Proc. of *CONCUR'08*, LNCS 5201, pages 492–507. © Springer.

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

Daniele Gorla

Dip. di Informatica, Univ. di Roma "La Sapienza"
PPS - Université Paris Diderot & CNRS, France

Abstract. In this paper, 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 good means for language comparison. We argue that the combination of such criteria is a valid proposal by noting that: (*i*) the best known encodings appeared in the literature satisfy them; (*ii*) this notion is not trivial, because there exist encodings that do not satisfy all the criteria we have proposed; (*iii*) the best 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 way in which we prove known separation results is easier and more uniform than the way in which such results were originally proved.

1 Introduction

As argued in [27], one of the hottest topic 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 [31].

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 in another one, also because of the proliferation of different process calculi.

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 in 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 [9, 17, 29, 32, 33], 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 "good" encoding of the latter in 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 [29, 32, 33] with those in [9, 17]). 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 good 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 [15] but it was formulated in a too demanding way.

Of course, in order to support our proposal, we have to give evidences of its reasonableness. To this aim, we exhibit both philosophical and pragmatic arguments. From the pragmatic side, we notice that most of the best known encodings appeared in the literature satisfy our criteria and that their combination is not trivial, because there exist some encodings (namely, the encodings of π-calculus in Mobile Ambients proposed in [10, 11]) that do not satisfy all the criteria we propose. Moreover, we also prove that the best known separation results can be formulated and proved (in an easier and more uniform way) in terms of our criteria; furthermost, some widely believed (but never proved) separation results can be now formally proved by using the criteria we propose (this task is carried out in [14]). The philosophical part is, instead, 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 that gets rid of 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. Then, in Section 3, we show how 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. In Section 4, we conclude by summing up our main contributions and discussing future work.

For space limitations, we shall work with process calculi (CCS [22]; the asynchronous π-calculus, π_a [5]; the separate and mixed choice π-calculus, π_{sep} and π_{mix} [35]; Mobile Ambients, MA [11]; the π-calculus with polyadic synchronizations, e^{π} and π^n [9]) without defining them; a sketch of their syntax and operational semantics is in the Appendix.

2 The Encodability Criteria

In this section we discuss the criteria an encoding should satisfy to be considered a good 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 $\mathbf{0}$ by at least using the parallel composition operator '$|$'.
- $\longmapsto$ 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*. As usual, $\Longmapsto$ denotes the reflexive and transitive closure of $\longmapsto$. To compositionally reason on process reductions, we shall also assume a labeled transition relation, $\xrightarrow{\mu}$, whose τ's characterize $\longmapsto$.
- $\asymp$ is a behavioural equivalence/preorder, 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 [25] or can be derived directly from the reduction semantics [20].

A *translation* of $\mathcal{L}_1 = (\mathcal{P}_1, \longmapsto_1, \asymp_1)$ into $\mathcal{L}_2 = (\mathcal{P}_2, \longmapsto_2, \asymp_2)$, written $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$, is a function from $\mathcal{P}_1$ into $\mathcal{P}_2$. We shall call *encoding* any translation that satisfies the criteria we are going to present. Moreover, to simplify reading, we let S range over processes of the source language (viz., $\mathcal{L}_1$) and T range over processes of the target language (viz., $\mathcal{L}_2$). 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 replaced the holes $\{\_{}_1; \ldots; \_{}_k\}$. 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 $\textsc{Fn}(\cdot)$) 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 op of $\mathcal{L}_1$ and for every subset of names N, there exists a k-ary context $C^N_{\mathsf{op}}[\_{}_1; \ldots; \_{}_k]$ such that, for all $S_1, \ldots, S_k$ with $\textsc{Fn}(S_1, \ldots, S_k) = N$, it holds that $[\![\mathsf{op}(S_1, \ldots, S_k)]\!] = C^N_{\mathsf{op}}[[\![S_1]\!]; \ldots; [\![S_k]\!]]$.

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 [8] or of persistent fragments of the asynchronous π-calculus [7]. However, for separation results, the most widely accepted criterion is homomorphism of parallel composition [9, 17, 29, 30, 32, 33]; 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, 6, 26].

