A Taxonomy of Process Calculi for Distribution and Mobility

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

Abstract

In this paper, we comparatively analyze some mainstream calculi for mobility and distribution, together with some of their variants: asynchronous π-calculus, distributed π-calculus, and some dialects of Mobile/Boxed/Safe ambients. In particular, we focus on their relative expressive power, i.e. we try to encode every language in the other while respecting some reasonable properties. According to the possibility or the impossibility for such results, we set up a taxonomy of these languages. Our study enables understanding, for every pair of calculi, which features of one can be rendered in the other and how this is possible, or which features cannot be rendered and why this is impossible.

Contents

1 Introduction

In the last years, one of the main research lines in the field of concurrency theory has been the development of new formalisms, paradigms and environments that better model distributed and mobile systems. These are systems whose configuration deeply varies in time, as a consequence of the interactions between the principals (usually called *processes*) they host. Several terms have been coined to name this research line (network-aware programming, WAN computing, global computing, ...) that is now a well-established field for many computer scientists around the world.

Calculi for mobility and/or distribution In this scenario, the term *mobility* has become the reference keyword to denote several possible dynamic evolutions of systems [12, 16]. The first language where mobility plays a crucial rôle is the *π-calculus* [40, 52], a calculus centered around the notion of *name mobility*. In π-calculus, a collection of concurrent processes communicate through named channels and the communicated objects are channel names as well. This is evident from the reduction rule

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

where a process P receives a name b from an output particle, after a synchronization on channel a. Thus, the dynamic modifications of a system consist in the variation of the interconnection structure underlying the processes as a result of inter-process communications. An evolution of the π-calculus is the *distributed π-calculus* [26, 28], where processes are located at network nodes and only co-located processes can communicate. In languages of this kind (featuring what is sometimes called *mobile computation* [12]), the net structure (seen as a collection of network nodes) is fixed and visible to the processes running in the system; processes can move across the net for communicating with remote processes, i.e. they can migrate from one node to another, as expressed by the two following sample reductions

$$l : go\_k.a(x).P \;\mid\; k : \bar{a}\langle b\rangle.R \longmapsto l : \mathbf{0} \;\mid\; k : (\bar{a}\langle b\rangle.R \,|\, a(x).P) \longmapsto l : \mathbf{0} \;\mid\; k : (R \,|\, P\{^{b}/_{x}\})$$

A more radical approach can be obtained by assuming that network nodes can move as a whole, i.e. together with the processes and data they host (this has been sometimes called *mobile computing* [12]). A typical example is the *Mobile Ambient calculus* [15]; ambients are collections of data and processes, and they can enter or exit other ambients, as expressed by the following two sample reductions:

$$n[in\_m.out\_m.P \mid Q \mid \langle b\rangle] \mid m[R] \longmapsto m[n[out\_m.P \mid Q \mid \langle b\rangle] \mid R] \longmapsto n[P \mid Q \mid \langle b\rangle] \mid m[R]$$

Different kinds of mobility stress different features of the system modeled; the static analysis of such features has longly been the primary research topic on calculi for mobility and distribution, yielding more and more sopisticated type theories ([7, 8, 13, 14, 28, 31, 33, 36, 49], just to cite a very few examples). More recently, calculi for mobility have also been the workbench of orthogonal research lines, like the development of efficient implementations of new programming paradigms [20, 21, 46, 50] and of easy-to-handle proof-techniques for proving behavioural properties of systems [8, 27, 33, 35, 37]. From the practical side, we would need real-life applications where the distinctive features of such formalisms are essential. From the theoretical side, one of the things that is still lacking is an exhaustive comparative analysis of all these proposals, from a linguistic perspective; in particular, there

is a plethora of results (either formal or informal) about the inter-relationships between the different languages and paradigms, but very few unified results are around.

In this paper, we approach this problem by comparing some mainstream calculi for mobility: asynchronous π-calculus (written π_a) [6, 30], distributed π-calculus (written Dπ) [26, 28], Mobile Ambients (written MA) [15], Safe Ambients (written SA) [33], together with its variant with passwords (written SAP) [35], and Boxed Ambients (written BA) [7], together with its variant with shared channels (written BA$_s$) [7], with co-actions (written SBA) [36] and with co-actions and passwords (written NBA) [8]. Our results formally prove some claims informally appeared in the literature and prove in a different way some formal results already known. Moreover, for the sake of systematization, we also consider and compare languages that, to the best of our knowledge, have never been contrasted, not even informally. Consequently, our results carry a two-fold contribution: on one hand, they help in better clarifying the peculiarities of the languages studied and their distinctive programming features; on the other hand, they allow us to formally compare the expressive power of the languages and organize them in a taxonomy based on their relative expressiveness.

First of all, let us briefly discuss the choice of the nine languages considered in this paper. The previous discussion on the three different kinds of mobility justifies the choice of π_a, Dπ and MA, since they are maybe the simplest and most representative samples of languages with name mobility, mobile computation and mobile computing, respectively. For example, several variations of the π-calculus featuring distribution and process mobility have appeared in the last fifteen years ([1, 19, 53], just to cite some samples); however, differently from Dπ [26, 28], none of them has become a reference model for a distributed π-calculus.

We then decided to include several variants of MA; this choice was driven by two reasons. First, in the last decade MA is the formalism for mobility and distribution that originated the highest number of (sometimes minor) variations; for the sake of completeness, we would like to consider the most representative ones. Second (and more important), for didactic reasons: we want to show that it may happen that even very small modifications in the syntax and/or in the operational semantics can produce a different (i.e. incomparable) formalism.

Let us briefly examine the six variations we are going to consider. First, we have SA, where every ambient activity has to be authorized by the target ambient. For example, in SA ambient n can enter into the sibling ambient m only if m authorizes such an entrance, via a proper *co-action*:

$$n[in\_m.P] \mid m[\overline{in}\_m.Q] \longmapsto m[n[P] \mid Q]$$

This is different from MA, where only the presence of an ambient named m suffices. To have a tighter control, SA has been evolved into SAP, where every movement is also associated to a password. For example, n can enter into m only if m authorizes the entrance and n provides the right password for entering:

$$n[in\_(m,p).P] \mid m[\overline{in}\_(m,p).Q] \longmapsto m[n[P] \mid Q]$$

An orthogonal way of modifying MA is related to the *open* primitive. In MA all communications are local; so, if a process P wants to access a datum contained into a sibling ambient m, it has to dissolve m and place the datum locally, in order to access it:

$$open\_m.(x).P \mid m[\langle b \rangle] \longmapsto (x).P \mid \langle b \rangle \longmapsto P\{^{b}/_{x}\}$$

Dissolving ambient boundaries is a sensible task; for this reason, BA was defined, by removing the *open* primitive and allowing a limited form of remote communication. For example, the previous interaction

can be alternatively rendered in BA as

$$(x)^m.P \mid m[\langle b\rangle^\star] \longmapsto P\{^b/_x\} \mid m[\mathbf{0}] \qquad \text{or} \qquad (x)^\star.P \mid m[\langle b\rangle^\uparrow] \longmapsto P\{^b/_x\} \mid m[\mathbf{0}]$$

Indeed, a local output (tagged with '$\star$') can interact either with a local input, or with an input from the parent (tagged with the name of the ambient towards which the input is directed, m in our example) or with an input from a son (tagged with '$\uparrow$'). This choice can create a lot of conflicts between processes competing for the same datum. For this reason, it has been defined BA_s, where an input/output tagged with m can only synchronize with an output/input tagged with '$\uparrow$'. Hence, in BA_s the previous remote communications would take a unique form:

$$(x)^m.P \mid m[\langle b\rangle^\uparrow] \longmapsto P\{^b/_x\} \mid m[\mathbf{0}]$$

Finally, BA and BA_s have been developed further to allow a finer control on ambient movements, by following paths similar to those put forward by SA and SAP. On one hand, SBA adds co-actions to BA; for example, an ambient n can enter into m if m allows the entrance of n or of any ambient (this feature is also an enhancement of SA's co-actions):

$$n[in\_m.P] \mid m[\overline{in}\_\delta.Q] \longmapsto m[n[P] \mid Q] \qquad \text{for } \delta \in \{n, *\}$$

On the other hand, NBA adds to BA_s co-actions, passwords and the possibility of dynamically learning the name of the entering ambient (the latter feature is an enhancement of SAP too):

$$n[in\_(m, p).P] \mid m[\overline{in}\_(x, p).Q] \longmapsto m[n[P] \mid Q\{^n/_x\}]$$

Relative expressiveness and Valid encodings Of course, it is crucial to fix the criteria to evaluate the expressive power of the languages considered. Too liberal criteria would lead us to poorly informative results: most (if not all) of the languages would satisfy them. But also too stringent criteria would be fruitless: (almost) none of the languages would satisfy it. A good compromise seems to be the notion of *relative expressiveness*: this approach relies on the possibility/impossibility of translating one language into another, while respecting some reasonable properties. Again, the definition of such properties is essential for the meaningfulness of our study.

In principle, a good encoding function should satisfy at least two properties: *compositionality* (roughly, this is a way to require that the encoding is defined inductively on the syntax of the encoded term) and *faithfulness* (the encoding of a term must have the same functionalities as the original term, without introducing new ones). There are different ways to formalize these notions; mainly for the second one, a number of different proposals have been considered in the literature (e.g., sensitiveness to barbs/divergence/deadlock, operational correspondence, full abstraction, ...). Here, we consider the proposal of [25] and consider *valid* only the encodings that satisfy the following five properties:

- *compositionality*: the encoding of a compound term must be expressed by combining the encoding of its components via a translating context that only depends on the top-level operator that is translated and on the free names of the term itself;
- *name invariance*: the encodings of two source processes that differ only in their free names must only differ in the associated free names;

- *operational correspondence*: computations of the source term must correspond to computations in the encoded term, and vice versa;
- *divergence sensitiveness*: non-terminating processes must be translated into non-terminating processes, and vice versa;
- *success sensitiveness*: successful terms (for some notion of success) must be translated into successful terms, and vice versa.

We think that these criteria form a valid proposal for language comparison; indeed, several well known encodings respect them (so our notion is consistent with the common understanding of the community), but there still exist encodings in the literature that do not satisfy them (so our notion is non-trivial). Here, we furthermore vindicate the validity of our proposal by showing that some widely believed (but never formally proved) separation results can be established by relying on the above mentioned criteria.

Of course, several alternatives are possible when formulating the criteria. We shall now discuss a couple of representative samples.

Compositionality requires that the encoding of $P|Q$, that denotes the parallel composition of processes P and Q, is $[\![P|Q]\!] \triangleq C_{|}^{fn(P,Q)}([\![P]\!] ; [\![Q]\!])$, where $C_{|}^{fn(P,Q)}(\__1 ; \__2)$ is the context used to translate the parallel composition of two processes that have as free names the union of the free names of P and Q. This formulation can be both weakened and strenghtened. A weakening is to require that the encoding is 'two-level' (in [45] this is called *weak compositionality*): $[\![P|Q]\!] \triangleq C((\!| P|Q |\!))$, where $C(\cdot)$ is some context and the second-level encoding function $(\!| \cdot |\!)$ is compositional (in the way defined above). Compositionality can also be strenghtened (see, e.g., [11, 44, 47, 48]), by imposing that the parallel operator is translated homomorphically: $[\![P|Q]\!] \triangleq [\![P]\!] \mid [\![Q]\!]$. All these formulations have their own merits and can be exploited in practice. We decided to work with the first one because it seems us a good compromise between generality and usability. Of course, all our results also hold when assuming homomorphism and most of them still hold also under weak compositionality.

As a second example, operational correspondence requires that every source reduction $P \longmapsto P'$ must be preserved by the encoding (this property is sometimes called *operational completeness*, see [42]), i.e. that $[\![P]\!]$ reduces (maybe, in several reduction steps) to $[\![P']\!]$ in the target language. However, it is too demanding to formulate this intuition by requiring that $[\![P]\!] \Longmapsto [\![P']\!]$: in general, the encoding leaves some junk dead processes around. So, a more liberal formulation of operational completeness is $[\![P]\!] \Longmapsto\!\!\simeq [\![P']\!]$, for some behavioural equivalence '$\simeq$' in the target language (that gets rid of the junk processes left around). Of course, a finer equivalence entails a stronger property of the encoding. In this paper we shall work with *strong berbed equivalence*, that is usually considered the strongest 'reasonable' equivalence for process calculi. Of course, this makes our encodability results very strong; furthermore, our impossibility results are not undermined by this choice, since we believe that they hold under any 'reasonable' notion of behavioural equivalence.

Contributions A full account of our study is given in Table 1 in Section 4.5; here, in Figure 1, we just report a pictorial summary of our taxonomy. Notationally, we write $\mathcal{L}_1 \longrightarrow \mathcal{L}_2$ if $\mathcal{L}_1$ can be encoded in $\mathcal{L}_2$ but not vice versa, and absence of an arrow means that no valid encoding exists from one to the other. The dashed arrows mean that an encodability result holds, but its properties are not as strong as the other arrows: in particular, they are formulated up to a coarser notion of process equivalence (namely, strong *translated* barbed equivalence) or by relying on weak compositionality.

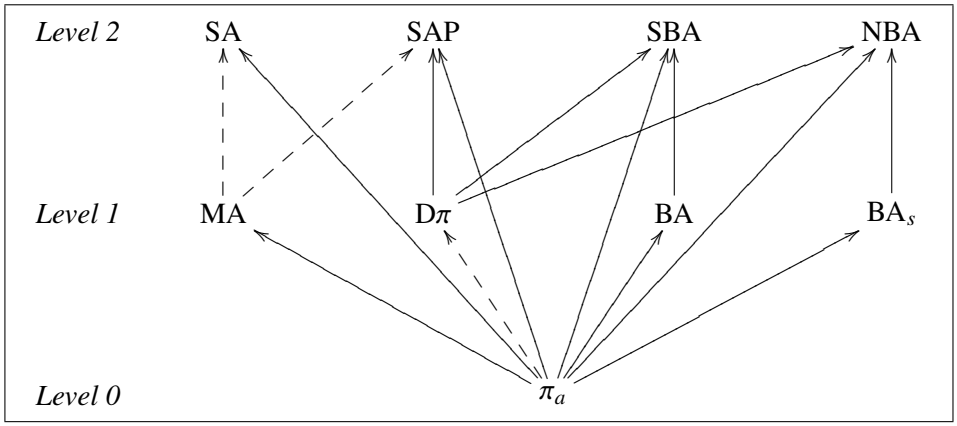

Figure 1: The Taxonomy of Calculi for Distribution and Mobility

With respect to the extended abstracts [23, 24], this paper fully builds up a unique taxonomy and it formally proves new results (either conjectured or not considered at all in the two preliminary versions). In detail, most of the separation results (i.e., the arrows that are not present in the figure) have already been mentioned in the extended abstracts but here we give full details on complex proofs just sketched in [23, 24]. It is worth noting that a very few of them (i.e., the non encodability of MA, SA and BA in π_a and Dπ) have already appeared in the literature [47, 48] but under a different set of encoding properties. Here, we also refine some results; for example, here we can prove the incomparability of BA and BA$_s$ in the setting of [25], i.e. by getting rid of an extra property assumed in [23]. Also some of the encodability results are new, or are considered in a new way in this paper. More precisely:

- the encodability of π_a in MA has already been presented in [24], but here we present a simpler and more elegant encoding;
- the encodability of π_a in SA (and in SAP) comes from [33];
- the encodability of π_a in Dπ has never been formally discussed, to the best of our knowledge; even if it may look trivial, it raises some interesting issues and, in detail, we show that only a weak compositional valid encoding exists (this fact justifies the dashed arrow);
- the encodability of π_a in BA (and in BA$_s$, SBA and NBA) comes from [7];
- the encodability of MA in SA (and in SAP) comes from [33], but here we spell out the precise properties enjoyed by the encoding proposed in *loc.cit.*, in particular that operational correspondence only holds under translated barbed equivalence (this fact justifies the dashed arrow);
- the encodability of Dπ in SAP, SBA and NBA are a totally new contribution of this paper (they not even appeared in the extended abtracts);
- the encodability of BA in SBA and of BA$_s$ in NBA are straightforward but, to the best of our knowledge, they have never been spelled out formally.

Some of our results are expected: for example, we confirm that π_a is the minimal common denominator of calculi for mobility, since it can be encoded in all the languages considered. Some other results, though expected, turned out very difficult to prove. For example, the task of encoding π_a in MA

is quite challenging with our formulation of operational correspondence. Indeed, ruling out target computations that are not present in the source process is a sensible task when dealing with MA, because of the high possibility of interferences between MA processes. A simpler encoding of π_a is possible, e.g., in SA (see [33]), because the latter language is "more controlled" than MA. Another issue that turned out to be surprisingly difficult to understand is the relative expressiveness of MA and BA. As a new contribution w.r.t. [24], here we formally prove a separation result between MA and BA, only conjectured in *loc.cit.*. Moreover, we also discovered that very few dialects of MA are related to it. This entails that, in many cases, the dialect is not an enhancement of the original language nor a minor variation on it, as it is sometimes believed. Indeed, the distinguishing features added to (or modified in) the original language can have advantages (e.g., in terms of ease-of-programming or of controlling interferences) that make the dialect non-encodable in the original language; the price to be paid is that some computational features of the original language get lost, thus making also the converse encoding impossible.

This paper is organized as follows. In Section 2, we formally present the syntax and operational semantics of the nine languages depicted in Figure 1. In Section 3, we recall from [25] the properties that a *valid* encoding should satisfy. In Section 4, we formally build up the taxonomy of Figure 1: for every pair of languages, we give a formal proof of encodability/non-encodability; in conclusion, we also discuss some sufficient conditions that allow us to have that the composition of valid encodings is still a valid encoding. Finally, in Section 5 we conclude the paper by also mentioning some related work.

2 The Process Calculi

In what follows, we assume a countable set of names, $\mathcal{N}$, ranged over by $a, b, c, \ldots, l, k, \ldots, m, n, \ldots$, $u, v, w, \ldots, x, y, z, \ldots$ and their decorated versions. To simplify reading, we use: $a, b, c, \ldots$ to denote channels; $l, k, \ldots$ to denote localities; $m, n, \ldots$ to denote ambients; $x, y, z, \ldots$ to denote input variables; finally, $u, v, w, \ldots$ are used to denote generic names (channels and variables in π_a; channels, localities and variables in Dπ; ambients and variables in ambient-based calculi).

A *process calculus* is a triple $\mathcal{L} = (\mathcal{P}, \longmapsto, \simeq)$, where

- $\mathcal{P}$ is the set of language terms, usually called *processes* and ranged over by $P, Q, R, \ldots$. All the process calculi we are going to consider 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 in Section 3); $P_1|P_2$ denotes 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 infinite processes; all our results do not rely on this choice and can be rephrased under different forms of recursion.

- $\longmapsto$ 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*. These are inductively defined judgements whose inference rules shared by all our process calculi 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, $\mathcal{E}(P)$ denotes the process obtained by replacing the hole ' $\cdot$ ' with process P, and $\equiv$ denotes structural equivalence (used to equate different ways of writing the same process). Of course, the operational axioms, the evaluation contexts and the structural equivalence are peculiar to every language and will be defined in a few moments. As usual, $\Longmapsto$ denotes the reflexive and transitive closure of $\longmapsto$.

- $\simeq$ is a behavioural equivalence, needed to describe the abstract behaviour of a process; usually, $\simeq$ is a congruence with respect to closure under evaluation contexts (or, at the very least, with respect to parallel composition).

2.1 The asynchronous π-calculus (π_a)

We consider the asynchronous version of the π-calculus, as defined in [6]. This language is nowadays widely considered the minimal common denominator of calculi for mobility, it is a good compromise between expressiveness and simplicity, and it also has a running implementation [50]. Its syntax extends the common syntax of processes by letting

$$P \;::=\; \ldots \;\big|\; \bar{u}\langle v\rangle \;\big|\; u(x).P$$

Intuitively, $\bar{u}\langle v\rangle$ represents message v unleashed along channel u. Dually, $u(x).P$ waits for some message from channel u and, once received, replaces with such a message every occurrence of variable x in P. Processes $u(x).P$ and $(\nu a)P$ bind x and a in P, respectively; a name occurring in P that is not bound is called free. Consequently, we define the free and bound names of a process P, written $fn(P)$ and $bn(P)$; alpha-conversion is then defined accordingly. For the sake of notation, we shall write $fn(P, Q)$ to denote $fn(P) \cup fn(Q)$, and similarly for bound names.

Evaluation contexts are defined as follows:

$$\mathcal{E}(\cdot) \;::=\; \cdot \;\big|\; \mathcal{E}(\cdot)\,|\,P \;\big|\; P\,|\,\mathcal{E}(\cdot) \;\big|\; (\nu n)\mathcal{E}(\cdot)$$

The *structural equivalence* relation, $\equiv$, is the least equivalence on processes closed by evaluation contexts, including alpha-conversion and satisfying the following axioms:

$$P|\mathbf{0} \equiv P \qquad P_1|P_2 \equiv P_2|P_1 \qquad P_1|(P_2|P_3) \equiv (P_1|P_2)|P_3 \qquad !P \equiv P|!P$$
$$(\nu a)\mathbf{0} \equiv \mathbf{0} \qquad (\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad P_1|(\nu a)P_2 \equiv (\nu a)(P_1|P_2) \text{ if } a \notin fn(P_1)$$

The *reduction relation*, $\longmapsto$, is the least relation on processes closed by the inference rules previously described and satisfying the following axiom:

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

where $P\{^b/_x\}$ denotes the capture-avoiding substitution of each occurrence of x in P with an occurrence of b.

2.2 Distributed π-calculus (Dπ)

We present a slightly simplified version of [28]; mainly, we elide typing information from the syntax. The main syntactic entity is the set of *nets*, that are collections of *located processes*, possibly sharing restricted names:

$$N \;::=\; \mathbf{0} \;\big|\; l : P \;\big|\; N|N \;\big|\; (\nu u)N$$

Processes are obtained from the common syntax by letting

$$P \;::=\; \ldots \;\big|\; u(x).P \;\big|\; \bar{u}\langle v\rangle.P \;\big|\; go\_u.P$$

The main differences between Dπ and π_a are: processes and channels are located at a specified locality; communication can only happen between co-located processes and, hence, there is a primitive to let processes migrate between localities (viz. action go_u); finally, communication is synchronous (i.e., it blocks both the sending and the receiving process).

