini, J. Aranda, and F. D. Valencia. Linearity, persistence and testing semantics in the asynchronous pi-calculus. *Proc. of EXPRESS'07*, ENTCS 194(2):59–84, 2008.

[11] D. Cacciagrano, F. Corradini, and C. Palamidessi. Separation of synchronous and asynchronous communication via testing. *Theoretical Computer Science*, 386(3):218–235, 2007.

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

[13] L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for mobile ambients. In *Proc. of ICALP'99*, volume 1644 of LNCS. Springer, 1999.

[14] L. Cardelli, G. Ghelli, and A. D. Gordon. Types for the ambient calculus. *Information and Computation*, 177(2):160–194, 2002.

[15] L. Cardelli and A. D. Gordon. Types for mobile ambients. In *Proc. of POPL'99*, pages 79–92. ACM, 1999.

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

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

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

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

[20] R.J. van Glabbeek. The linear time - branching time spectrum. Proc. of *CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer, 1990.

[21] R.J. van Glabbeek. The linear time - branching time spectrum II; the semantics of sequential systems with silent moves. Proc. of *CONCUR'93*, volume 715 of *LNCS*, pages 66–81. Springer, 1993.

[22] D. Gorla. On the Relative Expressive Power of Ambient-based Calculi. Proc. of *TGC'08*, volume 5474 of *LNCS*, pages 141–156. Springer, 2009.

[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. On the Relative Expressive Power of Calculi for Mobility. In *Proc. of MFPS XXV*, ENTCS 249, pages 269–286. Elsevier, 2009.

[25] D. Gorla. Synchrony vs asynchrony in communication primitives. In *Proc. of EXPRESS'06*, ENTCS 175(3), pages 87–108. Elsevier, 2007.

[26] D. Gorla. Comparing communication primitives via their relative expressive power. *Information and Computation*, 206(8):931–952, 2008.

[27] D. Gorla. Towards a unified approach to encodability and separation results for process calculi. In *Proc. of CONCUR'08*, volume 5201 of LNCS, pages 492–507. Springer, 2008.

[28] B. Haagensen, S. Maffeis, and I. Phillips. Matching systems for concurrent calculi. In *Proc. of EXPRESS'07*, ENTCS 194(2):85–99, 2008.

[29] M. Hennessy and J. Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation*, 173:82–120, 2002.

[30] M. Herlihy. Impossibility and universality results for wait-free synchronization. In *Proc. of PODC*, pages 276–290. ACM Press, 1988.

[31] 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.

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

[33] C. Laneve and B. Victor. Solos in concert. *Proc. of ICALP'99*, volume 1644 of LNCS, pages 513–523. Springer, 1999.

[34] F. Levi. A Typed Encoding of Boxed into Safe Ambients. *Acta Informatica*, 42(6):429–500, 2006.

[35] M. Merro and F. Z. Nardelli. Behavioral theory for mobile ambients. *Journal of the ACM*, 52(6):961–1023, 2005.

[36] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. In *Proc. of ICALP'98*, volume 1443 of LNCS, pages 856–867. Springer, 1998.

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

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

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

[40] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I/II. *Information and Computation*, 100:1–77, 1992.

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

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

[43] U. Nestmann. Welcome to the jungle: A subjective guide to mobile process calculi. In *Proc.of CONCUR'06*, volume 4137 of LNCS, pages 52–63. Springer, 2006.

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

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

[46] C. Palamidessi, V. A. Saraswat, F. D. Valencia, and B. Victor. On the expressiveness of linearity vs persistence in the asychronous pi-calculus. In *Proc. of LICS*, pages 59–68. IEEE Computer Society, 2006.

[47] C. Palamidessi and F. Valencia. Recursion vs Replication in Process Calculi: Expressiveness. *Bulletin of the EATCS*, 87:105–125, 2005.

[48] J. Parrow. Trios in concert. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Foundations of Computing, pages 621–637. MIT Press, 2000.

[49] J. Parrow. Expressiveness of process algebras. In *Emerging Trends in Concurrency*, ENTCS 209:173–186, 2008.

[50] I. Phillips and M. Vigliotti. Electoral systems in ambient calculi. *Information and Computation*, 206(1):34–72, 2008.

[51] I. Phillips and M. Vigliotti. Leader election in rings of ambient processes. *Theoretical Computer Science*, 356(3):468–494, 2006.

[52] G.D. Plotkin. A strucutral approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60-61:17–139, 2004.

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

[54] P. Quaglia and D. Walker. Types and full abstraction for polyadic pi-calculus. *Information and Computation*, 200:215–246, 2005.

[55] J. Rathke, V. Sassone, and P. Sobocinski. Semantic barbs and biorthogonality. In *Proc. of FoSSaCS*, volume 4423 of LNCS, pages 302–316. Springer, 2007.

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

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

[58] V. Vasconcelos. Processes, Functions, and Datatypes. *Theory and Practice of Object Systems*, 5(2):97110, 1999.

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