Our definition of compositionality allows two processes that only differ in their free names to have totally different translations: indeed, it could be that $C^N_{\mathsf{op}}[\ldots]$ is very different from $C^M_{\mathsf{op}}[\ldots]$, whenever $N \neq M$. We want to avoid this fact; indeed, a "good" translation cannot depend on the particular names involved in the source process, but only on its syntactic structure. However, it is possible that a translation fixes some names to play a precise rôle or it can translate a single name into a tuple of names. Thus, every translation assumes a *renaming policy*, that we now formally define.

Definition 1 (Renaming policy). *Given a translation $[\![\cdot]\!]$, its underlying renaming policy is a function $\varphi_{[\![\,]\!]} : \mathcal{N} \longrightarrow \mathcal{N}^k$, for some constant $k > 0$, such that $\forall u, v \in \mathcal{N}$ with $u \neq v$, it holds that $\varphi_{[\![\,]\!]}(u) \cap \varphi_{[\![\,]\!]}(v) = \emptyset$, where $\varphi_{[\![\,]\!]}(\cdot)$ is simply considered a set here.*

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 $\varphi_{[\![\,]\!]} : \mathcal{N} \longrightarrow \mathcal{N}$. 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 $\varphi_{[\![\,]\!]}(n_i) \triangleq n_{i+1}$, for every i; the reserved name is n_0.

The requirement that $\varphi_{[\![\,]\!]}$ 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 $\varphi_{[\![\,]\!]}(a)$ 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 $\varphi_{[\![\,]\!]}(a)$ 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 $\varphi_{[\![\,]\!]}(u) \cap \varphi_{[\![\,]\!]}(v) \neq \emptyset$; since there is no relationship between different names, this implies that, for every w, $\varphi_{[\![\,]\!]}(u) \cap \varphi_{[\![\,]\!]}(w) \neq \emptyset$. 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 $\varphi'_{[\![\,]\!]}$ satisfying the requirement of Definition 1. Otherwise, for every v and w, $\varphi_{[\![\,]\!]}(v)$ and $\varphi_{[\![\,]\!]}(w)$ must have a different name in common with $\varphi_{[\![\,]\!]}(u)$; thus, $\varphi_{[\![\,]\!]}(u)$ would contain an infinite number of names.

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 fuses two (or more) different names. 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 force name fusions. 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 to us a very reasonable requirement. Notice that our definition of name invariance is definitely more complex than those, e.g., of [9, 29, 32, 33], 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.

Up to now, we have presented and discussed properties dealing with the way in which an translation is defined; we are still left with the more crucial part of the criteria. We want to focus our attention on the computation capabilities of the languages (i.e., what the languages can calculate); 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 have considered it to have a set of criteria that works well both for encodability and for separation results. Nothing related to this property has ever been assumed for separation results, except in [15, 16] where, however, it was formulated in a too demanding way. Also notice that the original formulation of operational correspondence put forward in [28] 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 junks.

Another important semantic issue, borrowed from [9, 12, 19, 26], 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 $\mathcal{L}_1$ in $\mathcal{L}_2$ 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 [19]).

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 **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 [25] or by the set of *tests* that a process successfully passes [13, 34]. Barbs are often defined in a very ad hoc way, are chosen as the simplest predicates that induce meaningful congruences and strictly depend on their language (even though in

[34] 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 instead very natural in, e.g., [9, 17, 29] 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 the success term $\surd$), 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 latters can be identified via elementary tests and test passing is the basic mechanism for the 'canonical' definition of barbs in [34].

By following [3, 7, 8], 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 [17, 29], 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. 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 $[\![S]\!] \Downarrow$.

3 Proving Known Separation Results

The properties we have just presented are met by most of the best known encodings appearing in the literature (e.g. the encoding of polyadic communications into monadic ones [24], of synchronous into asynchronous communications [5], and so on). Moreover, their combination yields a non-trivial proposal: the first encoding of the asynchronous π-calculus into Mobile Ambients that satisfies all such criteria is in [14]. We now show that their combination allows us to prove in a simpler and more uniform way the best known separation results appearing in the literature.