Since the main syntactic entity is the set of nets, evaluation contexts, reductions and structural equivalence will be given for nets.

$$\mathcal{E}(\cdot) \;::=\; \cdot \;\big|\; \mathcal{E}(\cdot) \,|\, N \;\big|\; N \,|\, \mathcal{E}(\cdot) \;\big|\; (\nu n)\mathcal{E}(\cdot)$$

The structural axioms are:

$$l : P|\mathbf{0} \equiv l : P \qquad l : P_1|P_2 \equiv l : P_1 \mid l : P_2 \qquad l : !P \equiv l : P|!P \qquad (\nu l)N \equiv (\nu l)(N \mid l : \mathbf{0}) \qquad N|\mathbf{0} \equiv N$$

$$N_1|N_2 \equiv N_2|N_1 \qquad N_1|(N_2|N_3) \equiv (N_1|N_2)|N_3 \qquad (\nu u)(\nu w)N \equiv (\nu w)(\nu u)N \qquad (\nu n)\mathbf{0} \equiv \mathbf{0}$$

$$l : (\nu u)P \equiv (\nu u)l : P \;\text{ if } u \neq l \qquad N_1|(\nu u)N_2 \equiv (\nu u)(N_1|N_2) \;\text{ if } u \notin fn(N_1)$$

The reduction axioms are:

$$l : a(x).P \mid l : \bar{a}\langle b\rangle.Q \longmapsto l : P\{^b/_x\} \mid l : Q \qquad l : go\_l'.P \mid l' : \mathbf{0} \longmapsto l : \mathbf{0} \mid l' : P$$

A computation step of a Dπ net can happen either because of a communication between co-located processes, or because a migration to a remote locality. Notice that a migration at l' is legal only if l' is an existing locality of the net. In the original paper [28], this check, among other tasks, is carried out by the type system. We prefer the present formulation for the sake of simplicity; however, all what are going to prove does not rely on this choice.

2.3 Mobile Ambients (MA)

We consider the Ambient calculus as presented in [15].

$$P ::= \ldots \;\big|\; (x).P \;\big|\; \langle M\rangle \;\big|\; M.P \;\big|\; u[P]$$
$$M ::= u \;\big|\; in\_u \;\big|\; out\_u \;\big|\; open\_u \;\big|\; M.M$$

MA is somewhat related to Dπ in the sense that processes are located within ambients (viz. $u[P]$) and only co-located processes can communicate via a monadic, asynchronous and anonymous communication: $(x).P$ represents the anonymous input prefix, whereas $\langle M\rangle$ represents the asynchronous and anonymous output particle, where message M can be not only a raw name but also a sequence of actions. However, differently from Dπ, 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.

Evaluation contexts are defined as follows:

$$\mathcal{E}(\cdot) \;::=\; \cdot \;\big|\; \mathcal{E}(\cdot) \,|\, P \;\big|\; P \,|\, \mathcal{E}(\cdot) \;\big|\; (\nu n)\mathcal{E}(\cdot) \;\big|\; n[\mathcal{E}(\cdot)]$$

The structural equivalence relation extends structural equivalence of π_a with the following axioms:

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

The reduction axioms are:

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

2.4 Safe Ambients (SA) and Safe Ambients with Passwords (SAP)

Safe Ambients We consider the Safe Ambient calculus as presented in [33]. SA extends MA by adding *co-actions*, through which ambient movements/openings must be authorized by the target ambient. Hence, the syntax of SA is the same as MA's, with

$$M \;::=\; \ldots \;\big|\; \overline{in}\_u \;\big|\; \overline{out}\_u \;\big|\; \overline{open}\_u$$

Evaluation contexts and structural equivalence are the same as for MA; the reduction axioms are:

$$(x).P \mid \langle M \rangle \longmapsto P\{^M/_x\} \qquad open\_n.P_1 \mid n[\overline{open}\_n.P_2|P_3] \longmapsto P_1 \mid P_2 \mid P_3$$
$$n[in\_m.P_1|P_2] \mid m[\overline{in}\_m.P_3|P_4] \longmapsto m[P_3 \mid P_4 \mid n[P_1|P_2]]$$
$$m[n[out\_m.P_1|P_2] \mid \overline{out}\_m.P_3 \mid P_4] \longmapsto n[P_1|P_2] \mid m[P_3|P_4]$$

Safe Ambients with Passwords In [35] SA has been enriched with passwords, thus yielding SAP. In this calculus, an ambient n that aims at entering/exiting/opening another ambient m must not only be authorized by m via a corresponding co-action (like in SA), but it must also exhibit some credential to perform the action (credentials are simply names and are called *passwords*). Intuitively, passwords are a way to better control ambient movements and openings: for example, in SA any ambient can open an ambient m that performs a $\overline{open}\_m$ action; with passwords, the co-action becomes $\overline{open}\_(m, p)$ and only the ambients knowing the password p can open m. Moreover, the language proposed in [35] differs from SA in the semantics of the *out* action: in SAP, the co-action is not in the ambient left (like in SA) but it is in the receiving ambient.

The introduction of passwords and of the different semantics for the *out* were needed in [35] to coinductively characterize barbed equivalence in a SA-like language. Here, we analyze the expressiveness implications of these two modifications. Formally, let SAP be the language defined by the syntax of SA, with

$$\begin{aligned} M \;::=\;\; & u \;\big|\; in\_(u, v) \;\big|\; out\_(u, v) \;\big|\; open\_(u, v) \;\big|\; \\ & \overline{in}\_(u, v) \;\big|\; \overline{out}\_(u, v) \;\big|\; \overline{open}\_(u, v) \;\big|\; M.M \end{aligned}$$

and with the mobility and opening axioms modified as follows:

$$open\_(n, p).P_1 \mid n[\overline{open}\_(n, p).P_2|P_3] \longmapsto P_1 \mid P_2 \mid P_3$$
$$n[in\_(m, p).P_1|P_2] \mid m[\overline{in}\_(m, p).P_3|P_4] \longmapsto m[P_3 \mid P_4 \mid n[P_1|P_2]]$$
$$m[n[out\_(m, p).P_1|P_2] \mid P_3] \mid \overline{out}\_(m, p).P_4 \;\longmapsto\; n[P_1|P_2] \mid m[P_3] \mid P_4$$

2.5 The Family of Boxed-like Ambients (BA, BA_s, SBA, NBA)

Boxed Ambients We consider the original presentation of the Boxed Ambient calculus [7]. BA evolves from MA by removing the *open* action. To let different ambients communicate, BA allows a restricted form of non-local communication: in particular, every input/output action can be performed locally (if tagged with direction $\star$), towards the enclosing ambient (if tagged with direction $^\uparrow$) or towards an enclosed ambient n (if tagged with direction n).

$$P ::= \ldots \mid (x)^\eta.P \mid \langle M\rangle^\eta.P \mid M.P \mid u[P]$$
$$M ::= u \mid in\_u \mid out\_u \mid M.M \qquad \eta ::= \star \mid ^\uparrow \mid u$$

Evaluation contexts and structural equivalence are the same as for MA; the reduction axioms are:

$$n[in\_m.P_1|P_2] \mid m[P_3] \longmapsto m[P_3 \mid n[P_1|P_2]]$$
$$m[n[out\_m.P_1|P_2] \mid P_3] \longmapsto n[P_1|P_2] \mid m[P_3]$$
$$(x)^\star.P_1 \mid \langle M\rangle^\star.P_2 \longmapsto P_1\{^M/_x\} \mid P_2$$
$$(x)^\star.P_1 \mid n[\langle M\rangle^\uparrow.P_2|P_3] \longmapsto P_1\{^M/_x\} \mid n[P_2|P_3]$$
$$(x)^n.P_1 \mid n[\langle M\rangle^\star.P_2|P_3] \longmapsto P_1\{^M/_x\} \mid n[P_2|P_3]$$
$$\langle M\rangle^\star.P_1 \mid n[(x)^\uparrow.P_2|P_3] \longmapsto P_1 \mid n[P_2\{^M/_x\}|P_3]$$
$$\langle M\rangle^n.P_1 \mid n[(x)^\star.P_2|P_3] \longmapsto P_1 \mid n[P_2\{^M/_x\}|P_3]$$

Boxed Ambients with Shared Channels In BA the communication channel is *localized*, i.e. communications can happen either within the same ambient or via a channel owned by either the parent or the child. However, parent-child communications can be modeled in (at least) another way, i.e. by letting the communication channel be *shared*. In this case, remote communications happen via a channel shared by the parent and its child. The resulting calculus [7], that we call BA_s, has the same syntax as BA, but it has just two axioms for remote communications (instead of four):

$$(x)^n.P_1 \mid n[\langle M\rangle^\uparrow.P_2|P_3] \longmapsto P_1\{^M/_x\} \mid n[P_2|P_3]$$
$$\langle M\rangle^n.P_1 \mid n[(x)^\uparrow.P_2|P_3] \longmapsto P_1 \mid n[P_2\{^M/_x\}|P_3]$$

BA_s provides a more controlled form of communication, since it rules out the interferences that can arise, e.g., in the BA process

$$(x)^n \mid n[\langle M\rangle^\star \mid (y)^\star \mid m[(z)^\uparrow]]$$

where message M can be consumed by three different input actions placed in different ambients. By contrast, in BA_s this process can only perform the local communication.

Safe Boxed Ambients Another variant of BA is SBA (*Safe* BA, [36]): it is BA extended with co-actions to better control ambient movements, in the same spirit as SA. The syntax of SBA is the same as BA, with messages defined as follows:

$$M ::= u \mid in\_u \mid out\_u \mid \overline{in}\_\delta \mid \overline{out}\_\delta \mid M.M \qquad \delta ::= * \mid v$$

The reduction axioms for ambient movements are:

$$n[in\_m.P_1 \mid P_2] \mid m[\overline{in}\_\delta.P_3 \mid P_4] \longmapsto m[n[P_1 \mid P_2] \mid P_3 \mid P_4] \qquad \text{for } \delta \in \{n, *\}$$

$$m[n[out\_m.P_1 \mid P_2] \mid P_3] \mid \overline{out}\_\delta.P_4 \longmapsto n[P_1 \mid P_2] \mid m[P_3] \mid P_4 \qquad \text{for } \delta \in \{n, *\}$$

In SBA, like in SA, an ambient n can enter into (exit from) an ambient m only if authorized from a co-action. However, differently from SA, the co-action either names the ambient that is allowed to enter (exit), or it can specify that every ambient is allowed to do so, via the '$*$' tag. Moreover, notice that the $\overline{out}$ action is placed outside the ambient left, like in SAP.

New Boxed Ambients [8] presents an evolution of BA, called NBA (*New* BA), that adopts the shared-channel form of communication of BA_s, it introduces passwords in mobility actions (similarly to SAP) and let co-actions dynamically learn the name of the ambient that performed the corresponding action. Formally, its syntax extends the one of BA, by adding co-actions as prefixes and with the introduction of passwords:

$$P ::= \ldots \mid \overline{in}\_(x, v).P \mid \overline{out}\_(x, v).P \qquad\qquad M ::= u \mid in\_(u, v) \mid out\_(u, v) \mid M.M$$

The reduction axioms for ambient movements are:

$$n[in\_(m, p).P_1 \mid P_2] \mid m[\overline{in}\_(x, p).P_3 \mid P_4] \longmapsto m[n[P_1 \mid P_2] \mid P_3\{^n/_x\} \mid P_4]$$

$$m[n[out\_(m, p).P_1 \mid P_2] \mid P_3] \mid \overline{out}\_(x, p).P_4 \longmapsto n[P_1 \mid P_2] \mid m[P_3] \mid P_4\{^n/_x\}$$

In NBA, like in SAP, an ambient n can enter into (exit from) an ambient m only if authorized from a co-action and after the successful matching of a password p. However, differently from any other ambient-based calculus seen so far, the co-action is used to also learn the name of the ambient that is entering (exiting) and this name can be used in the continuation process.

3 Valid Encodings

An *encoding* of $\mathcal{L}_1 = (\mathcal{P}_1, \longmapsto_1, \simeq_1)$ into $\mathcal{L}_2 = (\mathcal{P}_2, \longmapsto_2, \simeq_2)$ 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 names $u \neq v$, 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 Section 4 we shall see examples of both kinds of such encodings). This justifies the presence of $\varphi_{\llbracket\,\rrbracket}$. To simplify reading, we shall usually write $\llbracket \cdot \rrbracket$ instead of $(\llbracket \cdot \rrbracket, \varphi_{\llbracket\,\rrbracket})$, by leaving the renaming policy understood.

An encoding is *valid* if it satisfies the five properties we are going to present now. There, 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., $\mathcal{L}_2$).

3.1 Properties for Valid Encodings

As already said in the introduction, an encoding should be compositional. To formally define this notion, we exploit the notion of *k-ary context*, written $C(\__1; \ldots; \__k)$, that is a term where k occurrences of $\mathbf{0}$ are linearly replaced by the k holes $\__1, \ldots, \__k$ (a context is *linear* if every hole occurs exactly once; see [52].)

Property 1. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* compositional *if, for every k-ary* $\mathcal{L}_1$*-operator* op *and finite subset of names N, there exists a k-ary* $\mathcal{L}_2$*-context* $C^N_{\mathsf{op}}(\_1; \ldots; \_k)$ *such that* $[\![\, \mathsf{op}(S_1, \ldots, S_k) \,]\!] = C^N_{\mathsf{op}}([\![\, S_1 \,]\!]; \ldots; [\![\, S_k \,]\!])$*, for every* $S_1, \ldots, S_k$ *with* $fn(S_1, \ldots, S_k) = N$*.*

Moreover, a good encoding should reflect in the encoded term all the name substitutions carried out in the source term. A substitution (of names for names) σ is a function $\sigma : \mathcal{N} \longrightarrow \mathcal{N}$. 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. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* name invariant *if, for every substitution* σ*, it holds that*

$$[\![\, S\sigma \,]\!] \begin{cases} = [\![\, S \,]\!]\sigma' & \text{if } \sigma \text{ is injective} \\ \simeq_2 [\![\, S \,]\!]\sigma' & \text{otherwise} \end{cases}$$

where σ' *is the substitution such that* $\varphi_{[\![\,]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![\,]\!]}(a))$*.*

Injectivity of σ must be taken into account because non-injective substitutions map distinct names to the same name, and this matters because compositionality also depends on the free names occurring in the encoded terms. Indeed, 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 '$\simeq_2$') 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.

A source term and its encoding should have the same operational behaviour, i.e. all the computations of the source term must be preserved by the encoding without introducing "new" computations. This intuition is formalized as follows.

Property 3. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* operationally corresponding *if*

- Completeness: *for every* S *and* S' *such that* $S \Longmapsto_1 S'$*, it holds that* $[\![\, S \,]\!] \Longmapsto_2 \simeq_2 [\![S']\!]$*;*
- Soundness: *for every* S *and* T *such that* $[\![\, S \,]\!] \Longmapsto_2 T$*, there exists* S' *such that* $S \Longmapsto_1 S'$ *and* $T \Longmapsto_2 \simeq_2 [\![\, S' \,]\!]$*.*

An important semantic issue that an encoding should avoid is the introduction of infinite computations, written $\longmapsto^\omega$, when translating a terminating process. Of course, once we assume that divergence is observable, it is natural to require that the encoding also preserves it (this is a slight difference w.r.t. [25], where we only require divergence reflection).

Property 4. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* divergence sensitive *whenever* $S \longmapsto^\omega_1$ *if and only if* $[\![\, S \,]\!] \longmapsto^\omega_2$*, for every* S*.*

Finally, we require that the source and the translated term behave in the same way with respect to success, a notion that can be used to define semantic theories for processes [18, 51]. To formulate our property in a simple way, we follow the approach in [51] and assume that all the languages contain the same success process $\surd$; then, we define the predicate $\searrow$, meaning reducibility (in some modality, e.g. may/must/fair-must) to a process containing a top-level unguarded occurrence of $\surd$ (i.e., an occurrence of $\surd$ that does not occur underneath any prefix). Clearly, different modalities in general lead to different results. In this paper, proofs will be carried out in a 'may' modality; so, $P \searrow$ means that there exists P' such that $P \Longrightarrow P'$ and $P' \equiv P'' \mid \surd$. For Dπ, this definition should be adapted to nets, by letting $N \searrow$ mean $\exists N'.N \Longrightarrow N' \wedge N' \equiv (\nu\widetilde{u})(N' \mid l : \surd \mid P)$. Finally, for the sake of coherence, we require the notion of success be caught by the semantic theory underlying the calculi, viz. $\simeq$; in particular, we assume that $\simeq$ never relates two processes P and Q such that $P \searrow$ and $Q \not\searrow$.

Property 5. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* success sensitive *if, for every* S*, it holds that* $S \searrow_1$ *iff* $[\![S]\!] \searrow_2$.

3.2 Derived Properties

In [25] we have shown that some separation results can be proved in the general framework we have just presented. However, to carry out more proofs, we have to slightly specialize the framework; this is mainly done by making some assumptions on the behavioural equivalence of the target language, viz. $\simeq_2$. In particular, in *loc.cit.* we have considered three alternative settings:

1. $\simeq_2$ is *exact*, i.e. $T \simeq_2 T'$ and T performs an action μ imply that T' (weakly) performs μ as well; moreover, parallel composition must be translated homomorphically, i.e. for every $N \subset \mathcal{N}$ it holds that $C^N_{|}(\__1; \__2) = \__1 \mid \__2$;

2. $\simeq_2$ is *reduction sensitive*, i.e. $T \simeq_2 T'$ and $T' \longmapsto_2$ imply that $T \longmapsto_2$;

3. the occurrences of $\simeq_2$ in Property 3 are restricted to pairs of kind $((\nu\widetilde{n})(T \mid T'), T)$, for $(\nu\widetilde{n})(T \mid T') \simeq_2 T$.

All these assumptions are discussed and justified at length in [25]. In particular, the third setting seems to us the most appropriate, since the purpose of $\simeq_2$ in Property 3 is exactly to garbage collect junk processes left by the encoding. Moreover, as stressed also in [25], pairs of kind $((\nu\widetilde{n})(T \mid T'), T)$ yield a reduction sensitive equivalence; so, every result proved in the second setting also holds in the third one. Finally, the first setting is not adequate for this paper, since all the calculi we consider (with the exception of Dπ) usually adopt behavioural equivalences that are *not* exact. To conclude, in this paper we confine ourselves to the third setting, even though the second one could work as well.

We can now list a number of auxiliary results that will be useful in carrying out the main proofs of this paper. Some of these results have been already proved in [25]; here, we adapt and prove them for the calculi considered in this paper.

Proposition 3.1 (from [25])**.** *Let* $[\![\cdot]\!]$ *be a valid encoding; then,* $S \mapfrom\!\!\!\!\rightarrow_1$ *implies that* $[\![S]\!] \mapfrom\!\!\!\!\rightarrow_2$.

Proof. By contradiction. If $[\![S]\!] \longmapsto_2 T$ then, by Property 3, $S \Longrightarrow S'$, for some S' such that $T \Longrightarrow_2 \simeq_2 [\![S']\!]$. But the only S' such that $S \Longrightarrow_1 S'$ is S itself; thus, $[\![S]\!] \longmapsto_2^+ \simeq_2 [\![S]\!]$, i.e. $[\![S]\!]$ diverges, against Property 4. □

Proposition 3.2 (from [25])**.** *Let $[\![\cdot]\!]$ be a valid encoding; if there exist two source terms S_1 and S_2 such that $S_1 \mid S_2 \searrow_1$, $S_1 \not\searrow_1$ and $S_2 \not\searrow_1$, then $[\![S_1 \mid S_2]\!] \longmapsto_2$.*

Proof. By Properties 1 and 5, $[\![S_1 \mid S_2]\!] = C^N_{|}([\![S_1]\!]; [\![S_2]\!]) \searrow_2$, for $N = fn(S_1, S_2)$. If $[\![S_1 \mid S_2]\!]$ did not reduce, then it could only be that $C^N_{|}([\![S_1]\!]; [\![S'_2]\!]) \searrow_2$ or $C^N_{|}([\![S'_1]\!]; [\![S_2]\!]) \searrow_2$, where S'_i such that $fn(S_i) = fn(S'_i)$ and $S'_i \simeq_1 \mathbf{0}$.[1] Property 5 would then imply that $S_1 \mid S'_2 \searrow_1$ or $S'_1 \mid S_2 \searrow_1$; this is not possible, since $S_1 \mid S'_2 \simeq_1 S_1 \not\searrow_1$ and $S'_1 \mid S_2 \simeq_1 S_2 \not\searrow_1$. Indeed, we have assumed that $\simeq_1$ is sensitive to successful termination, i.e. it cannot equate two processes different w.r.t. successful termination. □

Proposition 3.3. *Let $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ be a valid encoding. If there exist two source terms S_1 and S_2 such that $S_1 \not\mapsto_1$, $S_2 \not\mapsto_1$ and $[\![S_1 \mid S_2]\!] \longmapsto_2$, then*

1. *if $\mathcal{L}_2 \in \{\pi_a, \mathrm{D}\pi\}$, it can only be that $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto_2$;*
2. *if $\mathcal{L}_2 \in \{\mathrm{MA}, \mathrm{SA}, \mathrm{SAP}, \mathrm{BA}, \mathrm{BA}_s, \mathrm{SBA}, \mathrm{NBA}\}$, it can only be that $C_1([\![S_1]\!]) \mid C_2([\![S_2]\!]) \longmapsto_2$, where*
   - *$C^{fn(S_1,S_2)}_{|}(\_1; \_2)$, i.e. the context used to compositionally translate $S_1 \mid S_2$, is of the form $\mathcal{E}(C_1(\_1) \mid C_2(\_2))$ for some evaluation context $\mathcal{E}(\cdot)$*
   - *$C_i(\_i)$, for $i \in \{1, 2\}$, is either empty (viz., $\_i$) or a single top-level ambient containing a top-level hole (viz., $m[\ \_i \mid T]$, for some m and T).*

Proof. By Property 1, $[\![S_1 \mid S_2]\!] = C^{fn(S_1,S_2)}_{|}([\![S_1]\!]; [\![S_2]\!])$. The reduction of $[\![S_1 \mid S_2]\!]$ must be originated with the contribution of both $[\![S_1]\!]$ and $[\![S_2]\!]$; if it was not the case, we could always find S'_1 and S'_2 such that $fn(S_i) = fn(S'_i)$ and S'_i cannot act in any way (nor in isolation nor in any evaluation context), and have that $[\![S_1 \mid S'_2]\!] \longmapsto_2$ or $[\![S'_1 \mid S_2]\!] \longmapsto_2$, in contradiction with Proposition 3.1.