For example, let us start with the separation results in [17]. There, they work by assuming (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 debatable. 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 1. *There exists no encoding of π_a in CCS.*

Proof. By contradiction. Let a, b, c and d be pairwise distinct names and define $P \triangleq [x = b][c = c][d = d]\surd$. Property 3 implies that $\llbracket (a(x).P \,|\, \mathbf{0}) \mid \overline{a}\langle b\rangle \rrbracket$ reduces to a process equivalent to $\llbracket \surd \rrbracket$ that, by Property 5, reports success. Let $C_{|}^{\{a,b,c,d\}}[\_{1}\ ;\ \_{2}]$ be $(\nu\widetilde{n})(\_{1} \mid \_{2} \mid R)$; then, $\llbracket a(x).P \,|\, \mathbf{0} \rrbracket \stackrel{\rho}{\Rightarrow} K$ and $\llbracket \overline{a}\langle b\rangle \rrbracket \,|\, R \stackrel{\overline{\rho}}{\Rightarrow} K'$, for $(\nu\widetilde{n})(K \mid K') \asymp_2 \llbracket \surd \rrbracket$. In particular, $\rho \triangleq \mu_1 \cdot \ldots \cdot \mu_k$ and $\overline{\rho} \triangleq \overline{\mu}_1 \cdot \ldots \cdot \overline{\mu}_k$, for $\mu_i \in \{m_i, \bar{m}_i\}$ and $\{m_1, \ldots, m_k\} \cap \widetilde{n} = \emptyset$ (indeed, $\llbracket a(x).P \,|\, \mathbf{0} \rrbracket = (\nu\widetilde{n})(\llbracket a(x).P \rrbracket \mid \llbracket \mathbf{0} \rrbracket \mid R)$ and $\llbracket a(x).P \,|\, \mathbf{0} \rrbracket \stackrel{\rho}{\Rightarrow}$ imply that the names in ρ do not belong to $\widetilde{n}$). Let σ be the permutation that swaps a with c and b with d. By Property 2, $\llbracket c(x).P\sigma \,|\, \mathbf{0} \rrbracket \stackrel{\rho'}{\Longrightarrow} K\sigma'$ and $\llbracket \overline{c}\langle d\rangle \rrbracket \,|\, R \stackrel{\overline{\rho'}}{\Longrightarrow} K'\sigma'$, for $(\nu\widetilde{n})(K\sigma' \mid K'\sigma') \asymp_2 \llbracket \surd \rrbracket$ and $\rho' = \rho\sigma'$; here σ' denotes the permutation of names induced by σ, as defined in Property 2. More precisely, $\rho' \triangleq \mu'_1 \cdot \ldots \cdot \mu'_k$ and $\overline{\rho'} \triangleq \overline{\mu'_1} \cdot \ldots \cdot \overline{\mu'_k}$, for $\mu'_i \triangleq \mu_i\sigma'$ and $\{\sigma'(m_1), \ldots, \sigma'(m_k)\} \cap \widetilde{n} = \emptyset$.

Now, consider $Q \triangleq ((a(x).P \,|\, \mathbf{0}) \mid \overline{a}\langle d\rangle) \mid ((c(x).P\sigma \,|\, \mathbf{0}) \mid \overline{c}\langle b\rangle)$. Trivially, $Q \not\Downarrow$ whereas, as we shall see, $\llbracket Q \rrbracket \Downarrow$; this yields the desired contradiction. By compositionality, $\llbracket Q \rrbracket \triangleq (\nu\widetilde{n})((\nu\widetilde{n})(\llbracket a(x).P \,|\, \mathbf{0} \rrbracket \,|\, \llbracket \overline{a}\langle d\rangle \rrbracket \,|\, R) \mid (\nu\widetilde{n})(\llbracket c(x).P\sigma \,|\, \mathbf{0} \rrbracket \,|\, (\llbracket \overline{c}\langle b\rangle \rrbracket \,|\, R) \mid R)$. Then, consider $\llbracket Q \rrbracket \Longmapsto (\nu\widetilde{n})((\nu\widetilde{n})(K \mid K') \mid (\nu\widetilde{n})(K\sigma' \mid K'\sigma') \mid R) \asymp_2 (\nu\widetilde{n})(\llbracket \surd \rrbracket \mid \llbracket \surd \rrbracket \mid R)$, obtained by synchronizing

- μ_i produced by $\llbracket a(x).P \rrbracket$ with $\overline{\mu_i}$ produced by $\llbracket \overline{a}\langle d\rangle \rrbracket \mid R$, if $m_i \notin \varphi_{\llbracket\,\rrbracket}(b)$;
- μ_i produced by $\llbracket a(x).P \rrbracket$ with $\overline{\mu'_i}$ produced by $\llbracket \overline{c}\langle b\rangle \rrbracket \mid R$, if $m_i \in \varphi_{\llbracket\,\rrbracket}(b)$;
- μ'_i produced by $\llbracket c(x).P\sigma \rrbracket$ with $\overline{\mu'_i}$ produced by $\llbracket \overline{c}\langle b\rangle \rrbracket \mid R$, if $m_i \notin \varphi_{\llbracket\,\rrbracket}(b)$;
- μ'_i produced by $\llbracket c(x).P\sigma \rrbracket$ with $\overline{\mu_i}$ produced by $\llbracket \overline{a}\langle d\rangle \rrbracket \mid R$, if $m_i \in \varphi_{\llbracket\,\rrbracket}(b)$. □

Theorem 2. *There exists no encoding of MA in CCS.*

Proof. The previous proof can be adapted to MA: indeed, in [14] we provide an encoding of channel based communications of π_a in MA and in [32] it is shown how to encode name matching in MA. 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. □

We now aim at proving other separation results, viz. those in [9, 29, 32, 33], in a more uniform and abstract setting. 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.

3.1 First Setting

Let us assume that $\asymp_2$ is *exact*, i.e. $T \asymp_2 T'$ and $T \stackrel{\mu}{\longrightarrow}$ imply that $T' \stackrel{\mu}{\Rightarrow}$, whenever $\mu \neq \tau$. Notice that 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 if $C^N_{|}[\_{1};\_{2}]$, 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; thus, similarly to [9, 17, 29, 30, 32, 33], we are now working with encodings that translate '|' homomorphically.

Theorem 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, then there cannot exist any encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ that translates '|' homomorphically.*

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 $[\![S]\!] \xrightarrow{\bar{\mu}}$, for some pair of complementary actions μ and $\bar{\mu}$ (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 $(\!| S |\!) \xrightarrow{\bar{\mu}}$; thus, $(\!| S |\!) \mid (\!| S |\!) \longmapsto_2$ whereas, by $S \not\longmapsto_1$ and property A, $(\!| S |\!) \not\longmapsto_2$, in contradiction with the hypothesis. $\square$

Corollary 1. *There exist no encoding of π_{mix}, CCS and MA in π_{sep} that translates '|' homomorphically.*

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, $a(x).\mathbf{0} + \bar{a}\langle b\rangle.\surd$ in π_{mix} and $(\nu p)(open\_p.\surd \mid n[in\_n.p[out\_n.out\_n.\mathbf{0}]])$ in MA. □

3.2 Second Setting

We now consider a second setting where $\asymp_2$, the semantic theory of the target language, is *reduction sensitive*; this means that $T \asymp_2 T'$ and $T' \longmapsto$ imply that $T \longmapsto$. Examples of reduction sensitive equivalence/preorders are strong synchronous/asynchronous bisimulation [1, 22] and the expansion preorder [2].

Working under this assumption has the advantage that we are able to carry out proofs also under translations of '|' more liberal than the homomorphic one. As already said, the fact that parallel composition must be translated homomorphically can be criticized and some authors [26] advocate a more liberal formulation, that we now consider. In particular, for every N, we let $C^N_{|}[\_{}_1; \_{}_2] = (\nu\widetilde{n})(\_{}_1 \mid \_{}_2 \mid R)$, for some process R and restricted names $\widetilde{n}$ that only depend on N. We would like to remark that an unconstrained form of compositionality (where nothing is said on $C^N_{|}[\_{}_1; \_{}_2]$) would not change the validity of the results we obtain; it would just force us to prove Theorems 4 and 5 is specific cases and not in a general setting, as now they are.

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 [15] and corresponding to property A in the proof of Theorem 3).

Proposition 1. *If $\asymp_2$ is reduction sensitive and $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ is an 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, $[\![S]\!] \longmapsto_2 T \longmapsto_2 T'' \longmapsto_2 \ldots$, in contradiction with Property 4 (since $S \not\longmapsto_1$ implies that S does not diverge). □

Another crucial consequence of our criteria is the following proposition.