1. If $\mathcal{L}_2 \in \{\pi_a, \mathrm{D}\pi\}$, then this can only happen when $[\![S_1]\!]$ and $[\![S_2]\!]$ are put in parallel at top-level (i.e., $C^{fn(S_1,S_2)}_{|}(\_1 ; \_2) = (\nu\widetilde{n})(\_1 \mid \_2 \mid T)$, for some $\widetilde{n}$ and T) and the reduction originates from $[\![S_1]\!] \mid [\![S_2]\!]$.
2. If $\mathcal{L}_2 \in \{\mathrm{MA}, \mathrm{SA}, \mathrm{SAP}, \mathrm{BA}, \mathrm{BA}_s, \mathrm{SBA}, \mathrm{NBA}\}$, the situation is more complex: indeed, it can be that
   - either $[\![S_1]\!]$ and $[\![S_2]\!]$ are put in parallel (for example, to perform a communication);
   - or $[\![S_1]\!]$ is put in parallel with an ambient containing $[\![S_2]\!]$ at top-level (for example, to perform an open or a remote communication), or vice versa;
   - or both $[\![S_1]\!]$ and $[\![S_2]\!]$ are placed at top-level within some ambients put in parallel (for example, to perform an entrance of one into the other).

   and all these cases may happen at every nesting level in the ambient hierarchy. This can be expressed by saying that $C^{fn(S_1,S_2)}_{|}(\_1; \_2)$ is of the form $\mathcal{E}(C_1(\_1) \mid C_2(\_2))$, for some evaluation context $\mathcal{E}(\cdot)$ and contexts $C_1(\_1)$ and $C_2(\_2)$ that are either empty or a single ambient with a top-level hole. Moreover, since both $[\![S_1]\!]$ and $[\![S_2]\!]$ must contribute to the reduction of $[\![S_1 \mid S_2]\!]$, it must be that $C_1([\![S_1]\!]) \mid C_2([\![S_2]\!]) \longmapsto_2$. □

[1] It is always possible to find such an S'_i: it suffices to place in front of S_i a restricted blocking prefix without free names. For example, in π_a we could let $S' \triangleq (\nu a)a(x).S$, for any $a \notin fn(S)$; similar tricks can be applied to MA, SA, BA and all the other ambient-based calculi. The situation for Dπ is slightly more delicate: given a net S, we define S' to be the net $(\nu a)N$, where $a \notin fn(S)$ and N is obtained from S by placing the prefix $a(x)$ in fron of every process at every locality.

Theorem 3.4 (Adapted from [25]). *Let $\mathcal{L}_2 \in \{\pi_a, \mathrm{D}\pi\}$. Assume that there is a $\mathcal{L}_1$-process S such that $S \not\mapsto_1$, $S \not\searrow_1$ and $S \mid S \searrow_1$; then, there cannot exist any valid encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.*

Proof. We first show that, in π_a and Dπ, it holds that $T \mid T \longmapsto_2$ entails $T \longmapsto_2$, for every process/net T.

- If T is a π_a-process such that $T \mid T \longmapsto_2$, then $T \equiv (\nu \widetilde{n})(a(x).T' \mid \bar{a}\langle b \rangle \mid T'')$ for some $a \notin \widetilde{n}$. Thus, trivially, $T \longmapsto_2$.
- If T is a Dπ-process such that $T \mid T \longmapsto_2$, then $T \equiv (\nu \widetilde{n})(l : a(x).P \mid l : \bar{a}\langle b \rangle.Q \mid T')$ for $\{l, a\} \cap \widetilde{n} = \emptyset$, or $T \equiv (\nu \widetilde{n})(l : go\_k.P \mid k : \mathbf{0} \mid T')$ for $k \notin \widetilde{n}$. Thus, trivially, $T \longmapsto_2$.

Now, let us work by contradiction. Let S be such that $S \not\mapsto_1$, $S \not\searrow_1$ and $S \mid S \searrow_1$; by Proposition 3.3, $[\![S]\!] \mid [\![S]\!] \longmapsto_2$ that, as just shown, implies $[\![S]\!] \longmapsto_2$, in contradiction with Proposition 3.1. □

To state the following proof-technique, let us define the *matching degree* of a language $\mathcal{L}$, written $\mathrm{MD}(\mathcal{L})$, as the greatest number of names that must be matched to yield a reduction in $\mathcal{L}$. Formally,

Definition 3.1 (Matching degree). *We say that $\mathcal{L}$ atomically matches n names if, for every pair of $\mathcal{L}$-processes P and Q such that $P \not\mapsto$, $Q \not\mapsto$ and $P \mid Q \longmapsto$, it holds that $|fn(P) \cap fn(Q)| \geq n$. We let*

$$\mathrm{MD}(\mathcal{L}) \triangleq \sup\{\, n \;:\; \mathcal{L} \textit{ atomically matches } n \textit{ names}\}$$

The matching degree of π_a is 1 because the axiom for communication only checks whether the input and the output happen along the same channel. The matching degree of MA, BA, BA$_s$ and SA is 1: for communicating, no name is matched in MA and SA, and at most one name in BA and BA$_s$ (for communications towards a child); for ambient interactions, only the name of the ambient entered/left/opened is checked. For Dπ the matching degree is 2: for communicating, two processes must be co-located and perform an input and an output along the same channel. For SBA the matching degree is 2: consider the reductions for *in* and *out* when $\delta = n$. Finally, the matching degree of SAP and NBA is 2: both the name of the ambient entered/left/opened and the password are checked.

Theorem 3.5 (Adapted from [25]). *If $\mathrm{MD}(\mathcal{L}_1) = 2$ and $\mathrm{MD}(\mathcal{L}_2) = 1$, 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 3.2 and that synchronize only once (before reporting success) by matching exactly 2 names, viz. $\{n_1, n_2\}$. Let us also assume that S_1 and S_2 contain another name $m \notin \{n_1, n_2\}$. By Proposition 3.3, their encodings must both contribute to the reduction: i.e.,

- If $\mathcal{L}_2$ is π_a, then $[\![S_1]\!]$ must perform some action μ and $[\![S_2]\!]$ must perform some complementary action $\overline{\mu}$.
- If $\mathcal{L}_2 \in \{\text{MA, SA, BA, BA}_s\}$, then $C_1([\![S_1]\!])$ must perform some action μ and $C_2([\![S_2]\!])$ must perform some complementary action $\overline{\mu}$, where $C_i(\cdot)$ is defined in Proposition 3.3 and $[\![S_i]\!]$ must have contributed to the generation of the action.

Since $\text{MD}(\mathcal{L}_2) = 1$, it must be that at most one name $n \in fn(\mu) \cap fn(\overline{\mu})$ is matched when synchronizing μ and $\overline{\mu}$; this implies the existence of an n_i such that $n \notin \varphi_{[\![\,]\!]}(n_i)$. Let us consider the substitution σ that swaps m and n_i. Trivially, $S_1 \mid S_2\sigma \mapsto\!\!\!\!\!\!\rightarrow_1$ because, by construction, S_1 and S_2 can only synchronize by matching 2 names; thus, also S_1 and $S_2\sigma$ can only synchronize by matching 2 names and the match now fails, since S_1 contains n_i and $S_2\sigma$ contains m in place of it.

If $\mathcal{L}_2$ is π_a or at least one of the $C_i(\cdot)$ is empty (say, e.g., $C_2(\cdot)$), we can reason as follows. By Property 2, $[\![S_2\sigma]\!] = [\![S_2]\!]\sigma'$ and so $[\![S_2]\!]\sigma'$ performs action $\overline{\mu}\sigma'$. 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 n. Thus, $[\![S_1 \mid S_2\sigma]\!] \longmapsto_2$, in contradiction with Proposition 3.1.

If $C_1(\cdot) \triangleq m_1[\,\cdot \mid T_1]$ and $C_2(\cdot) \triangleq m_2[\,\cdot \mid T_2]$, then $\mathcal{L}_2$ must be SA, otherwise there would be no way to have that both $[\![S_1]\!]$ and $[\![S_2]\!]$ have contributed to the generation of the reduction. Hence, it must be that $m_i = n$, $[\![S_i]\!]$ has a top-level action $\overline{in}\_n$ and $[\![S_j]\!]$ has a top-level action in_n, for $\{i, j\} = \{1, 2\}$. By repeating the reasoning for the case in which $C_j(\cdot)$ is empty, we can conclude. Indeed, $C_j([\![S_j\sigma]\!])$ exhibits a top-level ambient aiming at entering into n and $C_i([\![S_i]\!])$ exhibits a top-level ambient n containing the co-action enbling such an entrance. □

Proposition 3.6. *Let $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ be an encoding that satisfies Property 2; for every S and $n \notin fn(S)$, it holds that $\varphi_{[\![\,]\!]}(n) \cap fn([\![S]\!]) = \emptyset$.*

Proof. By contradiction, let $n' \in \varphi_{[\![\,]\!]}(n) \cap fn([\![S]\!])$. Let m be such that $m \notin fn(S)$ and $\varphi_{[\![\,]\!]}(m) \cap fn([\![S]\!]) = \emptyset$; moreover, let σ be the permutation that swaps m and n. Trivially, $S = S\sigma$ and, hence, $[\![S]\!] = [\![S\sigma]\!]$. However, by Property 2, $[\![S\sigma]\!] = [\![S]\!]\sigma'$, for σ' that swaps $\varphi_{[\![\,]\!]}(m)$ and $\varphi_{[\![\,]\!]}(n)$ component-wise. The only possible way to have that $[\![S]\!] = [\![S]\!]\sigma'$ (that holds because of transitivity) is to have $dom(\sigma') \cap fn([\![S]\!]) = \emptyset$ that, however, does not hold, because $dom(\sigma') = \varphi_{[\![\,]\!]}(n) \cup \varphi_{[\![\,]\!]}(m)$ and $n' \in \varphi_{[\![\,]\!]}(n) \cap fn([\![S]\!])$: contradiction. □

4 The Taxonomy, bottom-up

For every pair of languages, we study whether one is more expressive than the other, or if they are incomparable. In the first case, we provide a valid encoding of the less expressive language in the most expressive one and prove that the converse is not possible. In the second case, we must prove that no valid encoding of one in the other exists.

We now give the crucial results underlying the taxonomy in Figure 1. The remaining pairs of languages can be compared by proving theorems similar to the ones we are going to prove. Full details are given in Table 1 in Section 4.5.

4.1 Technical Preliminaries

For encodability results, we shall rely on the notion of *barbed equivalence* [41], that is a uniformly defined notion of equivalence nowadays considered one of the reference equivalences in process calculi. It relies on a notion of *barb*, that is something observable that a process exhibits; essentially, it relates processes that exhibit the same barbs in any evaluation context and along any sequence of reductions.

Definition 4.1 (from [41])**.** *A symmetric binary relation $\mathfrak{R}$ on processes is a* (strong) barbed bisimulation *if, for every $(P, Q) \in \mathfrak{R}$, it holds that $P \downarrow_b$ if and only if $Q \downarrow_b$; moreover, for every $P \longmapsto P'$ there exists a Q' such that $Q \longmapsto Q'$ and $(P', Q') \in \mathfrak{R}$.* Barbed bisimilarity*, written $\dot{\simeq}$, is the largest barbed*

bisimulation. Two processes P and Q are barbed equivalent*, written $P \simeq Q$, if $\mathcal{E}(P) \stackrel{\bullet}{\simeq} \mathcal{E}(Q)$, for every evaluation context $\mathcal{E}(\cdot)$.*

We now define the barb predicate $\downarrow_b$ only for those languages that will be the target of an encoding where operational correspondence is formulated up-to $\simeq$. These definitions are the standard notions of barb for the given languages.

Definition 4.2 (Barbs)**.** *Predicate $P \downarrow_b$ is defined as follows:*

MA**,** BA **and** BA$_s$**:** *$\exists\, \widetilde{n}, Q, R$ such that $b \notin \widetilde{n}$ and $P \equiv (\nu\widetilde{n})(b[Q] \mid R)$;*

SA**:** *$\exists\, \widetilde{n}, Q, R$ such that $b \notin \widetilde{n}$ and $P \equiv (\nu\widetilde{n})(b[M.Q] \mid R)$, for $M \in \{\overline{in}\_b, \overline{open}\_b\}$;*

SAP**:** *$\exists\ \widetilde{n}, p, Q, R$ such that $\{b, p\} \cap \widetilde{n} = \emptyset$ and $P \equiv (\nu\widetilde{n})(b[M.Q] \mid R)$, for $M \in \{\overline{in}\_(b, p), \overline{open}\_(b, p)\}$;*

SBA**:** *$\exists\, \widetilde{n}, Q, R$ such that $b \notin \widetilde{n}$ and either $P \equiv (\nu\widetilde{n})(b[\overline{in}\_\delta.Q] \mid R)$, for some δ such that $fn(\delta) \cap \widetilde{n} = \emptyset$, or $P \equiv (\nu\widetilde{n})(b[(x)^{\eta}.Q] \mid R)$ or $P \equiv (\nu\widetilde{n})(b[\langle M\rangle^{\eta}.Q] \mid R)$, for some $\eta \in \{\star, \uparrow\}$;*

NBA**:** *$\exists\ \widetilde{n}, Q, R$ such that $b \notin \widetilde{n}$ and either $P \equiv (\nu\widetilde{n})(b[(x)^{\uparrow}.Q] \mid R)$ or $P \equiv (\nu\widetilde{n})(b[\langle M\rangle^{\uparrow}.Q] \mid R)$ or $P \equiv (\nu\widetilde{n})(b[\overline{in}\_(x, p).Q] \mid R)$, for some $p \notin \widetilde{n}$.*

To carry out proofs, we found it convenient to exploit the labeled transition systems developed for the languages studied. For the sake of conciseness, we do not give here full details on this topic; thus, we informally present only the technicalities strictly needed in our proofs.

Notation 4.1 (Labels for MA [37])**.**

- *If $P \xrightarrow{\langle - \rangle}$, then $P \equiv (\nu\widetilde{n})(\langle M\rangle \mid P')$, for some $\widetilde{n}, M, P'$;*
- *if $P \xrightarrow{(-)}$, then $P \equiv (\nu\widetilde{n})((x).P_1 \mid P_2)$, for some $\widetilde{n}, x, P_1, P_2$;*
- *if $P \xrightarrow{amb\_n}$, then $P \equiv (\nu\widetilde{n})(n[P_1] \mid P_2)$, for some $\widetilde{n}, P_1, P_2$ such that $n \notin \widetilde{n}$;*
- *if $P \xrightarrow{enter\_n}$, then $P \equiv (\nu\widetilde{n})(m[in\_n.P_1 \mid P_2] \mid P_3)$, for some $\widetilde{n}, m, P_1, P_2, P_3$ such that $n \notin \widetilde{n}$;*
- *if $P \xrightarrow{open\_n}$, then $P \equiv (\nu\widetilde{n})(open\_n.P_1 \mid P_2)$, for some $\widetilde{n}, P_1, P_2$ such that $n \notin \widetilde{n}$.*

Proposition 4.1 (Reductions in MA)**.** *In* MA*, it holds that $P_1 \mid P_2 \longmapsto$ if and only if one of the following conditions holds (possibly with P_1 and P_2 swapped):*

1. $P_1 \longmapsto$
2. $P_1 \xrightarrow{\langle - \rangle}$ *and* $P_2 \xrightarrow{(-)}$
3. $P_1 \xrightarrow{enter\_n}$ *and* $P_2 \xrightarrow{amb\_n}$
4. $P_1 \xrightarrow{open\_n}$ *and* $P_2 \xrightarrow{amb\_n}$

Notation 4.2 (Labels for SA [33] and SAP [35])**.** *In* SA*:*

- *If $P \xrightarrow{\mu}$, for $\mu \in \{\langle - \rangle, (-), enter\_n, open\_n\}$: see Notation 4.1;*
- *if $P \xrightarrow{?enter\_n}$, then $P \equiv (\nu\widetilde{n})(n[\overline{in}\_n.P_1 \mid P_2] \mid P_3)$, for some $\widetilde{n}, P_1, P_2, P_3$ such that $n \notin \widetilde{n}$;*

- *if* $P \xrightarrow{?open\_n}$, *then* $P \equiv (\nu\widetilde{n})(n[\overline{open}\_n.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, P_1, P_2, P_3$ *such that* $n \notin \widetilde{n}$.

In SAP, *labels also contain the specified password, originating labels* $enter\_(n,p)$, $?enter\_(n,p)$, $open\_(n,p)$ *and* $?open\_(n,p)$ *with the expected meaning; moreover, there are two more labels:*

- *if* $P \xrightarrow{exit\_(n,p)}$, *then* $P \equiv (\nu\widetilde{n})(n[m[out\_(n,p).P_1 \mid P_2] \mid P_3] \mid P_4)$, *for some* $\widetilde{n}, m, P_1, P_2, P_3, P_4$ *such that* $\{n,p\} \cap \widetilde{n} = \emptyset$;
- *if* $P \xrightarrow{?exit\_(n,p)}$, *then* $P \equiv (\nu\widetilde{n})(\overline{out}\_(n,p).P_1 \mid P_2)$, *for some* $\widetilde{n}, P_1, P_2$ *such that* $\{n,p\} \cap \widetilde{n} = \emptyset$.

Proposition 4.2 (Reductions in SA and SAP)**.** *In* SA, *it holds that* $P_1 \mid P_2 \longmapsto$ *if and only if one of the following conditions holds (possibly with* P_1 *and* P_2 *swapped):*

1., 2.: like the corresponding points in Proposition 4.1

3. $P_1 \xrightarrow{enter\_n}$ *and* $P_2 \xrightarrow{?enter\_n}$
4. $P_1 \xrightarrow{open\_n}$ *and* $P_2 \xrightarrow{?open\_n}$

For SAP, *the previous transitions are also labeled with the password associated to the action fired; moreover, there is one more possible interaction:*

$$P_1 \xrightarrow{exit\_(n,p)} \text{ and } P_2 \xrightarrow{?exit\_(n,p)}$$

Notation 4.3 (Labeled actions[2] for BA, BA_s, SBA and NBA [8])**.** *In* BA*:*

- *If* $P \xrightarrow{\mu}$, *for* $\mu \in \{enter\_n, amb\_n\}$*: see Notation 4.1;*
- *if* $P \xrightarrow{\langle - \rangle^\star}$, *then* $P \equiv (\nu\widetilde{n})(\langle M \rangle^\star.P_1 \mid P_2)$, *for some* $\widetilde{n}, M, P_1, P_2$;
- *if* $P \xrightarrow{(-)^\star}$, *then* $P \equiv (\nu\widetilde{n})((x)^\star.P_1 \mid P_2)$, *for some* $\widetilde{n}, x, P_1, P_2$;
- *if* $P \xrightarrow{\mu}$, *for* $\mu \in \{\langle - \rangle^n, (-)^n\}$, *we adapt the previous two cases by replacing* $\langle M \rangle^\star$ *and* $(x)^\star$ *with* $\langle M \rangle^n$ *and* $(x)^n$, *and by also imposing that* $n \notin \widetilde{n}$;
- *if* $P \xrightarrow{up\langle - \rangle}$, *then* $P \equiv (\nu\widetilde{n})(n[\langle M \rangle^\uparrow.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, n, M, P_1, P_2, P_3$;
- *if* $P \xrightarrow{up(-)}$, *then* $P \equiv (\nu\widetilde{n})(n[(x)^\uparrow.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, n, x, P_1, P_2, P_3$;
- *if* $P \xrightarrow{\mu}$, *for* $\mu \in \{n\langle - \rangle, n(-)\}$, *we adapt the previous two cases by replacing* $\langle M \rangle^\uparrow$ *and* $(x)^\uparrow$ *with* $\langle M \rangle^\star$ *and* $(x)^\star$, *and by also imposing that* $n \notin \widetilde{n}$;

In BA_s, *labels* $up\langle - \rangle$ *and* $up(-)$ *are not exploited; moreover, if* $P \xrightarrow{\mu}$, *for* $\mu \in \{n\langle - \rangle, n(-)\}$, *then* $P \equiv (\nu\widetilde{n})(n[\langle - \rangle^\uparrow.P_1 \mid P_2] \mid P_3)$ *or* $P \equiv (\nu\widetilde{n})(n[(x)^\uparrow.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, n, M, x, P_1, P_2, P_3$ *such that* $n \notin \widetilde{n}$.

In SBA, *we have the same labels as in* BA, *but label* amb_n *is not used anymore and label* $enter\_n$ *is defined as*

[2] We are not aware of any LTS for BA, BA_s and SBA in the literature. However, they can be formally defined by following the philosophy underlying the LTS for NBA and for the other ambient-based calculi.

- *if* $P \xrightarrow{enter\_n}$, *then* $P \equiv (\nu\widetilde{n})(m[in\_n.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, m, P_1, P_2, P_3$ *such that* $n \notin \widetilde{n}$ *and* $m \in \widetilde{n}$;

Moreover, we also have the following new labels:

- *if* $P \xrightarrow{m:enter\_n}$, *then* $P \equiv (\nu\widetilde{n})(m[in\_n.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, P_1, P_2, P_3$ *such that* $\{m, n\} \cap \widetilde{n} = \emptyset$;
- *if* $P \xrightarrow{m:?enter\_}$, *then* $P \equiv (\nu\widetilde{n})(m[\overline{in}\_ \ast.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, P_1, P_2, P_3$ *such that* $m \notin \widetilde{n}$;
- *if* $P \xrightarrow{m:?enter\_n}$, *then* $P \equiv (\nu\widetilde{n})(m[\overline{in}\_\delta.P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, \delta, P_1, P_2, P_3$ *such that* $\delta \in \{\ast, n\}$ *and* $\{m, n\} \cap \widetilde{n} = \emptyset$;
- *if* $P \xrightarrow{m:exit\_}$, *then* $P \equiv (\nu\widetilde{n})(n[m[out\_n.P_1 \mid P_2] \mid P_3] \mid P_4)$, *for some* $\widetilde{n}, n, P_1, P_2, P_3, P_4$ *such that* $m \notin \widetilde{n}$;
- *if* $P \xrightarrow{exit\_}$, *then* $P \equiv (\nu\widetilde{n})(n[m[out\_n.P_1 \mid P_2] \mid P_3] \mid P_4)$, *for some* $\widetilde{n}, n, P_1, P_2, P_3, P_4$ *such that* $m \in \widetilde{n}$;
- *if* $P \xrightarrow{?exit\_n}$, *then* $P \equiv (\nu\widetilde{n})(\overline{out}\_\delta.P_1 \mid P_2)$, *for some* $\widetilde{n}, \delta, P_1, P_2$ *such that* $\delta \in \{\ast, n\}$ *and* $n \notin \widetilde{n}$.
- *if* $P \xrightarrow{?exit\_}$, *then* $P \equiv (\nu\widetilde{n})(\overline{out}\_ \ast.P_1 \mid P_2)$, *for some* $\widetilde{n}, P_1, P_2$.