Proposition 2. *Let $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ be an 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 $S_i \not\longmapsto$ for $i = 1, 2$, then $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto$.*

Proof. In this proof, let us assume the following notation: $block(S)$ denotes any term S' such that $\textsc{Fn}(S') = \textsc{Fn}(S)$, $S' \not\longmapsto_1$ and $S' \asymp_1 \mathbf{0}$. It is easy to build such a S': it suffices to prefix S with any blocking action involving a fresh restricted name.

By Properties 1 and 5, $[\![S_1 \mid S_2]\!] = (\nu\widetilde{n})([\![S_1]\!] \mid [\![S_2]\!] \mid R) \Downarrow$. 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 either because $[\![S_1]\!] \mid R \longmapsto_2$, or because $[\![S_2]\!] \mid R \longmapsto_2$, or because $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto_2$. The first two possibilities are impossible, because otherwise $[\![S_1 \mid block(S_2)]\!] \longmapsto_2$ or $[\![block(S_1) \mid S_2]\!] \longmapsto_2$ and this would violate Proposition 1: $S_1 \mid block(S_2) \not\longmapsto$ because $S_1 \not\longmapsto_1$, $block(S_2) \not\longmapsto_1$ and $block(S_2) \asymp_1 \mathbf{0}$, and similarly for $block(S_1) \mid S_2$. □

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 an 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 2; by such a result, $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto$;
(c) from S_2 obtain a process S_2' such that $S_1 \mid S_2' \not\longmapsto$ but $[\![S_1]\!] \mid [\![S_2']\!] \longmapsto$;
(d) by Property 1, this implies that $[\![S_1 \mid S_2']\!] \longmapsto$, in contradiction with Proposition 1.

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 in $\mathcal{L}_2$. This is different from [9, 17, 29, 32, 33] where, instead, a lot of efforts 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 3 by changing the hypothesis on $\asymp_2$; this modification will allow us to obtain Corollary 1 under a different choice of semantic theories for π_{sep}.

Theorem 4. *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 $T \mid T \not\longmapsto_2$. Also assume that $\asymp_2$ is reduction sensitive. Then, there cannot exist any encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.*

Proof. By contradiction. Let S be such that $S \not\longmapsto_1$, $S \not\Downarrow$ and $S \mid S \Downarrow$; by Proposition 2, $[\![S]\!] \mid [\![S]\!] \longmapsto_2$ that, by hypothesis, implies that $[\![S]\!] \longmapsto_2$, in contradiction with Proposition 1. □

We now give a second proof-technique that allows us to obtain the hierarchy for polyadic synchronizations in [9] and to adapt the results in [15, 16] 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 greatest number of names that must be matched to yield a reduction in $\mathcal{L}$. For example, the matching degree of CCS [22], of the π-calculus [22] and of Mobile Ambients [11] is 1; the matching degree of Dπ [18] is 2; the matching degree of π^n (the π-calculus with n-ary polyadic synchronizations [9]) is n; the matching degree of e^π (the π-calculus with arbitrary polyadic synchronizations [9]) 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).[1]

Theorem 5. *If* $\textsc{Md}(\mathcal{L}_1) > \textsc{Md}(\mathcal{L}_2)$*, then there exists no encoding* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$*.*

Proof. By contradiction. Pick two $\mathcal{L}_1$-processes S_1 and S_2 that satisfy the hypothesis of Proposition 2 and that synchronize only once (before reporting success) by matching exactly $k = \textsc{Md}(\mathcal{L}_1)$ names, viz. $\{n_1, \ldots, n_k\}$. By Proposition 2, their encodings must synchronize: i.e., $[\![S_1]\!] \xrightarrow{\mu}$ and $[\![S_2]\!] \xrightarrow{\bar{\mu}}$. Since $\textsc{Md}(\mathcal{L}_1) > \textsc{Md}(\mathcal{L}_2)$, it must be that the names in $\textsc{Fn}(\mu) \cap \textsc{Fn}(\bar{\mu})$ matched when synchronizing μ and $\bar{\mu}$ (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 fresh m (i.e., $m \notin \textsc{Fn}(S_1) \cup \textsc{Fn}(S_2) \cup \textsc{Fn}(\mu) \cup \textsc{Fn}(\bar{\mu})$) and consider the substitution σ that swaps m and n_i. Trivially, $S_1 \mid S_2\sigma \not\longmapsto_1$, whereas their encodings do synchronize (in contradiction with Proposition 1): by Property 2, $[\![S_2\sigma]\!] = [\![S_2]\!]\sigma' \xrightarrow{\bar{\mu}\sigma'}$, with $\bar{\mu}\sigma'$ that is still synchronizable with μ because σ' swaps component-wise $\varphi_{[\![\,]\!]}(n_i)$ and $\varphi_{[\![\,]\!]}(m)$ (and so it does not touch $\{m_1, \ldots, m_h\}$). $\square$

Corollary 2. *There exists no encoding from* e^π *into* π^m*, for every* m*, and from* π^m *into* π^n*, whenever* $m > n$*.*

Proof. Observe that $\textsc{Md}(e^\pi) = \infty$ and that $\textsc{Md}(\pi^m) = m$; then apply Theorem 5. $\square$

Proving New Separation Results and Building Hierarchies of Languages. We have just shown that our approach is more ‘usable’ than previous approaches to separation results, since it can be used to prove known results in a simpler and more uniform way. However, it also has the advantage of allowing the proof of new separation results: in [14], we exploit such criteria to compare the relative expressive power of several calculi for mobility (viz., the asynchronous π-calculus, a distributed π-calculus, a distributed version of Linda, and Mobile/Safe/Boxed Ambients together with several of their variants); moreover, the results in [15, 16] can be easily re-formulated under Properties 1–5. Finally, least but not last, the fact that our criteria are also well-suited for encodability results allows us to build hierarchies of languages in a uniform way.

3.3 Third Setting

The setting presented in Section 3.2 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 [26, 28], where weak (asynchronous) bisimilarity [1] is exploited. We now sketch a weaker setting, that covers all the separation results we are aware of (including [26, 28]) without breaking the elegant and powerful proof-techniques developed in Section 3.2.

[1] Incidentally, the early-style LTS for the π-calculus also verifies that $\xrightarrow{\bar{a}b}$ 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.

	Electoral Systems	Matching Systems	Our Criteria		
			1st setting	2nd setting	3rd setting
$CCS \not\to \pi_{sep}$	[29] (a)	$\times$	✓	✓	✓
$\pi_{mix} \not\to \pi_{sep}$	[29] (a)	$\times$	✓	✓	✓
MA $\not\to \pi_{sep}$	[32] (a)	$\times$	✓	✓	✓
$e^{\pi} \not\to \pi^{m} \not\to \pi^{n}$ $(m > n)$	$\times$	[9] (c)	?	✓	✓
MA $\not\to$ CCS	[33] (b)	[17] (a)	✓		
$\pi_a \not\to$ CCS	[29] (b)	[17] (a)	✓		

Table 1. Comparison between different separation methodologies. For every result, we list where it appears ('$\times$' if it has never been published and '?' if we believe that it holds but we have not been able to prove it) 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.

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 $T \Longmapsto_2 (\nu\widetilde{n})([\![\, S' \,]\!] \mid T') \asymp_2 [\![\, S' \,]\!]$.

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. Restricting $\asymp_2$ to pairs of kind $((\nu\widetilde{n})(T \mid T'), T)$, for $(\nu\widetilde{n})(T \mid T') \asymp_2 T$, yields a reduction sensitive relation, for any $\asymp_2$; thus, Propositions 1 and 2 (and, consequently, all the results proved in Section 3.2) hold also in this setting without requiring reduction sensitiveness of $\asymp_2$.

4 Conclusion

We have presented some criteria that an encoding should satisfy to be considered a good 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 of the best known 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 sample is given by the encodings of the π-calculus in Mobile Ambients [10, 11].

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 formulated and proved (in an easier and more uniform way) in terms of our criteria. In Table 1 we have comparatively listed such results. Roughly speaking, the approach taken in [9, 17, 29, 32, 33] 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 demanding than those in [9, 17, 29, 32, 33]. Concerning point (*i*), we are only aware of two kinds of problem: *symmetric electoral systems* [29, 32, 33] and *matching systems* [9, 17]. However, none of them is 'universal', in the sense that different separation results usually require different separation problems (see the '$\times$' in Table 1).