In NBA *we have labels* $\langle - \rangle^\star, (-)^\star, \langle - \rangle^n, (-)^n, n\langle - \rangle$ *and* $n(-)$ *that are defined like in* BA_s*: moreover, we also have the following labels:*

- *if* $P \xrightarrow{m:enter\_(n,p)}$, *then* $P \equiv (\nu\widetilde{n})(m[in\_(n,p).P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, P_1, P_2, P_3$ *such that* $\{n, p\} \cap \widetilde{n} = \emptyset$;
- *if* $P \xrightarrow{m:?enter\_(n,p)}$, *then* $P \equiv (\nu\widetilde{n})(m[\overline{in}\_(x,p).P_1 \mid P_2] \mid P_3)$, *for some* $\widetilde{n}, x, P_1, P_2, P_3$ *such that* $\{m, n, p\} \cap \widetilde{n} = \emptyset$;
- *if* $P \xrightarrow{m:exit\_p}$, *then* $P \equiv (\nu\widetilde{n})(n[m[out\_(n,p).P_1 \mid P_2] \mid P_3] \mid P_4)$, *for some* $\widetilde{n}, n, P_1, P_2, P_3, P_4$ *such that* $p \notin \widetilde{n}$;
- *if* $P \xrightarrow{?exit\_(n,p)}$, *then* $P \equiv (\nu\widetilde{n})(\overline{out}\_(x,p).P_1 \mid P_2)$, *for some* $\widetilde{n}, x, P_1, P_2$ *such that* $\{n, p\} \cap \widetilde{n} = \emptyset$.

Proposition 4.3 (Reductions in BA, BA_s, SBA and NBA)**.** *In* BA*, it holds that* $P_1 \mid P_2 \longmapsto$ *if and only if one of the following conditions holds (possibly with* P_1 *and* P_2 *swapped):*

1., 2., 3.: like the corresponding points in Proposition 4.1, with $\langle - \rangle^\star/(-)^\star$ *in place of* $\langle - \rangle/(-)$

4. $P_1 \xrightarrow{\langle - \rangle^n}$ *and* $P_2 \xrightarrow{n(-)}$
5. $P_1 \xrightarrow{(-)^n}$ *and* $P_2 \xrightarrow{n\langle - \rangle}$
6. $P_1 \xrightarrow{\langle - \rangle^\star}$ *and* $P_2 \xrightarrow{up(-)}$
7. $P_1 \xrightarrow{(-)^\star}$ *and* $P_2 \xrightarrow{up\langle - \rangle}$

In BA_s *only the first five interactions are possible.*

For SBA*, we have cases 1, 2, 4, 5, 6 and 7 of* BA*; moreover, there are four further possible interactions:*

$$P_1 \xrightarrow{n:enter\_m} \text{and } P_2 \xrightarrow{m:?enter\_n}$$
$$P_1 \xrightarrow{enter\_m} \text{and } P_2 \xrightarrow{m:?enter\_}$$
$$P_1 \xrightarrow{n:exit\_} \text{and } P_2 \xrightarrow{?exit\_n}$$
$$P_1 \xrightarrow{exit\_} \text{and } P_2 \xrightarrow{?exit\_}$$

For NBA*, we have cases 1, 2, 4 and 5 of* BA_s*; moreover, there are two further possible interactions:*

$$P_1 \xrightarrow{n:enter\_(m,p)} \text{and } P_2 \xrightarrow{m:?enter\_(n,p)}$$
$$P_1 \xrightarrow{n:exit\_p} \text{and } P_2 \xrightarrow{?exit\_(n,p)}$$

4.2 Building up the first level of the Taxonomy

4.2.1 Dπ is more expressive than π_a

It may seem that π_a can be trivially encoded in Dπ: it suffices to locate the π_a process in a reserved locality hosting all the channels needed. Actually, this encoding is *not* valid, because compositionality as defined in Property 1 fails. Indeed, we can easily prove the following result. However, this should not be surprising, since the source language has a "one-level" syntax, whereas the target one has not.

Theorem 4.4. *There exists no valid encoding of* π_a *into* Dπ.

Proof. By contradiction. Every $[\![\cdot]\!] : \pi_a \to D\pi$ should map a source process into a target net. Now, consider the process $a(x).P$; by compositionality, its encoding must be $C^{fn(P)}_{a(x)}([\![P]\!])$. Now, $C^{fn(P)}_{a(x)}(\cdot)$ cannot be obtained from a Dπ-net by replacing an occurrence of $\mathbf{0}$ located at some locality with the hole, otherwise $C^{fn(P)}_{a(x)}([\![P]\!])$ would locate a net (viz., $[\![P]\!]$) at that locality. Thus, $[\![a(x).P]\!] \equiv (\nu \widetilde{u})([\![P]\!] \mid N)$, for some $\widetilde{u}$ and N. If we now choose a diverging P, we have that $[\![a(x).P]\!]$ diverges whereas $a(x).P$ does not, against Property 4. □

Nevertheless, the trivial encoding proposed before enjoys what [45] calls *weak* compositionality: the encoding presented satisfies Properies 2/.../5 and $[\![P]\!] \triangleq l : P$, i.e. the top-level encoding is defined by a top-level context (viz., $l : \_$) and by a compositional process translation (i.e., the identity in this case). Of course, this encodability result is not as strong as the other ones in this paper, but it is similar to other encodings in the literature (e.g., [4, 5, 9]).

On the contrary, Dπ cannot be encoded in π_a, as a corollary of Theorem 3.5, since $\text{MD}(\text{D}\pi) = 2$ and $\text{MD}(\pi_a) = 1$.

4.2.2 BA and BA_s are more expressive than π_a

A valid encoding of π_a in BA is provided in [7]. It is a homomorphism w.r.t. all the operators, except for

$$[\![u(x).P]\!] \triangleq (x)^u.[\![P]\!]$$
$$[\![\bar{u}\langle v \rangle]\!] \triangleq (\nu k)(u[\langle v \rangle^\star.in\_k] \mid k[\mathbf{0}]) \qquad \text{for } k \notin \{u, v\}$$

Moreover, it is easy to prove that it satisfies all the properties mentioned in Section 3.1, with operational correspondence that holds up-to strong barbed equivalence.

The encoding of the π_a in BA_s is similar. It suffices to replace $\langle v\rangle^\star$ with $\langle v\rangle^\uparrow$ in the encoding of the output particle.

The fact that BA and BA_s cannot be encoded in π_a is proved in the following result.

Theorem 4.5. *There exists no valid encoding of* BA *and* BA_s *in* π_a.

Proof. This is a Corollary of Theorem 3.4: it suffices to prove that we can find in BA a process S such that $S \mapsto\!\!\!\!\!\!\rightarrow_1$, $S \searrow\!\!\!\!\!\!\searrow_1$ and $S \mid S \searrow_1$. Let S be $(\nu p)(n[in\_n.p[out\_n.out\_n.\langle p\rangle^\star]] \mid (x)^p.\surd)$. For BA_s, it suffices to replace $\langle p\rangle^\star$ with $\langle p\rangle^\uparrow$. □

4.2.3 MA is more expressive than π_a

First, notice that MA cannot be encoded in π_a:

Theorem 4.6. *There exists no valid encoding of* MA *in* π_a.

Proof. Similar to the proof of Theorem 4.5, by considering the MA process $(\nu p)(n[in\_n.p[out\_n.out\_n]] \mid open\_p.\surd)$. □

We are left with proving that π_a can be encoded in MA; this is not a trivial task, if we want to satisfy all the properties in Section 3.1. Indeed, in several papers [13, 14, 15] there are attempts to encode π_a in MA, but none of them satisfies operational soundness and divergence sensitiveness. What we have proposed in [24] is, to the best of our knowledge, the first valid encoding of π_a in MA. However, the encoding that we proposed in *loc.cit.* is not very intuitive and its validity is quite difficult to prove. For this reason, here we give a more intuitive encoding, whose validity will also be easier to prove. The price to be paid concerns efficiency: a single reduction of π_a requires more reductions here than in the valid encoding of [24].

The encoding relies on a renaming policy that maps every name a to a pair of pairwise different names (a_1, a_2), and fixes three reserved names `poly`, `p` and `q`. Such renaming policy can be obtained by linearly ordering the set of names $\mathcal{N}$ as $\{n_0, n_1, n_2, \ldots\}$, by letting `poly` be n_0, `p` be n_1, `q` be n_2 and $\varphi_{[\![\,]\!]}(n_i) \triangleq (n_{3+2i}, n_{3+2i+1})$, for every i. The translation is a homomorphism w.r.t. all the operators, except for restrictions, inputs and outputs, that are translated as follows:

$$
\begin{aligned}
[\![(\nu a)P]\!] &\triangleq (\nu\, a_1, a_2)[\![P]\!] \\
[\![\bar a\langle b\rangle]\!] &\triangleq a_1[\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle]] \\
[\![a(x).P]\!] &\triangleq open\_a_1.(\nu r, s, t, u)(open\_u \mid t[u[open\_a_2.out\_t]] \\
&\qquad\qquad \mid a_2[\mathtt{q}[in\_\mathtt{p}.(x_1, x_2).in\_r.s[out\_\mathtt{p}.out\_r.in\_t.in\_u.[\![P]\!]]] \\
&\qquad\qquad\quad \mid r[\,] \mid open\_s]\,) \\
&\quad \text{for } r, s, t, u \notin \{x_1, x_2, a_2, \mathtt{p}, \mathtt{q}\} \cup fn([\![P]\!])
\end{aligned}
$$

where (x_1, x_2) is a shortcut for $(x_1).open\_\mathtt{poly}.(x_2)$ and $\langle b_1, b_2\rangle$ is a shortcut for $\langle b_1\rangle \mid \mathtt{poly}[\langle b_2\rangle]$.

Let us now explain how the encoding works. First, to start a communication, we must check that there are processes aiming at performing an output and an input along the same channel a; in doing this, we must be sure that an output along a enables only one input from a. In MA this can be done only via an *open* primitive that involves a name related to a (in particular, a_1). Indeed, an in_a_1 is not suitable, otherwise it could be used to introduce divergence. Consider, for example, the encoding of $!a(x) \mid \bar a\langle b\rangle$:

by starting the encoding of the input via an in_a_1, we would have an infinite computation arising by letting infinitely many copies of $[\![a(x)]\!]$ enter into the same a_1 ambient corresponding to the encoding of the output particle $\bar{a}\langle b\rangle$.

Once a_1 has been opened, we are sure that a communication can happen. To actually perform the communication, we need to co-locate the input and the output actions of MA. This cannot happen at top-level, otherwise it is easy to give examples where the encodings of parallel communications along different channels can mix up. Thus, we need to co-locate the encoding of the input and of the output within an ambient whose name must again be related to a (in particular, a_2). Notice that a_1 must be different from a_2: they serve different purposes (the former is used for checking the possibility of communicating along a, the latter for actually performing the communication) and so they must be different (otherwise an $open\_a_1$ could open the ambient where the communication happens, and this can create interferences between different communications). So, we place the encoding of the input within a_2 and the let encoding of the output enter in a_2 via the pilot ambient $\mathtt{p}$.

If we have different possible communications along the same channel in parallel, it is possible that the encodings of several outputs enter in the same copy of a_2, thus aim at synchronizing with the same input action. This is of course something that must be handled but, for the sake of presentation, let us consider for a while the scenario where no conflict for the same input arises. In this ideal setting, the encoding can be simpler:

$$
\begin{aligned}
[\![\bar{a}\langle b\rangle]\!] &\triangleq a_1[\mathtt{p}[in\_a_2 \mid \langle b_1, b_2\rangle]] \\
[\![a(x).P]\!] &\triangleq open\_a_1.(\nu r, s)(open\_s \mid r[\,] \\
&\qquad\qquad \mid a_2[open\_\mathtt{p}.(x_1, x_2).in\_r.s[out\_a_2.out\_r.[\![P]\!]]])
\end{aligned}
$$

The problem when different copies of $\mathtt{p}$ (say, n) enter into the same copy of a_2 is that at the end of the communication the unselected copies of $\mathtt{p}$ (and there are $n-1$) cannot be used anymore in other communications because they consumed their in_a_2 prefix and so cannot enter anymore into any other a_2. This would correspond to output particles of π_a that are not consumed in a communication but that are nomore available for communicating: this fact can be used to break operational soundness, by introducing computations that do not correspond to any π_a computation. This leads us to replicate the in_a_2 action in the encoding of an output. However, this creates another problem: if we now open $\mathtt{p}$, all its content, including $!in\_a_2$, will become part of a_2: thus, a_2 can enter within a sibling copy of a_2, and this again undermines operational soundness.

For this reason, we cannot select the output to consume by opening a $\mathtt{p}$; this fact forces us to select one $\mathtt{p}$ by letting another pilot ambient $\mathtt{q}$ enter into it. Now, $\mathtt{q}$ can be safely opened within $\mathtt{p}$ and the communication can happen. Now, we are left with three tasks: (1) get rid of $\mathtt{p}[!in\_a_2]$; (2) lead $[\![P\{^b/_x\}]\!]$ at top-level; and (3) unleash the possible copies of $\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b'_1, b'_2\rangle]$ entered into a_2 and not selected for communication. For (1), we lead $\mathtt{p}[!in\_a_2]$ into the restricted ambient r where it will remain for ever without affecting any other computation (actually, this is a junk that can be garbage collected). Tasks (2) and (3) will be carried out at once, by first leading $[\![P\{^b/_x\}]\!]$ within a_2, by then moving a_2 within the restricted ambient u where it will be opened and by finally opening u, thus unleashing at top-level $[\![P\{^b/_x\}]\!]$ and all the possible copies of $\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b'_1, b'_2\rangle]$. The presence of the restricted ambient t is justified by the fact that u must be opened after opening a_2; if u was at top-level, this fact would not be guaranteed.

Theorem 4.7. *The encoding of π_a into* MA *is valid, with operational correspondence that holds up-to strong barbed equivalence.*

The proof requires some ingenuity, mostly the part related to operational soundness and divergence reflection. It is instructive but quite technical; for this reason, it is relegated to the appendix and left to the interested reader.

4.3 Building the second level of the Taxonomy

4.3.1 SA is more expressive than MA

In [33] MA is translated into SA by mapping all the operators homomorphically, except for

$$[\![\, u[P] \,]\!] \;\triangleq\; u[\,!\,\overline{in}\_u \mid !\,\overline{out}\_u \mid !\,\overline{open}\_u \mid [\![\, P \,]\!]]$$

However, such an encoding enjoys all the properties listed in Section 3.1 only under some cautions. The problem is that the MA process $open\_n \mid n[\mathbf{0}]$ reduces to $\mathbf{0}$, whereas $[\![\, open\_n \mid n[\mathbf{0}] \,]\!]$ can only reduce to $!\,\overline{in}\_n \mid !\,\overline{out}\_n \mid !\,\overline{open}\_n$ and the latter process is *not* barbed equivalent to the encoding of $\mathbf{0}$ (viz., $\mathbf{0}$ itself): context $n[\cdot]$ can distinguish the two processes in SA. Hence, we have to accept a weaker formulation of operational correspondence, that only holds up to strong barbed equivalence restricted to translated contexts (written $\simeq^{tr}$).

Definition 4.3. *Given an encoding $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$, we say that two $\mathcal{L}_2$-processes T_1 and T_2 are* translated barbed equivalent*, written $T_1 \simeq^{tr} T_2$, if, for every $\mathcal{L}_1$-context $\mathcal{E}(\cdot)$, it holds that $[\![\mathcal{E}]\!](T_1) \stackrel{\bullet}{\simeq} [\![\mathcal{E}]\!](T_2)$.*

Proposition 4.8. *The encoding of* MA *in* SA *is valid, with operational correspondence that holds up-to translated barbed equivalence.*

Proof. It suffices to prove that $P_u \triangleq !\,\overline{in}\_u \mid !\,\overline{out}\_u \mid !\,\overline{open}\_u \simeq^{tr} \mathbf{0}$. To prove such an equality, we first notice that P_u behaves exactly as $P_u \mid P_u$ (this is one of Milner's replication laws [39] and can be easily proved). We now show that $C(P_u)$ and $C(\mathbf{0})$ are barbed bisimilar, whenever $C(\cdot)$ is a translated context. To this aim, we show that relation

$$\mathfrak{R} \;\triangleq\; \{(C(P_u), C(\mathbf{0})) \;:\; C(\cdot) \text{ is such that every ambient } u \text{ contains } P_u\}$$

is a barbed bisimulation. We distinguish whether the hole is immediately contained in an ambient u or not. In the first case, $C(\cdot)$ is of the form $\mathcal{D}(u[\cdot \mid P])$, for some context $\mathcal{D}(\cdot)$ and process P; by construction, $P \equiv P_u \mid P'$, for some P'. Hence, $u[P_u \mid P]$ behaves like $u[P]$; so, $C(P_u)$ and $C(\mathbf{0})$ are barbed bisimilar. If the hole is not immediately contained in an ambient u, then P_u does not contribute to the production of any barb nor to any reduction; thus, $C(P_u)$ exhibits a barb if and only if $C(\mathbf{0})$ exhibits a barb. Moreover, if $C(P_u) \longmapsto P'$, P' can only be $C'(P_u)$, for some $C'(\cdot)$ such that $C(\cdot) \longmapsto C'(\cdot)$; then, $C(\mathbf{0}) \longmapsto C'(\mathbf{0})$ and $(C'(P_u), C'(\mathbf{0})) \in \mathfrak{R}$, as desired. Indeed, for any possible reduction, every ambient u in $C'(\cdot)$ contains P_u, since $C(\cdot)$ satisfies this property, being a translated context. □

Thus, we have proved that the encoding in [33] is valid only when $\simeq_2$ is translated barbed equivalence. This result is not as strong as an analogous one proved under (general) barbed equivalence, but it is enjoys the same properties as the encoding of separated choice into π_a in [43].

We now prove that SA cannot be encoded in MA; in particular, the effect of SA's co-actions cannot be properly rendered in MA.

Theorem 4.9. *There exists no valid encoding of* SA *in* MA.

Proof. By contradiction. Consider the pair of SA processes $P \triangleq m[in\_n]$ and $Q \triangleq n[\overline{in}\_n.\overline{open}\_n] \mid open\_n.\surd$, for $n \neq m$; by Proposition 3.2, $[\![P \mid Q]\!]$ must reduce and, because of Propositions 3.3 and 4.1, it can only be

1. either $C_1([\![P]\!]) \xrightarrow{amb\_n'}$ and $C_2([\![Q]\!]) \xrightarrow{\mu}$, for $\mu \in \{enter\_n', open\_n'\}$
2. or $C_2([\![Q]\!]) \xrightarrow{amb\_n'}$ and $C_1([\![P]\!]) \xrightarrow{\mu}$, for $\mu \in \{enter\_n', open\_n'\}$.

for some context $C_1(\cdot)$ and $C_2(\cdot)$ that are empty or have a single top-level ambient containing a top-level hole. Notice that the reduction cannot happen because of a communication, say $[\![P]\!] \xrightarrow{\langle - \rangle}$ and $[\![Q]\!] \xrightarrow{(-)}$, otherwise, by Property 2, $[\![n[in\_m] \mid Q]\!]$ would reduce, against Proposition 3.1. For the same reason, it must be that $n' \in \varphi_{[\![\ ]\!]}(n)$.

We now prove that both cases are impossible and assume that we fall in case 1 (case 2 is similar). First, notice that $C_1(\cdot)$ must be empty: if it was not, we would have that $[\![m[out\_n] \mid Q]\!] \longmapsto$ (recall that $C_1(\cdot)$ is part of $C_{|}^{\{n,m\}}(\_1;\_2)$, the context used to encode parallel composition of processes with free names $\{n, m\}$; so, it only depends on parallel composition and on such names). Thus, we have that $[\![P]\!] \xrightarrow{amb\_n'}$; but also this leads to a contradiction. Indeed, by Property 1, it holds that $[\![P]\!] \triangleq C_{m[\ ]}^{\{n\}}([\![in\_n]\!])$; so, the ambient named n' can be exhibited either by $C_{m[\ ]}^{\{n\}}(\cdot)$ or by $[\![in\_n]\!]$. In both cases, we can contradict Proposition 3.1: in the first case, we would have that $[\![m[out\_n]]\!] \xrightarrow{amb\_n'}$ and so $[\![m[out\_n] \mid Q]\!] \longmapsto$; in the second case, we would have that $[\![in\_n \mid Q]\!] \longmapsto$. □

4.3.2 SBA **is more expressive than** BA

It is easy to prove that SBA can encode BA: it suffices to translate every operator homomorphically, except for

$$[\![u[P]]\!] \triangleq\ !\,\overline{out}\_* \mid u[!\,\overline{in}\_* \mid [\![P]\!]] \qquad\qquad [\![\mathbf{0}]\!] \triangleq\ !\,\overline{out}\_*$$

This encoding enjoys operational correspondence up-to strong barbed equivalence, thanks to Milner's replication law $!P \simeq !P \mid !P$.

The fact that SBA cannot be encoded in BA is a corollary of Theorem 3.5, since $\text{MD}(\text{SBA}) = 2$ and $\text{MD}(\text{BA}) = 1$.

4.3.3 NBA **is more expressive than** BA_s

NBA can encode BA_s: it suffices to translate every operator homomorphically, except for

$$[\![\mathbf{0}]\!] \triangleq\ !\,\overline{out}\_(x, \mathtt{p}) \qquad [\![u[P]]\!] \triangleq\ !\,\overline{out}\_(x, \mathtt{p}) \mid u[!\,\overline{in}\_(x, \mathtt{p}) \mid [\![P]\!]]$$

$$[\![in\_u]\!] \triangleq\ in\_(u', \mathtt{p}) \qquad [\![out\_u]\!] \triangleq\ out\_(u', \mathtt{p}) \qquad [\![u]\!] \triangleq\ u'$$

for some fixed (constant) password $\mathtt{p}$. We let u' denote $\varphi_{[\![\ ]\!]}(u)$.

The fact that NBA cannot be encoded in BA_s is a corollary of Theorem 3.5, since $\text{MD}(\text{NBA}) = 2$ and $\text{MD}(\text{BA}_s) = 1$.

4.3.4 SAP, SBA and NBA are more expressive than Dπ

Dπ cannot encode SAP/SBA/NBA, as proved in the following result.

Theorem 4.10. *There exists no valid encoding of* SAP*,* SBA *and* NBA *in* Dπ.

Proof. This is a Corollary of Theorem 3.4. Indeed, we can find in SAP, SBA and NBA a process S such that $S \mapsto\!\!\!\!\!\rightarrow_1$, $S \searrow\!\!\!\!\!\searrow_1$ and $S \mid S \searrow_1$: it suffices to let S be

- in SAP: $n[in\_(n,n) \mid \overline{in}\_(n,n).\overline{open}\_(n,n)] \mid open\_(n,n).\surd$;
- in SBA: $n[in\_(n,n) \mid \overline{in}\_(n,n).\langle n\rangle^{\uparrow}] \mid (x)^{\star}.\surd$;
- in NBA: $n[in\_(n,n) \mid \overline{in}\_(x,n).\langle n\rangle^{\uparrow}] \mid (x)^{n}.\surd$. □

On the contrary, Dπ can be encoded both in SAP, in SBA and in NBA. Indeed, thanks to passwords, SAP, SBA and NBA can all atomically match two names: the name of the channel where the Dπ processes communicate and the locality hosting them. In SAP the encoding is slightly more tedious than in SBA and in NBA, where it is simpler, thanks to remote communications.

For SAP, the main idea is that an output over channel u located at w is represented as an occurrence of ambient w that can be entered by a pilot ambient $\mathtt{p}$ by using u as password; once entered, the pilot ambient must be opened, the communication takes place locally and the continuation processes are activated (notice that the continuation of the output must be activated after consumption of the output message; this is the aim of the synchronizing ambient $\mathtt{go}$). Formally, the encoding acts homomorphically on all the operators, except for

$$
\begin{array}{rcll}
[\![\, l : P \,]\!] & \triangleq & [\![\, P \,]\!]_{l'} \mid\, !\, l'[\overline{open}\_(l', l')] & \\
[\![\, \bar{u}\langle v\rangle.P \,]\!]_w & \triangleq & (\nu k)(w[\overline{in}\_(w,u').open\_(\mathtt{p},\mathtt{p}). & \\
 & & \quad (\langle v'\rangle \mid \mathtt{go}[\overline{open}\_(\mathtt{go},\mathtt{go}).\overline{open}\_(w,k)])\,] & \text{for } k \text{ fresh} \\
 & & \quad \mid open\_(w,k).[\![\, P \,]\!]_w) & \\
[\![\, u(x).P \,]\!]_w & \triangleq & \mathtt{p}[in\_(w,u').\overline{open}\_(\mathtt{p},\mathtt{p}).(x').open\_(\mathtt{go},\mathtt{go}).[\![\, P \,]\!]_w] & \\
[\![\, go\_u.P \,]\!]_w & \triangleq & open\_(u',u').[\![\, P \,]\!]_{u'} &
\end{array}
$$

where $\mathtt{p}$ and $\mathtt{go}$ are reserved names and l', u', v' and x' are the renamings of l, u, v and x, respectively.

This encoding is valid. Most of the properties are easy to prove. We now just formally prove operational correspondence and divergence sensitiveness. Just notice that, in the encoding of a go_l, existence of the target locality l is checked by opening an occurrence of an ambient named l'; such ambients are replicated since localities in Dπ cannot disappear along computations and duplicates of the same locality behave like a single copy of that locality (see the srucutural law $l : P|Q \equiv l : P \mid l : Q$).

Proposition 4.11. *The encoding of* Dπ *in* SAP *is valid.*

Proof. Let us start with operational completeness. It suffices to consider a single reduction of Dπ, since sequences of reductions can be obtained by straightforward inductive arguments. Let $S \longmapsto S'$; the proof is by induction on the inference of this reduction. The inductive step is easy. We only discuss the base step (this will be needed for proving the remaining properties). There are two possible base cases:

one for $S \triangleq l : go\_k.P \mid k : \mathbf{0}$ and one for $S \triangleq l : \bar{a}\langle b\rangle.P \mid l : a(x).Q$. The first case is trivial, since $[\![S]\!] \longmapsto [\![S']\!]$. For the second case, it suffices to spell out the reductions:

$$
\begin{aligned}
[\![S]\!] \;\longmapsto T_0 &= (\nu k)(l'[\,\mathsf{p}[\overline{open}\_(\mathsf{p},\mathsf{p}).(x').open\_(\mathsf{go},\mathsf{go}).[\![Q]\!]_{l'}] \\
&\qquad\qquad \mid open\_(\mathsf{p},\mathsf{p}).(\langle b'\rangle \mid \mathsf{go}[\overline{open}\_(\mathsf{go},\mathsf{go}).\overline{open}\_(l',k)])\,] \\
&\qquad\quad \mid open\_(l',k).[\![P]\!]_{l'}) \\
&\quad \mid\, !\, l'[\overline{open}\_(l',l')] \mid\, !\, l'[\overline{open}\_(l',l')] \\
\longmapsto T_1 &= (\nu k)(l'[\,(x').open\_(\mathsf{go},\mathsf{go}).[\![Q]\!]_{l'} \mid \langle b'\rangle \mid \mathsf{go}[\overline{open}\_(\mathsf{go},\mathsf{go}).\overline{open}\_(l',k)]\,] \\
&\qquad\quad \mid open\_(l',k).[\![P]\!]_{l'}) \\
&\quad \mid\, !\, l'[\overline{open}\_(l',l')] \mid\, !\, l'[\overline{open}\_(l',l')] \\
\longmapsto T_2 &= (\nu k)(l'[\,open\_(\mathsf{go},\mathsf{go}).[\![Q\{^b/_x\}]\!]_{l'} \mid \mathsf{go}[\overline{open}\_(\mathsf{go},\mathsf{go}).\overline{open}\_(l',k)]\,] \\
&\qquad\quad \mid open\_(l',k).[\![P]\!]_{l'}) \\
&\quad \mid\, !\, l'[\overline{open}\_(l',l')] \mid\, !\, l'[\overline{open}\_(l',l')] \\
\longmapsto T_3 &= (\nu k)(l'[[\![Q\{^b/_x\}]\!]_{l'} \mid \overline{open}\_(l',k)] \mid open\_(l',k).[\![P]\!]_{l'}) \\
&\quad \mid\, !\, l'[\overline{open}\_(l',l')] \mid\, !\, l'[\overline{open}\_(l',l')] \\
\longmapsto T_4 &= [\![l : Q\{^b/_x\}]\!] \mid [\![l : P]\!] \triangleq [\![S']\!]
\end{aligned}
$$

Notice also that $T_0/\ldots/T_3$ (that we call *intermediate states*) can only interact with an outer context by letting one of their replicated ambients be opened. Thanks to replication, this interaction does not change the process at all.

Let us give names to the reductions in an encoded term: the reduction of the encoding of a *go* and a reduction that creates a T_0 are called *of kind* 0; a reduction that turns a T_i into a T_{i+1} is called *of kind* $i+1$. Thus, any reduction of kind 0 corresponds to a reduction in the source term. Moreover, it is easy to see that in any computation of an encoded term the number of reductions of kind i cannot be smaller than the number of reductions of kind $i+1$, for every $i \in \{0,\ldots,3\}$. This fact easily entails that the encoding cannot introduce divergence: if $[\![S]\!]$ diverges, it must produce infinitely many reductions of kind 0; thus, S can produce infinitely many reductions, i.e. it diverges.

We are left with operational soundness. Let $[\![S]\!] \longmapsto^n T$ and let n_i be the number of reductions of kind i in this computation. Since the intermediate states cannot significantly interact with any other process, not even among themselves, it is easy to prove that $S \longmapsto^{n_0} S'$ (by executing the n_0 reductions associated to the reductions of kind 0 in the encoding) and that $T \longmapsto^{4n'_0-n_1-n_2-n_3-n_4} [\![S']\!]$, where n'_0 is the number of reductions of kind 0 that have not not been originated by the encoding of a *go*. Indeed, it suffices to turn every T_0 (and there are $n'_0 - n_1$ copies of it in T) in an encoding; this requires 4 reductions for every copy of T_0. Then we need to turn every T_1 (and there are $n_1 - n_2$ copies of it in T) in an encoding; this requires 3 reductions for every copy of T_1. And so on. In total, we need $4(n'_0 - n_1) + 3(n_1 - n_2) + 2(n_2 - n_3) + (n_3 - n_4) = 4n'_0 - n_1 - n_2 - n_3 - n_4$ reductions and in this way we exactly obtain the encoding of S'. □

Let us now quickly give the encoding of Dπ in SBA and NBA. In the first case, it suffices to let

$$
\begin{aligned}
&[\![\bar{u}\langle v\rangle.P]\!]_w \triangleq (\nu k)(w[\overline{in}\_u.(y)^u.(y[out\_w.\langle v\rangle^\star] \mid k[out\_w])] \mid \overline{out}\_k.[\![P]\!]_w) && \text{for } k \text{ and } y \text{ fresh} \\
&[\![u(x).P]\!]_w \triangleq (\nu h)(u[in\_w.\langle h\rangle^\star] \mid \overline{out}\_h.(x)^h.[\![P]\!]_w) && \text{for } h \text{ fresh}
\end{aligned}
$$

In the second case, the encoding is very similar; just notice that, thanks to the dynamic learning of the

name of a moving ambient, the first communication becomes useless.

$$[\![\, \bar{u}\langle v\rangle.P \,]\!]_w \triangleq (\nu k)(w[\overline{in}\_(y,u).(y[out\_(w,y).\langle v\rangle^{\uparrow}] \mid k[out\_(w,k)])] \mid \overline{out}\_(w,k).[\![\, P \,]\!]_w) \quad \text{for } y,k \text{ fresh}$$

$$[\![\, u(x).P \,]\!]_w \triangleq (\nu h)(h[in\_(w,u)] \mid \overline{out}\_(y,h).(x)^h.[\![\, P \,]\!]_w) \quad \text{for } y,h \text{ fresh}$$

The fact that these encodings are valid can be proved in the same way as in Proposition 4.11 and is left to the reader.

4.4 Further Impossibility Results

In this section, we prove some separation results that will allow us to complete the taxonomy in Figure 1. We start by showing that the remote communications typical of BA and BA$_s$ cannot be properly rendered in SA and SAP, not even by exploiting the combination of *open* and local communications.

Theorem 4.12. *There exists no valid encoding of* BA *and* BA$_s$ *in* SA *and* SAP.

Proof. Let us first consider the non-encodability of BA in SAP. Consider the processes $(x)^n.\surd$ and $n[\langle b\rangle^\star]$, for $n \neq b$. Because of Proposition 3.2, $[\![\, (x)^n.\surd \mid n[\langle b\rangle^\star] \,]\!]$ must reduce and, because of Propositions 3.3 and 4.2, this can only happen because:

1. either $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{enter\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{?enter\_(h,k)}$
2. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{?enter\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{enter\_(h,k)}$
3. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{open\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{?open\_(h,k)}$
4. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{?open\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{open\_(h,k)}$.
5. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{exit\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{?exit\_(h,k)}$
6. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{?exit\_(h,k)}$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{exit\_(h,k)}$

Indeed, $C_1([\![\, (x)^n.\surd \,]\!])$ and $C_2([\![\, n[\langle b\rangle^\star] \,]\!])$ cannot perform a communication, otherwise, by Property 2, $[\![\, (x)^n.\surd \mid b[\langle n\rangle^\star] \,]\!]$ would reduce; for the same reason, it must be that $\{h,k\} \cap \varphi_{[\![\ ]\!]}(n) \neq \emptyset$.

However, we now prove that all the cases depicted above lead to contradict Proposition 3.1. Let $C_2([\![\, n[\langle b\rangle^\star] \,]\!]) \xrightarrow{\mu}$, for $\mu \in \{?enter\_(h,k), enter\_(h,k), ?open\_(h,k), open\_(h,k), ?exit\_(h,k), exit\_(h,k)\}$. If $C_2(\cdot)$ is empty we can work as follows. First, observe that $[\![\, n[\langle b\rangle^\star] \,]\!] \triangleq C^{\{b\}}_{n[\ ]}([\![\, \langle b\rangle^\star \,]\!])$. If μ is produced by $C^{\{b\}}_{n[\ ]}(\cdot)$, also $[\![\, n[in\_b] \,]\!]$ would exhibit label μ; thus, $[\![\, (x)^n.\surd \mid n[in\_b] \,]\!] \longmapsto$, in contradiction with Proposition 3.1. If the production of μ involves $[\![\, \langle b\rangle^\star \,]\!]$, we would have that $\{h,k\} \subseteq fn([\![\, \langle b\rangle^\star \,]\!])$, in contradiction with Proposition 3.6. So, assume that $C_2(\cdot)$ is not empty; this rules out cases 4 and 5 above, and imposes that $[\![\, n[\langle b\rangle^\star] \,]\!] \xrightarrow{\mu'}$, for $\mu' \in \{\overline{in}\_(h,k), in\_(h,k), \overline{open}\_(h,k), out\_(h,k)\}$. We then work like in the case in which $C_2(\cdot)$ is empty to prove that there is no way for $[\![\, n[\langle b\rangle^\star] \,]\!]$ to produce μ' without contradicting Propositions 3.1 and 3.6.

The non-encodability of BA in SA can be proved in a similar way, but cases 5 and 6 are not possible and no password is involved. For BA$_s$ the proof is formally identical, once replaced $\langle b\rangle^\star$ with $\langle b\rangle^{\uparrow}$ within ambient n. □

We now show that the power of MA's *open* primitive cannot be rendered in BA.

Theorem 4.13. *There exists no valid encoding of* MA *in* BA.

Proof. By contradiction. Consider the MA process $open\_n.\langle m\rangle \mid n[\surd \mid P]$, for any P and $n \neq m$; by Propositions 3.2 and 3.3, it must be that $C_1([\![open\_n.\langle m\rangle]\!]) \xrightarrow{\mu}$ and $C_2([\![n[\surd \mid P]]\!]) \xrightarrow{\mu'}$ where, by Proposition 4.3, it can only be that

1. $\mu = \langle - \rangle^{n'}$ and $\mu' = n'(-)$;
2. $\mu = n'(-)$ and $\mu' = \langle - \rangle^{n'}$;
3. $\mu = (-)^{n'}$ and $\mu' = n'\langle - \rangle$;
4. $\mu = n'\langle - \rangle$ and $\mu' = (-)^{n'}$;
5. $\mu = enter\_n'$ and $\mu' = amb\_n'$;
6. $\mu = amb\_n'$ and $\mu' = enter\_n'$.

In all cases, $n' \in \varphi_{[\![\ ]\!]}(n)$, otherwise $[\![open\_m.\langle n\rangle \mid n[\surd \mid P]]\!] \longmapsto$. Let us consider the cases in isolation.

1. In this case, $C_1(\cdot)$ must be empty, so $[\![open\_n.\langle m\rangle]\!] \xrightarrow{\mu}$. If $C_2(\cdot)$ is empty, then it must be that $C^{fn(P)}_{n[\ ]}(\cdot)$ must either be of the form $(\nu\widetilde{m})(n'[C(\cdot) \mid (x)^*.Q_1 \mid Q_2] \mid Q_3)$ or $(\nu\widetilde{m})(n'[(x)^*.Q_1 \mid Q_2] \mid n''[C(\cdot) \mid Q_3] \mid Q_4)$ or $(\nu\widetilde{m})(n'[\,\cdot \mid Q_1] \mid Q_2)$ with $[\![\surd \mid P]\!] \xrightarrow{(-)^\star}$. Indeed, the hole cannot be at top-level, otherwise $[\![n[open\_n.\langle m\rangle]]\!]$ would reduce. Moreover, in the first two cases, $C(\cdot)$ must have the hole at top-level. Indeed, $[\![(open\_m \mid n[in\_n.m[out\_n.out\_n.\surd]]) \mid n[P]]\!]$ must reduce and the reduction should be originated also with the contribution of $[\![in\_n.m[out\_n.out\_n.\surd]]\!]$. If the hole in $C^{fn(P)}_{n[\ ]}(\cdot)$ was at a nesting depth greater than 1, this would not be possible and, hence, also $[\![(open\_m \mid n[out\_n.m[out\_n.out\_n.\surd]]) \mid n[P]]\!]$ would reduce. However, having the hole at nesting depth 1 makes it impossible to move $[\![\surd \mid P]\!]$ out from its enclosing ambient (and this must happen because of operational completeness). Indeed, ambients contained in n'/n'' can somehow be forced to exit from n'/n''; sequences of actions can be transmitted via a child-to-parent communication; but it turns out impossible to move processes like $(x)^*.(\langle m\rangle^n \mid out\_k.h[\ldots])$. Notice that such processes can appear in an encoded term, and so in $[\![P]\!]$: as we have seen, local inputs, downward outputs, *out* actions and ambients are all needed to encode process $open\_n.\langle m\rangle \mid n[\surd \mid P]$.

   If $C_2(\cdot)$ is not empty, by Proposition 3.3 it must be an ambient containing a top-level hole, and the ambient can only be named n' (otherwise, action μ' could not have been produced). Thus, $[\![n[\surd \mid P]]\!]$ must exhibit a top-level local input; because of Property 2, also $[\![m[\surd \mid P']]\!]$ exhibits a top-level local input (where P' is P with n and m swapped). But this would imply that $[\![open\_n.\langle m\rangle \mid m[\surd \mid P']]\!] \longmapsto$, in contradiction with Proposition 3.1.

2. In this case, $C_2(\cdot)$ must be empty and it must be that $C^{fn(P)}_{n[\ ]}(\cdot) \xrightarrow{\mu'}$; indeed, by Proposition 3.6, it cannot be that $[\![\surd \mid P]\!] \xrightarrow{\mu'}$ (to see this, it suffices to let P be a process whose free names do not

contain n, e.g. $\mathbf{0}$). Moreover, it must be that the hole of $C^{fn(P)}_{n[\,]}(\cdot)$ is not underneath the output prefix that originates μ', otherwise the reduction of $[\![\,(open\_m \mid n[in\_n.m[out\_n.out\_n.\surd]]) \mid n[P]\,]\!]$ can only be originated without the contribution of $[\![\,in\_n.m[out\_n.out\_n.\surd]\,]\!]$. Hence, $C^{fn(P)}_{n[\,]}(\cdot)$ must be of the form $(\nu\widetilde{m})(\langle M\rangle^{n'}.Q \mid C(\cdot))$, where $C(\cdot)$ cannot have the hole at top-level otherwise $[\![\,n[open\_n.\langle m\rangle]\,]\!]$ would reduce. However, it cannot either have the hole at nesting depth greater than 1 otherwise the reduction of $[\![\,(open\_m \mid n[in\_n.m[out\_n.out\_n.\surd]]) \mid n[P]\,]\!]$ can only be originated without the contribution of $[\![\,in\_n.m[out\_n.out\_n.\surd]\,]\!]$. Then, like in case 1, we are in a situation where we cannot move $[\![\,\surd \mid P\,]\!]$ out from its enclosing ambient; this suffices to conclude.

3. Similar to case 1.

4. Similar to case 2.

5. Similar to case 2.

6. If $C_2(\cdot)$ is empty, the case is similar to case 1; just notice that here it cannot be that $C^{fn(P)}_{n[\,]}(\cdot)$ is of the form $(\nu\widetilde{m})(k[\,\cdot \mid Q_1] \mid Q_2)$, since $[\![\,\surd \mid P\,]\!]$ cannot perform action in_n' whenever $n \notin fn(P)$, see Proposition 3.6. If $C_2(\cdot)$ is not empty, the case is more delicate: indeed, nothing prevents $C^{fn(P)}_{n[\,]}(\cdot)$ to perform action in_n'. However, we can reason as follows. By compositionality, $[\![\,!\,n[\surd \mid P]\,]\!] = C^{\{n\}\cup fn(P)}_{!}([\![\,n[\surd \mid P]\,]\!])$. The hole of $C^{\{n\}\cup fn(P)}_{!}(\cdot)$ must be at top-level, otherwise $[\![\,open\_n.\langle m\rangle \mid !\,open\_n.P\,]\!] \longmapsto$. Moreover, the hole cannot be replicated, otherwise $[\![\,open\_n.\langle m\rangle \mid !\,n[\surd \mid P]\,]\!] \longmapsto^{\omega}$, against Property 4 whenever P does not diverge. Hence, $C^{\{n\}\cup fn(P)}_{!}(\cdot)$ must be of the form $(\nu\widetilde{m})(\,\cdot \mid Q)$.

   Let us now consider $[\![\,!\,n[in\_n.P]\,]\!]$ that must diverge (since $!\,n[in\_n.P]$ diverges) and, hence, reduces. By what we have just proved, $[\![\,!\,n[in\_n.P]\,]\!] \equiv (\nu\widetilde{m})([\![\,n[in\_n.P]\,]\!] \mid Q)$. Since neither $[\![\,n[in\_n.P]\,]\!]$ nor Q can reduce in isolation (without contradicting Proposition 3.1), it must be that $[\![\,n[in\_n.P]\,]\!]$ and Q interact: $[\![\,n[in\_n.P]\,]\!] \xrightarrow{\mu}$ and $Q \xrightarrow{\bar{\mu}}$, for some μ produced by the contribution of both $C^{\{n\}\cup fn(P)}_{n[\,]}(\cdot)$ and $[\![\,in\_n.P\,]\!]$; hence, $\mu \in \{n''(-), n''\langle-\rangle, enter\_n''\}$ and $C^{\{n\}\cup fn(P)}_{n[\,]}(\cdot)$ has the hole at nesting depth 1 (i.e., occurring at top-level within some ambient).