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 because, in that setting, such results are very strong, being the formulation of our criteria more liberal and abstract.

An orthogonal research line could be the study of enhanced kinds of translation. For example, it may happen to have a 'two-level' encoding [4, 6] where $[\![\cdot]\!]$ is a translation that satisfies Properties 2–5 and is such that $[\![P]\!] \triangleq C^{\mathrm{FN}(P)}[(\!| P |\!)]$, where $(\!| \cdot |\!)$ is a compositional translation (this property is called *weak* compositionality in [31]). The proof-techniques presented in Sections 3.2 and 3.3 can be readily adapted to this enhanced notion of encoding, whereas the proof-technique of Section 3.1 cannot (recall that there we had to work with homomorphic translations of parallel composition). Another possibility [21, 23] is to define an encoding as a family of translations $[\![\cdot]\!]_{\Xi}$ indexed with a set or a sequence of names Ξ (representing, e.g., an upper bound on the free names of the translated process or some auxiliary parameters for the translation). In this case, our framework is less adequate: it is difficult to adapt 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 $[\![\cdot]\!]_{\Xi}$? Are Ξ names in the source or in the target language? The latter question is very delicate: in the first case, Property 2 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 $([\![S]\!]_{\Xi})\sigma'$. Thus, even if we believe that such an enhanced form of encoding is reasonable, we have problems in adapting our framework without specifying anything on the index.

To conclude, the challenge raised in [27] is still open, but we think and hope that our proposal can contribute to its final solution.

Acknowledgments: I am grateful to Daniele Varacca, Jesus Aranda, Frank Valencia and Cosimo Laneve for several comments that improved an earlier draft of this work.

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 Pi-Calculus with Data Terms. in *Proc. of ICALP*, volume 3580 of *LNCS*, pages 1202–1213. Springer, 2005.
5. G. Boudol. Asynchrony and the π-calculus (note). Rapp. de Recherche 1702, INRIA 1992.
6. M. Bugliesi, M. Giunti. Secure implementations of typed channel abstractions. In *Proc. of POPL*, pages 251–262. ACM, 2007.
7. D. Cacciagrano, F. Corradini, J. Aranda, F. Valencia. Persistence and Testing Semantics in the Asynchronous π-calculus. In *Proc. of EXPRESS*, ENTCS 194(2): 59–84, 2007.
8. D. Cacciagrano, F. Corradini, C. Palamidessi. Separation of Synchronous and Asynchronous Communication Via Testing. *Theoretical Computer Science*, 386(3): 218–235, 2007.
9. M. Carbone and S. Maffeis. On the expressive power of polyadic synchronisation in pi-calculus. *Nordic Journal of Computing*, 10(2):70–98, 2003.
10. L. Cardelli, G. Ghelli, and A. D. Gordon. Types for the ambient calculus. *Information and Computation*, 177(2):160–194, 2002.
11. L. Cardelli and A. Gordon. Mobile ambients. *Theor. Comp. Science*, 240(1):177–213, 2000.
12. F. de Boer and C. Palamidessi. Embedding as a tool for language comparison. *Information and Computation*, 108(1):128–157, 1994.
13. R. De Nicola and M. Hennessy. Testing equivalence for processes. *TCS*, 34:83–133, 1984.
14. D. Gorla. Comparing calculi for mobility via their relative expressive power. Technical Report 09/2006, Dipartimento di Informatica, Università di Roma "La Sapienza".
15. 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.
16. D. Gorla. Synchrony vs asynchrony in communication primitives. In *Proc. of EXPRESS'06*, volume 175 of *ENTCS*, pages 87–108. Elsevier, 2007.
17. B. Haagensen, S. Maffeis, and I. Phillips. Matching systems for concurrent calculi. In *Proc. of EXPRESS'07*, ENTCS 194(2):85–99, 2007.
18. M. Hennessy and J. Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation*, 173:82–120, 2002.
19. M. Herlihy. Wait-free synchronization. *ACM ToPLaS*, 13(1):124–149, 1991.
20. K. Honda and N. Yoshida. On reduction-based process semantics. *TCS*, 152:437-486, 1995.
21. F. Levi. A Typed Encoding of Boxed into Safe Ambients. *Acta Inform.*, 42(6):429-500, 2006.
22. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
23. R. Milner. Functions as Processes. *Mathem. Struct. in Comp. Science*, 2(2):119–141, 1992.
24. R. Milner. The polyadic π-calculus: A tutorial. In *Logic and Algebra of Specification*, volume 94 of *Series F*. NATO ASI, Springer, 1993.
25. R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proc. of ICALP '92*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.
26. U. Nestmann. What is a 'good' encoding of guarded choice? *Inf. Comp.*, 156:287-319, 2000.
27. 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.
28. U. Nestmann and B. C. Pierce. Decoding choice encodings. *Inf. and Comp.*, 163:1–59, 2000.
29. C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