   - If $\mu \in \{n''(-), n''\langle-\rangle\}$, then $C^{\{n\}\cup fn(P)}_{n[\,]}(\cdot)$ is of the form $(\nu\widetilde{p})(n''[\,\cdot \mid Q_1] \mid Q_2)$ and $[\![\,in\_n.P\,]\!] \xrightarrow{\mu'}$, for $\mu' \in \{(-)^{\star}, \langle-\rangle^{\star}\}$. Let P be such that $\{b, n\} \subseteq fn(P)$, and let P' be P with n and b swapped; then, $[\![\,!\,n[in\_b.P']\,]\!] \equiv (\nu\widetilde{m}, \widetilde{p})(n''[[\![\,in\_b.P'\,]\!] \mid Q_1] \mid Q_2 \mid Q) \longmapsto$: indeed, $n''[[\![\,in\_b.P\,]\!] \mid Q_1] \xrightarrow{\mu}$ and $Q \xrightarrow{\bar{\mu}}$. This fact contradicts Proposition 3.1, since $!\,n[in\_b.P'] \mapsto\!\!\!\!\!\!\rightarrow$.
   - If $\mu = enter\_n''$, we choose P such that $\{b, n\} \subseteq fn(P)$ and consider $!\,b[in\_b.P']$, where P' is P with n and b swapped. By Property 2, we have that $C^{fn(P)}_{b[\,]}(\cdot)$ has the hole at nesting depth 1: say, $C^{fn(P)}_{b[\,]}(\cdot)$ is of the form $(\nu\widetilde{p})(p[\,\cdot \mid Q_1] \mid Q_2)$. Thus, $[\![\,!\,b[in\_n.P]\,]\!] \equiv (\nu\widetilde{m}, \widetilde{p})(p[[\![\,in\_n.P\,]\!] \mid Q_1] \mid Q_2 \mid Q) \longmapsto$: indeed, $p[[\![\,in\_n.P\,]\!] \mid Q_1] \xrightarrow{\mu}$ and $Q \xrightarrow{\bar{\mu}}$. This fact contradicts Proposition 3.1, since $!\,b[in\_n.P] \mapsto\!\!\!\!\!\!\rightarrow$. □

We now prove that the different position of the $\overline{out}$ primitive in SA and SAP makes a valid encoding of the former into the latter impossible. The converse is a trivial consequence of Theorem 3.5;

nevertheless, in [23] we have proved that also the different position of the $\overline{out}$ action would make the converse encoding impossible. In particular, we defined SA_p to be SA with passwords (hence, with the $\overline{out}$ in the ambient left) and proved that SAP and SA_p to be incomparable. Here we prefer avoiding the introduction of SA_p because it is an ad-hoc calculus, never appeared in the literature.

Theorem 4.14. *There exists no valid encoding of* SA *in* SAP.

Proof. Consider $P \triangleq m[n[out\_m.\overline{open}\_n] \mid \overline{out}\_m]$ and $Q \triangleq open\_n.\surd$, for $n \neq m$. By Proposition 3.2, we know that $[\![P \mid Q]\!] \longmapsto$; if $[\![P]\!] \not\longmapsto$, then, by Propositions 3.3 and 4.2, $[\![P \mid Q]\!] \longmapsto$ can be produced in six ways:

1. $C_1([\![P]\!]) \xrightarrow{enter\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{?enter\_(h,k)}$;
2. $C_1([\![P]\!]) \xrightarrow{?enter\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{enter\_(h,k)}$;
3. $C_1([\![P]\!]) \xrightarrow{open\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{?open\_(h,k)}$;
4. $C_1([\![P]\!]) \xrightarrow{?open\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{open\_(h,k)}$;
5. $C_1([\![P]\!]) \xrightarrow{exit\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{?exit\_(h,k)}$;
6. $C_1([\![P]\!]) \xrightarrow{?exit\_(h,k)}$ and $C_2([\![Q]\!]) \xrightarrow{exit\_(h,k)}$.

We now prove that all these cases are not possible. In cases 3 and 6, it must be that $C_1(\cdot)$ is empty; so, $[\![P]\!] \xrightarrow{\mu}$, for $\mu \in \{open\_(h,k), ?exit\_(h,k)\}$. By Property 1, $[\![P]\!] \triangleq C^{\{n,m\}}_{m[\,]}([\![P']\!])$, where $P' \triangleq n[out\_m.\overline{open}\_n] \mid \overline{out}\_m$. However, it cannot be that $C^{\{n,m\}}_{m[\,]}(\cdot) \xrightarrow{\mu}$, otherwise $[\![m[in\_n.in\_m] \mid Q]\!] \longmapsto$, nor that $[\![P']\!] \xrightarrow{\mu}$, otherwise $[\![m[out\_m.\overline{open}\_n \mid \overline{out}\_m] \mid Q]\!] \longmapsto$; thus, cases 3 and 6 are impossible.

In the remaining cases, we can work as follows. First, suppose that $C_1(\cdot)$ is not empty; thus, $C_1(\cdot) \triangleq a[\,\cdot \mid R]$ and $[\![P]\!] \xrightarrow{\mu'}$, for $\mu' \in \{in\_(h,k), \overline{in}\_(h,k), \overline{open}\_(h,k), out\_(h,k)\}$. Like before, we can prove that there is no way for $[\![P]\!]$ to perform μ' without contradicting Proposition 3.1. Hence, it must be that $C_1(\cdot)$ is empty. Again, $[\![P]\!] \triangleq C^{\{n,m\}}_{m[\,]}([\![P']\!]) \xrightarrow{\mu}$, for $\mu \in \{enter\_(h,k), ?enter\_(h,k), ?open\_(h,k), exit\_(h,k)\}$. If $C^{\{n,m\}}_{m[\,]}(\cdot) \xrightarrow{\mu}$ or $[\![P']\!] \xrightarrow{\mu}$, we can work like for cases 3 and 6 above. So, it must be that $C^{\{n,m\}}_{m[\,]}(\cdot)$ is of the form $(\nu\widetilde{p})(a[\,\cdot \mid R_1] \mid R_2)$ and $[\![P']\!] \xrightarrow{\mu'}$, for $\mu' \in \{in\_(h,k), \overline{in}\_(h,k), \overline{open}\_(h,k), out\_(h,k)\}$. Furthermore, by Property 1, $[\![P']\!] \triangleq C^{\{n,m\}}_{|}([\![n[out\_m.\overline{open}\_n]]\!]; [\![\overline{out}\_m]\!])$; thus, $[\![P']\!] \xrightarrow{\mu'}$ can happen in three ways:

- $C^{\{n,m\}}_{|}(\cdot) \xrightarrow{\mu'}$;
- $[\![n[out\_m.\overline{open}\_n]]\!] \xrightarrow{\mu'}$;
- $[\![\overline{out}\_m]\!] \xrightarrow{\mu'}$.

All these cases lead to contradict Proposition 3.1: in the first two cases, it is easy to prove that also $[\![\, m[n[out\_m.\overline{open}\_n] \mid \overline{in}\_m] \mid Q \,]\!] \longmapsto$; in the third case, we would have that $[\![\, m[out\_m.\overline{open}\_n \mid \overline{out}\_m] \mid Q \,]\!] \longmapsto$.

Let us then consider the case in which $[\![\, P \,]\!] \longmapsto$. By Property 1, $[\![\, P \,]\!] \triangleq C^{\{m,n\}}_{m[\ ]}([\![\, n[out\_m.\overline{open}\_n] \mid \overline{out}\_m \,]\!])$. Of course, the reduction of $[\![\, P \,]\!]$ cannot be generated by $C^{\{m,n\}}_{m[\ ]}(\cdot)$ nor by $[\![\, n[out\_m.\overline{open}\_n] \mid \overline{out}\_m \,]\!]$ in isolation. Thus, it must be that $C^{\{m,n\}}_{m[\ ]}(\cdot)$ is of the form $\mathcal{E}(\ \cdot \mid R)$, with $[\![\, n[out\_m.\overline{open}\_n] \mid \overline{out}\_m \,]\!] \xrightarrow{\mu}$ and $R \xrightarrow{\bar{\mu}}$. Again by Property 1, $[\![\, n[out\_m.\overline{open}\_n] \mid \overline{out}\_m \,]\!] \triangleq C^{\{m,n\}}_{|}([\![\, n[out\_m.\overline{open}\_n] \,]\!] \,;\, [\![\, \overline{out}\_m \,]\!])$. Thus, at least one between $[\![\, n[out\_m.\overline{open}\_n] \,]\!]$ and $[\![\, \overline{out}\_m \,]\!]$ is excluded from the production of μ. This fact can be used to contradict Proposition 3.1: if $[\![\, n[out\_m.\overline{open}\_n] \,]\!]$ is excluded, we would have that $[\![\, m[m[out\_n.\overline{open}\_m]] \mid \overline{out}\_m \,]\!] \longmapsto$; if $[\![\, \overline{out}\_m \,]\!]$ is excluded, we would have that $[\![\, m[n[out\_m.\overline{open}\_n]] \mid \overline{open}\_m \,]\!] \longmapsto$. □

We now show that the shared and the localized versions of Boxed ambients are incomparable. This means that shared channels cannot be properly simulated via localized ones (there is no way in BA to restrict access to a channel by allowing, e.g., access to a parent process and not to a co-located process), nor vice versa (there is no way in BA_s to render the more liberal use of channels put forward by the localized approach).

Theorem 4.15. BA_s *and* BA *are incomparable.*

Proof. We start with the non-encodability of BA in BA_s. Consider the following pair of BA processes: $P_1 \triangleq (x)^n.(b[\mathbf{0}] \mid \surd)$ and $P_2 \triangleq n[\langle b \rangle^\star]$. By Proposition 3.2, $[\![\, P_1 \mid P_2 \,]\!]$ must reduce; this can only happen in three possible ways:

a) $C_i([\![\, P_i \,]\!]) \xrightarrow{enter\_n'}$ and $C_j([\![\, P_j \,]\!]) \xrightarrow{amb\_n'}$, for $\{i, j\} = \{1, 2\}$;

b) $C_i([\![\, P_i \,]\!]) \xrightarrow{\langle - \rangle^{n'}}$ and $C_j([\![\, P_j \,]\!]) \xrightarrow{n'(-)}$, for $\{i, j\} = \{1, 2\}$;

c) $C_i([\![\, P_i \,]\!]) \xrightarrow{(-)^{n'}}$ and $C_j([\![\, P_j \,]\!]) \xrightarrow{n'\langle - \rangle}$, for $\{i, j\} = \{1, 2\}$.

Indeed, by Property 2, no other form of interaction can take place; moreover, it must be that $n' = \varphi_{[\![\ ]\!]}(n)$. We now prove that only cases (b) and (c) with $i = 1$ and $j = 2$ do not contradict Proposition 3.1.

- Concerning case (a), if $i = 1$ and $j = 2$, then $C_2(\cdot)$ must be empty and either $C^{\{b\}}_n(\cdot) \xrightarrow{amb\_n'}$ or $[\![\, \langle b \rangle^\star \,]\!] \xrightarrow{amb\_n'}$; the former could be used to contradict Proposition 3.1, the latter Proposition 3.6. If $j = 1$ and $i = 2$, we can prove that both $C_1(\cdot)$ and $C_2(\cdot)$ must be empty, that $C^{\{b\}}_{(x)^{n'}}(\cdot) \xrightarrow{amb\_n'}$, $C^{\{b\}}_{n[\ ]}(\cdot)$ is of the form $(\nu \widetilde{h})(h[\ \cdot \mid Q_1] \mid Q_2)$ and $[\![\, b[\mathbf{0}] \mid \surd \,]\!] \xrightarrow{in\_n'}$; but the latter fact is not possible, thanks to Proposition 3.6.

- Concerning cases (b) and (c), with $i = 2$ and $j = 1$, we have that μ (that is $\langle - \rangle^{n'}$ in case (b) and $(-)^{n'}$ in case (c)) cannot be produced: indeed, if $C_1(\cdot) \xrightarrow{\mu}$, then $C_1(b[\langle n \rangle^\star]) \xrightarrow{\mu}$ and $[\![\, P_1 \mid b[\langle n \rangle^\star] \,]\!] \longmapsto$; if $C^{\{b\}}_{n[\ ]}(\cdot) \xrightarrow{\mu}$, then $C_1(n[\langle b \rangle^\uparrow]) \xrightarrow{\mu}$ and $[\![\, P_1 \mid n[\langle b \rangle^\uparrow] \,]\!] \longmapsto$; finally, $[\![\, \langle b \rangle^\star \,]\!] \xrightarrow{\mu}$ is not possible because of Proposition 3.6.

Hence, it must be that $C_2(\llbracket P_2 \rrbracket) \xrightarrow{\mu}$, for $\mu \in \{n'(-), n'\langle - \rangle\}$. Again, the only way to respect Proposition 3.1 is when $C_2(\cdot)$ is empty, $C^{\{b\}}_{n[\,]}(\cdot)$ is of the form $(\nu\widetilde{m})(n'[\,\cdot \mid Q_1] \mid Q_2)$ and $\llbracket \langle b \rangle^\star \rrbracket \xrightarrow{\mu_1}$, for $\mu_1 \in \{(-)^\uparrow, \langle - \rangle^\uparrow\}$.

Now, consider processes $P_3 \triangleq \langle b \rangle^n.\surd$ and $P_4 \triangleq n[(x)^\star]$. With a similar reasoning, we have that $\llbracket (x)^\star \rrbracket \xrightarrow{\mu_2}$, for $\mu_2 \in \{(-)^\uparrow, \langle - \rangle^\uparrow\}$. Moreover, μ_2 must be of a different kind from μ_1: indeed, if they were both inputs (outputs), then we would have that $\llbracket P_3 \mid P_2 \rrbracket \longmapsto$.

Now, consider processes $P_5 \triangleq (x)^\star.\surd$ and $P_6 \triangleq\!! n[\langle b \rangle^\uparrow]$; it must be that $\llbracket P_5 \mid P_6 \rrbracket$ reduces but it cannot diverge. The possible interactions between their encodings are $\llbracket P_5 \rrbracket \xrightarrow{\mu_3}$ and $\llbracket P_6 \rrbracket \xrightarrow{\bar{\mu}_3}$, for $\mu_3 \in \{amb\_m, \langle - \rangle^m, (-)^m\}$ and, correspondingly, $\bar{\mu}_3 \in \{enter\_m, m(-), m\langle - \rangle\}$. Indeed, both $C_1(\cdot)$ and $C_2(\cdot)$ must be empty; moreover, $\llbracket P_6 \rrbracket$ cannot perform μ_3, otherwise at least one between $C^{\{b\}}_n(\cdot)$ and $\llbracket \langle b \rangle^\uparrow \rrbracket$ would be excluded from the production of μ_3. Moreover, it must be that $C^{\{n,b\}}_!(\cdot)$ has the hole at top-level, otherwise either $\llbracket P_5 \mid !\, n[\langle b \rangle^\star] \rrbracket \longmapsto$ or $\llbracket P_5 \mid !\, (\langle b \rangle^\uparrow \mid \langle n \rangle^\star) \rrbracket \longmapsto$. If the hole was underneath a '!', then we would have that $\llbracket P_5 \mid P_6 \rrbracket \longmapsto^\omega$, against Property 4; so, it must be that $C^{\{n,b\}}_!(\cdot)$ is of the form $(\nu\widetilde{m})(\cdot \mid Q)$. We then consider $\llbracket !\, n[in\_n.\langle b \rangle^\star] \rrbracket$ and work like in point 6 of the proof of Theorem 4.13.

The non-encodability of BA_s in BA is similar, but simpler. First, consider the BA_s processes $P_1 \triangleq (x)^n.\surd$ and $P_2 \triangleq n[\langle b \rangle^\uparrow]$. Like in the non-encodability of BA in BA_s, we have that $\llbracket \langle b \rangle^\uparrow \rrbracket \xrightarrow{\mu_1}$, for $\mu_1 \in \{(-)^\star, \langle - \rangle^\star\}$. Second, consider $P_3 \triangleq \langle b \rangle^n.\surd$ and $P_4 \triangleq n[(x)^\uparrow.\langle b \rangle^\star]$; again, we have that $\llbracket (x)^\uparrow \rrbracket \xrightarrow{\mu_2}$, for $\mu_2 \in \{(-)^\star, \langle - \rangle^\star\}$. We are now ready to violate Proposition 3.1: if μ_1 and μ_2 are of the same kind, then $\llbracket P_1 \mid P_4 \rrbracket \longmapsto$; otherwise, $\llbracket (x)^\uparrow \mid \langle b \rangle^\uparrow \rrbracket \longmapsto$. □

We now prove that the separation between BA and BA_s still holds, even if the target language is enhanced with the more powerful features of NBA.

Theorem 4.16. *There is no valid encoding of* BA *in* NBA.

Proof. The proof is similar to the separation between BA and BA_s. Let us again consider the BA processes $P_1 \triangleq (x)^n.(b[\mathbf{0}] \mid \surd)$ and $P_2 \triangleq n[\langle b \rangle^\star]$. Now, there is a further possible interaction in NBA:

d) $C_i(\llbracket P_i \rrbracket) \xrightarrow{h:exit\_n'}$ and $C_j(\llbracket P_j \rrbracket) \xrightarrow{?exit\_(h,n')}$, for $\{i, j\} = \{1, 2\}$ and $n' \in \varphi_{\llbracket\,\rrbracket}(n)$.

We now prove that this form of interaction is not possible and then the proof proceeds like in Theorem 4.15, by just using the richer labels of NBA in place of those of BA_s. First of all, $C_j(\cdot)$ must be empty (and the same holds for $C_i(\cdot)$ if we consider $\llbracket P_2 \mid P_1 \rrbracket$). Moreover, it must be that $j = 1$; indeed, if it were $j = 2$, either $\llbracket P_1 \mid n[\langle b \rangle^\uparrow] \rrbracket \longmapsto$ or $\llbracket P_1 \mid \langle b \rangle^\star \rrbracket \longmapsto$, according to whether $?exit\_(h, n')$ is performed by $C^{\{b\}}_{n[\,]}(\cdot)$ or by $\llbracket \langle b \rangle^\star \rrbracket$. Hence, it must be that $\llbracket P_2 \rrbracket \xrightarrow{h:exit\_n'}$. It must be that both $C^{\{b\}}_{n[\,]}(\cdot)$ and $\llbracket \langle b \rangle^\star \rrbracket$ contribute to the production of $h : exit\_n'$, otherwise we can easily violate Proposition 3.1 like before. Thus, $C^{\{b\}}_{n[\,]}(\cdot)$ must be of the form $(\nu\widetilde{m})(k[\cdot \mid R_1] \mid R_2)$ or $(\nu\widetilde{m})(k[h[\cdot \mid R_1] \mid R_2] \mid R_3)$. In any case, action $out\_(k, n')$ must appear within $\llbracket \langle b \rangle^\star \rrbracket$, in contradiction with Proposition 3.6 because $n' \in \varphi_{\llbracket\,\rrbracket}(n)$ and $n \neq b$. □

Similarly, we now prove that the separation between BA_s and BA still holds, even if the target language is enhanced with the more powerful features of SBA.

Theorem 4.17. *There is no valid encoding of* BA_s *in* SBA.

Proof. The proof is similar to the separation between BA_s and BA. Let us consider the BA_s processes $P_1 \triangleq (x)^n.\surd$ and $P_2 \triangleq n[\langle b\rangle^\uparrow]$. Now there is three further possible interactions in SBA; we just consider one of them, since the other ones are similar and can be obtained by adapting this case and the corresponding ones in the proof of Theorem 4.15.

d) $C_i([\![P_i]\!]) \xrightarrow{n':exit\_}$ and $C_j([\![P_j]\!]) \xrightarrow{?exit\_n'}$, for $\{i, j\} = \{1, 2\}$ and $n' \in \varphi_{[\![\ ]\!]}(n)$.

Like in the proof of Theorem 4.16, we can prove that $[\![P_2]\!] \xrightarrow{n':exit\_}$, with both $C^{\{b\}}_{n[\ ]}(\cdot)$ and $[\![\langle b\rangle^\uparrow]\!]$ that contribute to the production of $n' : exit\_$. This is possible only if $C^{\{b\}}_{n[\ ]}(\cdot)$ is of the form $(\widetilde{\nu k})(m[n'[\cdot \mid R_1] \mid R_2] \mid R_3)$ and $[\![\langle b\rangle^\uparrow]\!]$ performs an out_m.

Let us then consider $P_3 \triangleq \langle b\rangle^n.\surd$ and $P_4 \triangleq n[(x)^\uparrow.\langle b\rangle^\star]$. By what we have just said, it must be that $[\![P_4]\!]$ is of the form $(\widetilde{\nu k})(m[n'[[\![(x)^\uparrow.\langle b\rangle^\star]\!] \mid R_1] \mid R_2] \mid R_3)$. Hence, the only way for $[\![P_4]\!]$ to interact with $[\![P_3]\!]$ by checking a name of $\varphi_{[\![\ ]\!]}(n)$ and by involving in this both $C^{\{b\}}_{n[\ ]}(\cdot)$ and $[\![(x)^\uparrow.\langle b\rangle^\star]\!]$ is by having $[\![P_3]\!] \xrightarrow{?exit\_n'}$ and $[\![P_4]\!] \xrightarrow{n':exit\_}$. But then we would have that $[\![P_1 \mid P_4]\!] \longmapsto$, against Proposition 3.1. □

4.5 Completing the Taxonomy and Composing Valid Encodings

We have just given several encodability and separation results for the considered process calculi. However, to fully define the taxonomy of Figure 1, we have to consider *every* pair of calculi and establish an encodability or a separation result. We can now proceed in two ways: either we prove every encodability/separation result, or we can work by composing encodings. The second way may seem much better than the first one. However, as we shall now show, it has strong drawbacks, and indeed we shall adopt the first way. Actually, several results can be proved similarly to theorems we have just proved. In Table 1 we exhaustively compare all our calculi and say how a result relating them can be proved.

Let us discuss the other approach, i.e. the one based on composing encodings. Let us consider an example. The encodability of π_a in SAP can be proved by composing the encoding of π_a in Dπ and the encoding of Dπ in SAP. Similarly, the non-encodability of SAP into π_a can be proved by contradiction: if SAP were encodable in π_a it would also been encodable in Dπ, by composing the two encodings, against what we have proved in Theorem 4.10. However, both these arguments rely on the fact that the composition of valid encodings is still a valid encoding. Actually, this is *not* always the case. We now prove that all properties hold in general, except for Property 3 that only holds under some assumptions on the encodings and on the behavioural equivalences considered.

Definition 4.4. *An encoding* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ preserves equivalences *if, for every* $S \simeq_1 S'$*, it holds that* $[\![S]\!] \simeq_2 [\![S']\!]$.

Definition 4.5. *An equivalence* $\simeq$ *is* reduction closed *if, for every* $P \simeq Q$ *and* $P \Longmapsto P'$*, there exists* Q' *such that* $Q \Longmapsto Q'$ *and* $P' \simeq Q'$.