30. C. Palamidessi, V. Saraswat, F. Valencia and B. Victor. On the Expressiveness of Linearity vs Persistence in the Asynchronous π-calculus. In *Proc. of LICS*, pages 59–68. IEEE, 2006.
31. J. Parrow. Expressiveness of Process Algebras. Proc. of *Emerging trends in Concurrency Theory*, ENTCS 209:173–186. Elsevier, 2008.
32. I. Phillips and M. Vigliotti. Electoral systems in ambient calculi. In *Proc. of FoSSaCS*, volume 2987 of *LNCS*, pages 408–422. Springer, 2004.
33. I. Phillips and M. Vigliotti. Leader election in rings of ambient processes. *Theoretical Computer Science*, 356(3):468–494, 2006.
34. J. Rathke, V. Sassone and P. Sobocinski. Semantic Barbs and Biorthogonality. In *Proc. of FoSSaCS*, volume 4423 of *LNCS*, pages 302–316. Springer, 2007.
35. D. Sangiorgi and D. Walker. The π-calculus: a Theory of Mobile Processes. C.U.P., 2001.

Syntax, Operational and Behavioural Semantics of the Calculi Considered

We now very briefly present the syntax and the operational semantics of the languages considered; for more details, the interested reader can refer to [5, 9, 11, 22]. All the languages have a common syntax given by

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

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 to 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.

Terms of this syntax are equated up-to structural congruence, that is the least congruence closed under alpha-renaming of bound names and under the following axioms:

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

$$!P \equiv P \mid !P \qquad (\nu n)\mathbf{0} \equiv \mathbf{0} \qquad (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$$

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

where $\textsc{Fn}(P)$ denotes the *free names* (i.e., the names not bound) in P. The inference rules that define the operational semantics of processes are:

$$\frac{P \longmapsto P'}{P \mid Q \longmapsto P' \mid Q} \qquad \frac{P \longmapsto P'}{(\nu n)P \longmapsto (\nu n)P'} \qquad \frac{P \equiv P' \quad P' \longmapsto Q' \quad Q' \equiv Q}{P \longmapsto Q}$$

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

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

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

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:

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

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

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

Here, $\overline{a}\langle b\rangle$ denotes the emission of name b along channel a; $a(x).P$ in an input prefixed process that waits 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:

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

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:

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

and it includes 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:

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

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, 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:

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

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

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

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

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:

$$P ::= \ldots \;\Big|\; a[P] \;\Big|\; M.P \;\Big|\; \langle M \rangle \;\Big|\; (x).P$$
$$M ::= n \;\Big|\; in\_a \;\Big|\; out\_a \;\Big|\; open\_a \;\Big|\; M.M$$

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:

$$n[in\_m.P_1|P_2] \;\mid\; m[P_3] \longmapsto m[P_3 \mid n[P_1|P_2]]$$
$$m[n[out\_m.P_1|P_2] \mid P_3] \longmapsto n[P_1|P_2] \;\mid\; m[P_3]$$
$$open\_n.P_1 \;\mid\; n[P_2] \longmapsto P_1 \mid P_2$$
$$\langle M \rangle \;\mid\; (x).P \longmapsto P\{^M\!/_x\}$$

and the new reduction rule:
$$\frac{P \longmapsto P'}{n[P] \longmapsto n[P']}$$

Moreover, structural congruence also includes the following axioms:

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

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 [10].