Proposition 4.18. *Let* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *and* $(\!| \cdot |\!) : \mathcal{L}_2 \to \mathcal{L}_3$ *be valid encodings; let* $\{\!\{ \cdot \}\!\} : \mathcal{L}_1 \to \mathcal{L}_3$ *be their composition (i.e.,* $\{\!\{ \cdot \}\!\} \triangleq (\!| [\![\cdot]\!] |\!)$*). Then,* $\{\!\{ \cdot \}\!\}$ *enjoys Properties 1, 2, 4 and 5; moreover, if* $(\!| \cdot |\!)$ *preserves equivalences and* $\simeq_3$ *is reduction closed, then* $\{\!\{ \cdot \}\!\}$ *also enjoys Property 3.*

Proof. Trivially, $\{\!\{ \cdot \}\!\}$ enjoys Properties 4 and 5. Let us now consider the remaining properties. To this aim, we shall denote with $P, P', \ldots$ processes of $\mathcal{L}_1$, with $Q, Q', \ldots$ processes of $\mathcal{L}_2$ and with $R, R', \ldots$ processes of $\mathcal{L}_3$.

From \ To	π_a	Dπ	MA	SA	SAP	BA	BA$_s$	SBA	NBA
π_a	Id	Sec. 4.2.1	Sec. 4.2.3	[33]	like [33]	[7]	like [7]	like [7]	like [7]
Dπ	Corollary of Th. 3.5	Id	Corollary of Th. 3.5	Corollary of Th. 3.5	Sec. 4.3.4	Corollary of Th. 3.5	Corollary of Th. 3.5	Sec. 4.3.4	Sec. 4.3.4
MA	Corollary of Th. 3.4	Corollary of Th. 3.4	Id	Sec. 4.3.1	Similar to Sec. 4.3.1	Th. 4.13	Similar to Th. 4.13	Similar to Th. 4.13	Similar to Th. 4.13
SA	Corollary of Th. 3.4	Corollary of Th. 3.4	Th. 4.9	Id	Th. 4.14	Similar to Th. 4.13	Similar to Th. 4.13	Similar to Th. 4.13	Similar to Th. 4.13
SAP	Corollary of Th. 3.4	Th. 4.10	Corollary of Th. 3.5	Corollary of Th. 3.5	Id	Corollary of Th. 3.5	Corollary of Th. 3.5	Similar to Th. 4.13	Similar to Th. 4.13
BA	Th. 4.5	Corollary of Th. 3.4	Similar to Th. 4.12	Th. 4.12	Th. 4.12	Id	Th. 4.15	Sec. 4.3.2	Th. 4.16
BA$_s$	Th. 4.5	Corollary of Th. 3.4	Similar to Th. 4.12	Th. 4.12	Th. 4.12	Th. 4.15	Id	Th. 4.17	Sec. 4.3.3
SBA	Corollary of Th. 3.4	Th. 4.10	Corollary of Th. 3.5	Corollary of Th. 3.5	Similar to Th. 4.12	Corollary of Th. 3.5	Corollary of Th. 3.5	Id	Similar to Th. 4.16
NBA	Corollary of Th. 3.4	Th. 4.10	Corollary of Th. 3.5	Corollary of Th. 3.5	Similar to Th. 4.12	Corollary of Th. 3.5	Corollary of Th. 3.5	Similar to Th. 4.17	Id

Table 1: Comparing every pair of languages. (The cell at row $\mathcal{L}_1$ and column $\mathcal{L}_2$ describes the possibility for a valid encoding of $\mathcal{L}_1$ into $\mathcal{L}_2$. A highlighted cell means that a valid encoding exists; a plain cell means that it does not. 'Id' stands for the identity, i.e. the encoding mapping every term to itself.)

Property 1: By hypothesis, $[\![\, \mathsf{op}(P_1,\ldots,P_k) \,]\!] = C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}([\![\, P_1 \,]\!];\ldots;[\![\, P_k \,]\!])$. By induction on $C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}(\_1;\ldots;\_k)$, we prove that there exists a k-ary $\mathcal{L}_3$-context $\mathcal{D}(\_1;\ldots;\_k)$ that only depends on op and $fn(P_1,\ldots,P_k)$ such that $(\!|\, C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}([\![\, P_1 \,]\!];\ldots;[\![\, P_k \,]\!]) \,|\!) = \mathcal{D}(\{\!|P_1|\!\};\ldots;\{\!|P_k|\!\})$. The simplest possible context is when $k = 0$ and so $C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}(\_1;\ldots;\_k)$ is a normal process (i.e., a context without holes); thus, op is a 0-ary operator (i.e., a constant) and also for $\{\!|\cdot|\!\}$ we need a 0-ary context, that can be defined to be $(\!|\, C_{\mathsf{op}}^{\emptyset}() \,|\!)$. Another base case is when $C_{\mathsf{op}}^{N}(\cdot) = \cdot$. In this case, $[\![\, \mathsf{op}(P) \,]\!] = [\![\, P \,]\!]$ and, hence, $\{\!|\mathsf{op}(P)|\!\} = \{\!|P|\!\}$; also in this case we conclude, by letting $\mathcal{D}(\cdot)$ be ' $\cdot$ '.

For the inductive step, let op' be the outermost operator of $C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}(\_1;\ldots;\_k)$, i.e. $[\![\, \mathsf{op}(P_1,\ldots,P_k) \,]\!] = \mathsf{op}'(C_1([\![\, P_{1_1} \,]\!];\ldots;[\![\, P_{1_{h_1}} \,]\!]);\ldots;C_m([\![\, P_{m_1} \,]\!];\ldots;[\![\, P_{m_{h_m}} \,]\!]))$, with $\{P_1,\ldots,P_k\} = \{P_{1_1},\ldots,P_{1_{h_1}},\ldots,P_{m_1},\ldots,P_{m_{h_m}}\}$. Now, every $C_i([\![\, P_{i_1} \,]\!];\ldots;[\![\, P_{i_{h_i}} \,]\!])$ is a $\mathcal{L}_2$-process Q_i; thus, $(\!|\, \mathsf{op}'(Q_1,\ldots,Q_m) \,|\!) = \mathcal{D}'^{fn(Q_1,\ldots,Q_m)}_{\mathsf{op}'}((\!|\, Q_1 \,|\!);\ldots;(\!|\, Q_m \,|\!))$. Since every $C_i(\_{i_1};\ldots;\_{i_{h_i}})$ is smaller than $C_{\mathsf{op}}^{fn(P_1,\ldots,P_k)}(\_1;\ldots;\_k)$, by inductive hypothesis there exists $\mathcal{D}_i(\_{i_1};\ldots;\_{i_{h_i}})$ depending only on the outermost operator of C_i and on $fn(C_i) \cup (fn(P_{i_1},\ldots,P_{i_{h_i}}) \setminus bn(C_i))$ such that $(\!|\, C_i([\![\, P_{i_1} \,]\!];\ldots;[\![\, P_{i_{h_i}} \,]\!]) \,|\!) = \mathcal{D}_i(\{\!|P_{i_1}|\!\};\ldots;\{\!|P_{i_{h_i}}|\!\})$. Then, notice that $fn(Q_i) = fn(C_i) \cup (fn(P_{i_1},\ldots,P_{i_{h_i}}) \setminus bn(C_i))$. Then, the desired $\mathcal{D}(\_1;\ldots;\_k)$ is $\mathcal{D}'^{fn(Q_1,\ldots,Q_m)}_{\mathsf{op}'}(\mathcal{D}_1(\_1{}_1;\ldots;\_1{}_{h_1});\ldots;\mathcal{D}_m(\_m{}_1;\ldots;\_m{}_{h_m}))$: it only depends on op and $fn(P_1,\ldots,P_k)$ since $\mathsf{op}', fn(Q_1,\ldots,Q_m)$ and all the $\mathcal{D}_i$'s depend only on them.

Property 2: Let $\varphi_{[\![\,]\!]} : \mathcal{N} \to \mathcal{N}^k$ and $\varphi_{(\!|\,|\!)} : \mathcal{N} \to \mathcal{N}^h$ be the renaming policies of $[\![\cdot]\!]$ and of $(\!|\cdot|\!)$. We let $\varphi_{\{\!|\,|\!\}} : \mathcal{N} \to \mathcal{N}^{kh}$ be defined by composing $\varphi_{[\![\,]\!]}$ and $\varphi_{(\!|\,|\!)}$, i.e. $\varphi_{\{\!|\,|\!\}}(a) = \varphi_{(\!|\,|\!)}(a_1)\ldots\varphi_{(\!|\,|\!)}(a_k)$, whenever $\varphi_{[\![\,]\!]}(a) = a_1\ldots a_k$. Let σ be any injective name substitution (the case for non-injective substitutions is similar: just replace '=' with '$\simeq$'); by hypothesis, $[\![\, P\sigma \,]\!] = [\![\, P \,]\!]\sigma'$, for $\sigma' = \{{}^{\varphi_{[\![\,]\!]}(a)}/_{\varphi_{[\![\,]\!]}(b)} \;:\; \sigma(b) = a\}$. Now consider $\{\!|P\sigma|\!\} = (\!|\, [\![\, P \,]\!]\sigma' \,|\!)$; by hypothesis, $(\!|\, [\![\, P \,]\!]\sigma' \,|\!) = \{\!|P|\!\}\sigma''$, with $\sigma'' = \{{}^{\varphi_{(\!|\,|\!)}(a')}/_{\varphi_{(\!|\,|\!)}(b')} \;:\; \sigma'(b') = a'\} = \{{}^{\varphi_{[\![\,]\!]}(a)}/_{\varphi_{[\![\,]\!]}(b)} \;:\; \sigma(b) = a\}$. This suffices to conclude.

Property 3: Let $P \Longmapsto_1 P'$; by hypothesis, there exists Q such that $[\![\, P \,]\!] \Longmapsto_2 Q \simeq_2 [\![\, P' \,]\!]$ and there exists R such that $\{\!|P|\!\} \Longmapsto_3 R \simeq_3 (\!|\, Q \,|\!)$. Since $(\!|\cdot|\!)$ preserves equivalences, this entails that $R \simeq_3 \{\!|P'|\!\}$, and this suffices to conclude the completeness part. For soundness, let $\{\!|P|\!\} \triangleq (\!|\, [\![\, P \,]\!] \,|\!) \Longmapsto_3 R$. By hypothesis, $[\![\, P \,]\!] \Longmapsto_2 Q$ and $R \Longmapsto_3 R' \simeq_3 (\!|\, Q \,|\!)$; moreover, $P \Longmapsto_1 P'$ and $Q \Longmapsto_2 Q' \simeq_2 [\![\, P' \,]\!]$. This entails that $(\!|\, Q \,|\!) \Longmapsto_3 \simeq_3 (\!|\, Q' \,|\!) \simeq_3 (\!|\, [\![\, P' \,]\!] \,|\!) \triangleq \{\!|P'|\!\}$, since $(\!|\cdot|\!)$ preserves equivalences. By reduction closure of $\simeq_3$, we also have that $R' \Longmapsto_3 \simeq_3 \{\!|P'|\!\}$, as desired. □

It has to be said that the two hypotheses needed for proving transitivity of operational correspondence are too demanding. In particular, preservation of equivalences is one direction of full abstraction, a criterion that we prefer not to use; actually, it is the most demanding implication of full abstraction and several well-known encodings ([6, 30, 38, 39], just to cite the most notable examples) do not enjoy it. Moreover, also reduction closure can be criticized: there are several behavioural equivalences that do not enjoy it. For this reason, let us now consider another set of properties that are sufficient for ensuring transitivity of operational correspondence. Let us consider the following formulation of the latter property.

Definition 4.6 (Operational Correspondence, revised)**.**

- Completeness: *for every S and S' such that $S \Longmapsto_1 S'$, it holds that $[\![S]\!] \Longmapsto_2 [\![S']\!] \mid T$, for some $T \simeq_2 \mathbf{0}$;*
- Soundness: *for every S and T such that $[\![S]\!] \Longmapsto_2 T$, there exists S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 [\![S']\!] \mid T$, for some $T \simeq_2 \mathbf{0}$.*

This formulation of our property is a specialization of the third setting presented in Section 3.2. We believe that it is not too demanding, since all the encodings we are aware of satisfy it. Another property, much less demanding than equivalence preservation, is the following one.

Definition 4.7. *An encoding $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$* preserves the equivalence class of $\mathbf{0}$ *if, for every $S \simeq_1 \mathbf{0}$, it holds that $[\![S]\!] \simeq_2 \mathbf{0}$.*

Proposition 4.19. *Let $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ and $(\!| \cdot |\!) : \mathcal{L}_2 \to \mathcal{L}_3$ be valid encodings (with Property 3 formulated as in Definition 4.6); let $\{\!| \cdot |\!\} : \mathcal{L}_1 \to \mathcal{L}_3$ be their composition (i.e., $\{\!| \cdot |\!\} \triangleq (\!| [\![\cdot]\!] |\!)$). Then, $\{\!| \cdot |\!\}$ enjoys Properties 1, 2, 4 and 5; moreover, if $(\!| \cdot |\!)$ preserves the equivalence class of $\mathbf{0}$ and is homomorphic w.r.t. '$\mid$', then also $\{\!| \cdot |\!\}$ enjoys Property 3 (formulated as in Definition 4.6).*

Proof. Properties 1, 2, 4 and 5 for $\{\!| \cdot |\!\}$ can be proved like in Proposition 4.18. Let us consider Property 3 formulated as in Definition 4.6.

Completeness: Let $P \Longmapsto_1 P'$; by hypothesis, there exists $Q \simeq_2 \mathbf{0}$ such that $[\![P]\!] \Longmapsto_2 [\![P']\!] \mid Q$ and there exists $R \simeq_3 \mathbf{0}$ such that $\{\!|P|\!\} \Longmapsto_3 (\!| [\![P']\!] \mid Q |\!) \mid R \triangleq \{\!|P'|\!\} \mid (\!| Q |\!) \mid R$, where the latter equality holds by homomorphism of $(\!| \cdot |\!)$. Since $(\!| \cdot |\!)$ preserves the equivalence class of $\mathbf{0}$ and $Q \simeq_2 \mathbf{0}$, this entails that $(\!| Q |\!) \mid R \simeq_3 \mathbf{0}$, and this suffices to conclude the completeness part.

Soundness: Let $\{\!|P|\!\} = (\!| [\![P]\!] |\!) \Longmapsto_3 R$. By hypothesis, $[\![P]\!] \Longmapsto_2 Q$ and $R \Longmapsto_3 (\!| Q |\!) \mid R'$, for some $R' \simeq_3 \mathbf{0}$; moreover, $P \Longmapsto_1 P'$ and $Q \Longmapsto_2 [\![P']\!] \mid Q'$, for some $Q' \simeq_2 \mathbf{0}$. By completeness, this entails that $(\!| Q |\!) \Longmapsto_3 (\!| [\![P']\!] \mid Q' |\!) \mid R''$, for some $R'' \simeq_2 \mathbf{0}$, and, hence, $R \Longmapsto_3 (\!| [\![P']\!] \mid Q' |\!) \mid R'' \mid R'$. Since $(\!| \cdot |\!)$ translates '$\mid$' homomorphically and preserves of the equivalence class of $\mathbf{0}$, we have that $(\!| [\![P']\!] \mid Q' |\!) \mid R'' \mid R' \triangleq \{\!|P'|\!\} \mid (\!| Q' |\!) \mid R'' \mid R'$ and $(\!| Q' |\!) \mid R'' \mid R' \simeq_3 \mathbf{0}$, as desired. □

To conclude, the fact that the encodability relation put forward by valid encodings is not transitive is annoying. The idea of having composable encodings could be seen as a basic issue in a "modular" metatheory for expressiveness, where independent results can be combined to yield new results. A simple way to solve this problem is to assume an "exact" formulation of operational correspondence, i.e. by removing the occurrences of $\simeq_2$ from Property 3 (or, equivalently, to assume $\simeq_2$ to be the identity) or to let them be structural equivalences. Indeed, both the identity and the structural equivalence are reduction closed and every encoding usually preserves them. However, as we have discussed in the Introduction, such formulations of operational correspondence are too demanding: with them, the encodings of MA in π_a and of MA in SA would not be acceptable anymore, to focus only on some encodings of this paper. We prefer to adopt a more generous notion of validity, even if the price to be paid is the loss of a more elegant metatheory. The development of better notions of validity, that also require less demanding conditions for transitivity, is a challenging direction for future research.

5 Conclusions and Related Work

We have comparatively studied several mainstream calculi for mobility and distribution, together with some of their variants, namely the asynchronous π-calculus, a distributed π-calculus, Mobile Ambients, Safe Ambients (and its dialect with passwords) and Boxed Ambients (and some variations of its primitives). We have organized all these languages in a clear hierarchy based on their relative expressiveness. To this aim, we have exploited the criteria presented and discussed in [25].

It is now worth discussing the notion of expressiveness we have considered when comparing these languages. One might intuitively consider a language more expressive than another one if the former allows more sophisticated inter-process interactions than the latter; moreover, it could also be expected that systems in the former language should be expressible with a more compact syntax and simpler operational semantics than in the latter one. Quite surprisingly, the notion of expressiveness put forward by our results in some cases clashes with this intuition. For example, SA and SAP, defined to limit the possible computations of MA, turned out to be more expressive than MA (a similar situation holds for SBA and NBA w.r.t. BA and BA_s). This apparent contradiction is related to operational soundness, viz. the second item of Property 3. Not incidentally, by ignoring it, more and simpler encodability results do hold.

Finally, the throughout comparison between the different dialects of ambient-based calculi has also clarified some important issues. In some cases, we have discovered that the dialect proposed is comparable, in terms of expressive power, with the language it comes from: for example, SA and SBA/NBA enhance the expressiveness of MA and BA/BA_s, respectively. In other cases, we have discovered that the dialect and its original language are incomparable, i.e. no relative encoding exists: the most notable cases are BA_s vs BA and SAP vs SA. In these cases, we must be aware that the dialect is not an enhancement of the original language nor a minor variation on it, as it is sometimes believed.

Related work. To conclude, we want to mention some strictly related results. First, [54] provides an encoding of the synchronous π-calculus in 'pure' SA (i.e. SA without communications) and claims that the same cannot be done in 'pure' MA (i.e. MA without communications). Second, [32] provides an encoding of BA_s in a variant of SA that exploits mobility primitives similar to those in SBA. The encoding respects all our criteria but the target language is still another variant of the languages we have presented. Third, the results in [11] entail that Dπ cannot be encoded in π_a, under properties similar to ours; notably, they need homomorphism w.r.t. parallel composition whereas we just rely on compositionality. Fourth, [47, 48] are inspired by Palamidessi's work on electoral systems [44] and separate several calculi for mobility according to the possibility of solving the problem of leader election. Though their approach is different from ours, our results confirm theirs. However, our approach is more informative than theirs, since we are also able to compare pairs of languages in which leader election is possible/impossible (e.g., SA and MA, or π_a and Dπ).

Finally, calculi for mobility have been a workbench for investigations on the expressiveness of operators like restriction, communication primitives, non-deterministic choice and replication ([10, 17, 22, 34, 44], just to cite some examples). These works are quite orthogonal to ours, since they compare different sub-calculi of the same language, whereas we aimed at comparing different programming paradigms.

Acknowledgments. Thanks to Iain Phillips that introduced me to [47, 48]; thanks also to Rosario Pugliese, Ivano Salvo and Maria Grazia Vigliotti that read a preliminary version of this work. 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: Validity of the Encoding of π_a into MA

Properties 1 and 2 hold by construction. Let us prove the remaining properties. In the rest of this section, $\simeq$ will denote strong barbed equivalence for MA.

Operational Completeness We shall only prove the following claim:

$$\text{If } P \longmapsto P' \text{ then } [\![P]\!] \Longmapsto\simeq [\![P']\!]$$

since the more general case with '$\Longmapsto$' in place of '$\longmapsto$' in the premise can be obtained by straightforward inductive arguments. The proof is by induction on the inference for '$\longmapsto$' and only the base case is interesting. Thus, we have that $P \triangleq \bar{a}\langle b\rangle \mid a(x).Q$ and $P' \triangleq Q\{^{b}/_{x}\}$. To make the following proofs easier, let us spell out the reductions of $[\![P]\!]$ and give them a number, to better refer them later on. For the sake of simplicity, we shall use biadic communications, with the caution that this is just a notational shortcut that can be rendered via monadic communications and opening of the reserved ambient `poly`. Moreover, to ease reading, we highlight the parts of the process that are involved in the generation of

the next transition.

$$
\begin{aligned}
&[\![\, \bar{a}\langle b\rangle \mid a(x).Q \,]\!] \\
&\quad \triangleq\ a_1\,[\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle]] \\
&\qquad \mid\ open\_a_1\,.(\nu r, s, t, u)(open\_u \mid t[u[open\_a_2.out\_t]] \\
&\qquad\qquad \mid\ a_2[\mathtt{q}[in\_\mathtt{p}.(x_1, x_2).in\_r.s[out\_\mathtt{p}.out\_r.in\_t.in\_u.[\![\, Q \,]\!]]] \mid r[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{1}} \mathrm{PR}^{\boxed{1}}_{a,b,x,Q} \triangleq\ \mathtt{p}[!\,in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \\
&\qquad\qquad \mid\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \mid a_2\,[\mathtt{q}[\cdots] \mid r[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{2}} \mathrm{PR}^{\boxed{2}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[\,\mathtt{p}\,[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \mid \mathtt{q}[\,in\_\mathtt{p}\,.(x_1, x_2).\cdots] \mid r[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{3}} \mathrm{PR}^{\boxed{3}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[\mathtt{p}[!in\_a_2 \mid\, open\_\mathtt{q}\,.\langle b_1, b_2\rangle \mid \mathtt{q}\,[(x_1, x_2).\cdots]] \mid r[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{4}} \mathrm{PR}^{\boxed{4}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[\mathtt{p}[!in\_a_2 \mid\, \langle b_1, b_2\rangle \mid (x_1, x_2)\,.in\_r.\cdots] \mid r[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{5}} \mathrm{PR}^{\boxed{5}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[\mathtt{p}[!in\_a_2 \mid\, in\_r\,.s[out\_\mathtt{p}.out\_r.in\_t.in\_u.[\![\, Q\{^b\!/_x\} \,]\!]]] \mid\ r\,[\,] \mid open\_s] \,) \\
&\longmapsto_{\boxed{6}} \mathrm{PR}^{\boxed{6}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[r[\,\mathtt{p}\,[!in\_a_2 \mid s[\,out\_\mathtt{p}\,.out\_r.in\_t.in\_u.[\![\, Q\{^b\!/_x\} \,]\!]]]] \mid open\_s] \,) \\
&\longmapsto_{\boxed{7}} \mathrm{PR}^{\boxed{7}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[\,r\,[\mathtt{p}[!in\_a_2]] \mid s[\,out\_r\,.in\_t.in\_u.[\![\, Q\{^b\!/_x\} \,]\!]] \mid open\_s] \,) \\
&\longmapsto_{\boxed{8}} \mathrm{PR}^{\boxed{8}}_{a,b,x,Q} \triangleq\ (\nu r, s, t, u)(open\_u \mid t[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[r[\mathtt{p}[!in\_a_2]] \mid\, s\,[in\_t.in\_u.[\![\, Q\{^b\!/_x\} \,]\!]] \mid\ open\_s\,] \,) \\
&\longmapsto_{\boxed{9}} \mathrm{PR}^{\boxed{9}}_{a,b,x,Q} \triangleq\ (\nu r, t, u)(open\_u \mid\, t\,[u[\cdots]] \\
&\qquad\qquad \mid\ a_2[r[\mathtt{p}[!in\_a_2]] \mid\, in\_t\,.in\_u.[\![\, Q\{^b\!/_x\} \,]\!]] \,) \\
&\longmapsto_{\boxed{10}} \mathrm{PR}^{\boxed{10}}_{a,b,x,Q} \triangleq\ (\nu r, t, u)(open\_u \mid t[\,u\,[\cdots] \mid a_2[r[\mathtt{p}[!in\_a_2]] \mid\, in\_u\,.[\![\, Q\{^b\!/_x\} \,]\!]]] \,) \\
&\longmapsto_{\boxed{11}} \mathrm{PR}^{\boxed{11}}_{a,b,x,Q} \triangleq\ (\nu r, t, u)(open\_u \mid t[u[\,open\_a_2\,.out\_t \mid\, a_2\,[r[\mathtt{p}[!in\_a_2]] \mid [\![\, Q\{^b\!/_x\} \,]\!]]]] \,) \\
&\longmapsto_{\boxed{12}} \mathrm{PR}^{\boxed{12}}_{a,b,x,Q} \triangleq\ (\nu r, t, u)(open\_u \mid\, t\,[u[\,out\_t\ \mid r[\mathtt{p}[!in\_a_2]] \mid [\![\, Q\{^b\!/_x\} \,]\!]]] \,) \\
&\longmapsto_{\boxed{13}} \mathrm{PR}^{\boxed{13}}_{a,b,x,Q} \triangleq\ (\nu r, t, u)(\,open\_u\ \mid t[\,] \mid\, u\,[r[\mathtt{p}[!in\_a_2]] \mid [\![\, Q\{^b\!/_x\} \,]\!] \,) \\
&\longmapsto_{\boxed{14}} \mathrm{PR}^{\boxed{14}}_{a,b,x,Q} \qquad (\nu r, t)(t[\,] \mid r[\mathtt{p}[!in\_a_2]]) \mid [\![\, Q\{^b\!/_x\} \,]\!]
\end{aligned}
$$

To conclude, ti suffices to notice that

$$\mathtt{Garb}_a \triangleq (\nu r, t)(t[\,] \mid r[\mathtt{p}[!in\_a_2]]) \simeq \mathbf{0}$$

Operational Soundness We now have to explicitly take care of the possible interferences between the encodings of different communications along the same channel. In such a case, some new reductions arise, i.e. those leading a copy of $\mathtt{p}$ within an a_2 already entered by at least another ambient $\mathtt{p}$. Of course, such an ambient can still be within a_2 (i.e., the communication is not yet finished), or it is

within r. However, such reductions are *spurious*, in the sense that they do not correspond to original reductions in π_a. Formally:

$$\begin{aligned}&\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \mid a_2\,[\mathtt{p}[\cdots] \mid \cdots]\\ &\quad \longmapsto_{\boxed{2s}} a_2\,[\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \mid \mathtt{p}[\cdots] \mid \cdots]\end{aligned}$$

$$\begin{aligned}&\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \mid a_2\,[r[\mathtt{p}[\cdots]] \mid \cdots]\\ &\quad \longmapsto_{\boxed{2s}} a_2\,[\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_1, b_2\rangle] \mid r[\mathtt{p}[\cdots]] \mid \cdots]\end{aligned}$$

In these cases, we denote the step with $\boxed{2s}$ to emphasize that it is of kind $\boxed{2}$ while stressing its spurious nature. A process of kind $\mathtt{PR}^{\boxed{k}}_{a,b,x,Q}$ can undergo a reduction of kind $\boxed{2s}$ whenever $k \in \{1,\ldots,9\}$. It then becomes a new process, very similar to $\mathtt{PR}^{\boxed{k}}_{a,b,x,Q}$ but with the new parallel component $\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b'_1, b'_2\rangle]$ within a_2. Let us denote with $\mathtt{PR}^{\boxed{k},s}_{a,b,x,Q}$ process $\mathtt{PR}^{\boxed{k}}_{a,b,x,Q}$ with s parallel components of kind $\mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle\cdots\rangle]$ within a_2, if $k \in \{1,\ldots,11\}$, or within u, otherwise.

Let us use metavariable ℓ to range over $\{1,\ldots,14,2s\}$. Then, the numbered reductions defined so far are closed under addition of spurious parallel components:

$$\mathtt{PR}^{\boxed{k},s}_{a,b,x,Q} \longmapsto_{\boxed{k+1}} \mathtt{PR}^{\boxed{k+1},s}_{a,b,x,Q} \quad \text{for } k \in \{1,\ldots,12\}$$

$$\mathtt{PR}^{\boxed{13},s}_{a,b,x,Q} \longmapsto_{\boxed{14},s} \mathtt{PR}^{\boxed{14}}_{a,b,x,Q} \mid \prod_{i=1}^{s} \mathtt{p}[!in\_a_2 \mid open\_\mathtt{q}.\langle b_{i_1}, b_{i_2}\rangle]$$

Indeed, after a reduction of kind $\boxed{14}$, all the s parallel components of kind $\mathtt{p}[\cdots]$ reappear at top-level, and it is necessary to know how many they are for carrying out proofs. Labeled reductions are then closed under evaluation contexts and structural congruence:

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

$$\frac{P \longmapsto_{\boxed{14},s} P'}{\mathcal{E}(P) \longmapsto_{\boxed{14},s} \mathcal{E}(P')} \qquad \frac{P \equiv Q \longmapsto_{\boxed{14},s} Q' \equiv P'}{P \longmapsto_{\boxed{14},s} P'}$$

Given a sequence of reductions of an encoded term, let us denote with $s(a)$ the sum of all the s labeling a reduction of kind $\boxed{14}$ originating from a communication along a. Moreover, we denote with $n^a_{\boxed{\ell}}$ the number of reductions of kind ℓ derived from a communication along a; $n_{\boxed{\ell}}$ stands for $\sum_{a\in\mathcal{N}} n^a_{\boxed{\ell}}$. We also let

$$\widehat{n^a_{\boxed{k}}} \triangleq \begin{cases} n^a_{\boxed{1}} - n^a_{\boxed{2}} - n^a_{\boxed{2s}} + s(a) & \text{if } k = 1\\ n^a_{\boxed{k}} - n^a_{\boxed{k+1}} & \text{if } k \in \{2,\ldots,13\}\end{cases}$$

Finally, we let $\mathtt{PR}_{a,x,Q}$ denote $[\![\, a(x).Q \,]\!]$ without the starting $open\_a_1$ prefix. We are now ready to prove the following lemma, that will easily entail operational soundness.

Lemma 5.1. *Let P be a π_a-process and Q be an* MA *process such that $[\![\, P \,]\!] \longmapsto^n Q$, for $n = \sum_{\ell\in\{1,\ldots,14,2s\}} n_{\boxed{\ell}}$. Then,*

$$Q \equiv (\nu\widetilde{m})\left([\![\, R \,]\!] \mid \prod_{a\in\mathcal{N}}\left(\prod_{i=1}^{n^a_{\boxed{2s}}-s(a)} \mathtt{PR}_{a,x_i,P_i} \mid \prod_{k=1}^{13}\prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathtt{PR}^{\boxed{k},\,s_i}_{a,b_i,x_i,P_i} \mid \prod_{i=1}^{n^a_{\boxed{14}}} \mathtt{Garb}_a\right)\right)$$

where $[\![R]\!]$ *has no top-level restrictions,* $s_i = 0$ *whenever* $k = 1$ *and* $n^a_{\boxed{2s}} = \sum_{k=1}^{13} \sum_{i=1}^{\widehat{n^a_{\boxed{k}}}} s_i$.

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

$$Q \equiv (\nu \widetilde{m})\left([\![R]\!] \mid \prod_{a \in \mathcal{N}} \left(\prod_{i=1}^{n^a_{\boxed{2s}} - s(a)} \mathtt{PR}_{a,x_i,P_i} \mid \prod_{k=1}^{13} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathtt{PR}^{\boxed{k},\, s_i}_{a,b_i,x_i,P_i} \mid \prod_{i=1}^{n^a_{\boxed{14}}} \mathtt{Garb}_a \right) \right)$$

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 $\boxed{1}$;
- $\mathtt{PR}^{\boxed{k},\, s_i}_{a,x_i,b_i,P_i}$, for $k > 1$, can only evolve in isolation by performing a reduction of kind $\boxed{k+1}$;
- $\mathtt{PR}^{\boxed{1},\, s_i}_{a,x_i,b_i,P_i}$ can evolve in isolation by performing a reduction of kind $\boxed{2}$, or it can perform a reduction of kind $\boxed{2s}$ by interacting with some $\mathtt{PR}^{\boxed{h},\, s_j}_{a,x_j,b_j,P_j}$, for $h \in \{2, \ldots, 9\}$.

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, $R \equiv \bar{a}\langle b \rangle \mid a(x).R' \mid R''$. The thesis easily follows by noting that, after the reduction $Q \longmapsto Q'$, the new value of $\widehat{n^a_{\boxed{1}}}$ is the old value plus one: the new process is $\mathtt{PR}^{\boxed{1},\, 0}_{a,b,x,R'}$. Finally, $[\![R'']\!]$ has no top-level restrictions, since $[\![R]\!]$ has none.

2. The reduction has been originated by $\mathtt{PR}^{\boxed{k},\, s_i}_{a,b_i,x_i,P_i}$, for some $k \in \{2, \ldots, 12\}$ and $i \in \{1, \ldots, \widehat{n^a_{\boxed{k}}}\}$. Then, the reduction is $\mathtt{PR}^{\boxed{k},\, s_i}_{a,b_i,x_i,P_i} \longmapsto_{\boxed{k+1}} \mathtt{PR}^{\boxed{k+1},\, s_i}_{a,b_i,x_i,P_i}$ and we can easily conclude, since the new value of $\widehat{n^a_{\boxed{k}}}$ is the old value minus one (corresponding to the fact that $\mathtt{PR}^{\boxed{1},\, 0}_{a,b_i,x_i,P_i}$ has evolved) and the new value of $\widehat{n^a_{\boxed{k+1}}}$ is the old value plus one (corresponding to the fact that $\mathtt{PR}_{a,x_i,P_i}$ is added to the first product).

3. The reduction has been originated by $\mathtt{PR}^{\boxed{1},\, 0}_{a,b_i,x_i,P_i}$, for some $i \in \{1, \ldots, \widehat{n^a_{\boxed{1}}}\}$. If it is of kind $\boxed{2}$, we reason like in case 2; if it is of kind $\boxed{2s}$, there exist $h \in \{2, \ldots, 9\}$ and $j \in \{1, \ldots, \widehat{n^a_{\boxed{h}}}\}$ such that $\mathtt{PR}^{\boxed{1},\, 0}_{a,b_i,x_i,P_i} \mid \mathtt{PR}^{\boxed{h},\, s_j}_{a,b_j,x_j,P_j} \longmapsto_{\boxed{2s}} \mathtt{PR}_{a,x_i,P_i} \mid \mathtt{PR}^{\boxed{h},\, s_j+1}_{a,b_j,x_j,P_j}$. Also in this case we can conclude by letting the new value of s_j be the old value plus one, by noting that the new value of $\widehat{n^a_{\boxed{1}}}$ is the old value minus one and that the new value of $n^a_{\boxed{2s}}$ is the old value plus one.

4. The reduction has been originated by $\mathtt{PR}^{\boxed{13},\, s_i}_{a,b_i,x_i,P_i}$, for some $i \in \{1, \ldots, \widehat{n^a_{\boxed{13}}}\}$, and it is of kind $\boxed{14}$. Such a reduction produces a new copy of $\mathtt{Garb}_a$ (that is collected in the last product and, indeed, the new value of $n^a_{\boxed{14}}$ is the old value plus one) and s_i copies of processes $\mathtt{p}[\cdots]$. By taking s_i processes of kind $\mathtt{PR}_{a,x_i,P_i}$ from the first product, we create s_i new components of kind $\mathtt{PR}^{\boxed{1},\, 0}_{a,b_i,x_i,P_i}$. Indeed, the new value of $s(a)$ is the old value plus s_i; this leads the new value of $n^a_{\boxed{2s}} - s(a)$ to the old one minus s_i and the new value of $\widehat{n^a_{\boxed{1}}}$ to the old one plus s_i. □

Proposition 5.2 (Operational soundness)**.** *Let* P *be a* π_a*-process and* Q *be an* MA *process such that* $[\![P]\!] \longmapsto^n Q$*. Then,* $P \longmapsto^n_{\boxed{1}} P'$*, for some* π_a*-process* P' *such that* $Q \Longmapsto \simeq [\![P']\!]$.

Proof. By Lemma 5.1,

$$Q \equiv (\nu\widetilde{m})\left([\![R]\!] \mid \prod_{a\in\mathcal{N}} \left(\prod_{i=1}^{n^a_{\boxed{2s}}-s(a)} \mathtt{PR}_{a,x_i,P_i} \mid \prod_{k=1}^{13} \prod_{i=1}^{\widehat{n^a_{\boxed{k}}}} \mathtt{PR}^{\boxed{k},\,s_i}_{a,b_i,x_i,P_i} \mid \prod_{i=1}^{n^a_{\boxed{14}}} \mathtt{Garb}_a \right) \right)$$

Then, reduce every $\mathtt{PR}^{\boxed{k},\,s_i}_{a,b_i,x_i,P_i}$ for $k > 1$; we now obtain a process structurally equivalent to

$$(\nu\widetilde{m})\left([\![R']\!] \mid \prod_{a\in\mathcal{N}} \left(\prod_i \mathtt{PR}^{\boxed{1},\,0}_{a,b_i,x_i,P_i} \mid \prod_i \mathtt{Garb}_a \right) \right)$$

(we have omitted the indexes of the products for the sake of simplicity). Now reduce every $\mathtt{PR}^{\boxed{1},\,0}_{a,b_i,x_i,P_i}$ and obtain a process of the form

$$[\![P']\!] \mid \prod_{a\in\mathcal{N}} \prod_i \mathtt{Garb}_a$$

that is strongly barbed equivalent to $[\![P']\!]$, where P' is the π_a-process obtained from P by performing the $n_{\boxed{1}}$ communications whose encodings have lead to the production of the reductions of kind $n_{\boxed{1}}$ in $[\![P]\!] \longmapsto^n Q$. □

Divergence sensitiveness The fact that the encoding preserves divergence is a trivial corollary of operational completeness. For proving that the encoding does not introduce divergence, let us first prove the following lemma.

Lemma 5.3. *Let $[\![P]\!] \longmapsto^n$; then the number of spurious reductions is at most $\frac{n_{\boxed{1}}(n_{\boxed{1}}-1)}{2}$.*

Proof. The worst case is when all the $n_{\boxed{1}}$ reductions are on the same channel, say a, and can be obtained as follows. Put all the $n_{\boxed{1}}$ $\mathtt{p}$ ambients in the same a_2 ambient; this introduces $n_{\boxed{1}} - 1$ spurious reductions. Then, complete the first communication; this will let all the remaining $n_{\boxed{1}} - 1$ $\mathtt{p}$ ambients reappear at top-level. Now, put all of them in the same a_2 ambient; this introduces $n_{\boxed{1}} - 2$ spurious reductions. And so on. Thus, the overall number of spurious reductions is at most

$$\sum_{k=1}^{n_{\boxed{1}}} (k-1) = \frac{n_{\boxed{1}}(n_{\boxed{1}}-1)}{2}$$

□

Proposition 5.4 (Divergence reflection)**.** *If $[\![P]\!] \longmapsto^\omega$, then $P \longmapsto^\omega$.*

Proof. Let $[\![P]\!] \longmapsto^n$ and observe that $n > 0$ implies that $n_{\boxed{1}} > 0$. Moreover, for every $k \in \{2, \ldots, 14\}$, it holds that $n_{\boxed{k}} \leq n_{\boxed{1}}$; indeed, by construction of the encoding, it is not possible to produce a reduction of kind $\boxed{k}$ without having produced a corresponding reduction of kind $\boxed{1}$. By Lemma 5.3, $n \to \infty$ implies that $n_{\boxed{1}} \to \infty$; by Proposition 5.2, we easily conclude. □

Success sensitiveness Let $P \searrow$, i.e. $P \Longmapsto P' \equiv P'' \mid \surd$. Since reductions are closed under structural equivalence, $P \Longmapsto P'' \mid \surd$. By operational completeness, $[\![P]\!] \Longmapsto T \simeq [\![P'' \mid \surd]\!] \triangleq [\![P'']\!] \mid \surd$. Since we have assumed that $\simeq$ is sensible to success, this imples that $T \searrow$; thus, $[\![P]\!] \searrow$.

Conversely, let $[\![P]\!] \searrow$, i.e. $[\![P]\!] \Longmapsto T \mid \surd$. By what we have shown in Proposition 5.2, it holds that $T \mid \surd \Longmapsto [\![P']\!] \mid \prod_{a\in\mathcal{N}} \prod_i \mathtt{Garb}_a$, for some P' such that $P \Longmapsto P'$. Since $\surd$ cannot disappear along reductions, it must be that $[\![P']\!] \equiv [\![P'']\!] \mid \surd \triangleq [\![P'' \mid \surd]\!]$. Thus, $P \searrow$.

References

[1] R. Amadio. On modelling mobility. *Theoretical Computer Science*, 240(1):147–176, 2000.

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

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

[4] 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.

[5] 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.

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

[7] M. Bugliesi, G. Castagna, and S. Crafa. Access Control for Mobile Agents: the Calculus of Boxed Ambients. *ACM Trans. on Programming Languages and Systems*, 26(1):57–124, 2004.

[8] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication and Mobility Control in Boxed Ambients. *Information and Computation*, 202(1):39–86, 2005.

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

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

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

[12] L. Cardelli. Abstractions for Mobile Computation. In *Secure Internet Programming*, volume 1603 of *LNCS*, pages 51-94. Springer, 1999.

[13] L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for mobile ambients. In *Proc. of ICALP*, vol. 1644 of *LNCS*, pages 230–239. Springer, 1999.

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

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

[16] S. Dal Zilio. Mobile Processes: A Commented Bibliography. Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes, volume 2067 of *LNCS*, pages 206–222. Springer, 2001.

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

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

[19] C. Fournet, G. Gonthier, J.J. Lévy, L. Maranget and D. Rémy. A Calculus of Mobile Agents. Proc. of *CONCUR*, volume 1119 of *LNCS*, pages 406–421. Springer, 1996.

[20] C. Fournet, J.-J. Lévy, and A. Schmitt. An asynchronous, distributed implementation of mobile ambients. In *Proc. of IFIP TCS*, volume 1872 of *LNCS*, pages 348–364. Springer, 2000.

[21] P. Giannini, D. Sangiorgi, and A. Valente. Safe ambients: Abstract machine and distributed implementation. *Science of Computer Programming*, 59(3):209–249, 2006.

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

[23] D. Gorla. On the relative expressive power of ambient-based calculi. *Proc. of TGC'08*, volume 5474 of *LNCS*, pages 141–156. Springer, 2009.

[24] D. Gorla. On the relative expressive power of calculi for mobility. *Proc. of MFPS*, ENTCS 249:269–286. Elsevier, 2009.

[25] D. Gorla. Towards a Unified Approach to Encodability and Separation Results. *Proc. of CONCUR'08*, volume 5201 of *LNCS*, pages 492–507. Springer, 2008.

[26] M. Hennessy. A Distributed Pi-calculus. Cambridge University Press, 2007.

[27] M. Hennessy, M. Merro, and J. Rathke. Towards a behavioural theory of access and mobility control in distributed systems. *Theoretical Computer Science*, 322(3):615–669, 2004.

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

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

[30] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communication. In *Proc. of ECOOP*, volume 512 of *LNCS*, pages 133–147. Springer, 1991.

[31] A. Igarashi and N. Kobayashi. A generic type system for the pi-calculus. In *Proc. of POPL*, pages 128–141. ACM, 2001.

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

[33] F. Levi and D. Sangiorgi. Mobile safe ambients. *ACM Transactions on Programming Languages and Systems*, 25(1):1–69, 2003.

[34] S. Maffeis and I. Phillips. On the computational strength of pure ambient calculi. *Theoretical Computer Science*, 330(3):501–551, 2005.

[35] M. Merro and M. Hennessy. A Bisimulation-based Semantic Theory of Safe Ambients. *ACM Transactions on Programming Languages and Systems*, 28(2):290–330, 2006.

[36] M. Merro and V. Sassone. Typing and subtyping mobility in boxed ambients. *Proc. CONCUR*, vol. 2421 of *LNCS*, pages 304-320. Springer, 2002.

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

[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*, volume 623 of *LNCS*, pages 685–695. Springer, 1992.

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

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

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

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

[46] A. T. Phillips, N. Yoshida, and S. Eisenbach. A distributed abstract machine for boxed ambient calculi. In *Proc. of ESOP*, volume 2986 of *LNCS*, pages 155–170. Springer, 2004.

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

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

[49] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

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

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

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

[53] P. Sewell. Global/Local Subtyping and Capability Inference for a Distributed pi-calculus. Proc. of *ICALP*, volume 1443 of *LNCS*, pages 695–706. Springer, 1998.

[54] P. Zimmer. On the Expressiveness of Pure Safe Ambients. *Mathematical Structures in Computer Science*, 13:721–770, 2003.