Technical Report 09/2006, Dip. di Informatica, Univ. di Roma “La Sapienza”

Comparing Calculi for Mobility via their Relative Expressive Power

Daniele Gorla
Dip. di Informatica, Univ. di Roma “La Sapienza”

Abstract

In this paper, we comparatively analyze some mainstream calculi for mobility: asynchronous π-calculus, distributed π-calculus, a distributed version of LINDA and Mobile/Boxed/Safe ambients. In particular, we focus on their relative expressive power, i.e. we try to encode one in the other while respecting some reasonable properties. According to the possibility or the impossibility for such results, we set up a hierarchy of these languages. Finally, we discuss and compare some variants of ambient-like languages, including objective moves, passwords and different semantics for the mobility primitives and for parent-child communications.

Contents

1 Introduction **2**

2 The Process Calculi **4**
2.1 The asynchronous π-calculus (π_a-calculus) 5
2.2 Distributed π-calculus (Dπ) 6
2.3 micro KLAIM (μKLAIM) 7
2.4 Mobile Ambients (MA) 8
2.5 Safe Ambients (SA) 8
2.6 Boxed Ambients (BA) 9

3 Properties of Encodings **9**
3.1 Derived Properties 10

4 The Hierarchy, bottom-up **12**
4.1 Technical Preliminaries 12
4.2 Dπ and BA are more expressive than π_a-calculus 13
4.3 MA is more expressive than π_a-calculus 13
4.4 SA is more expressive than MA 14
4.5 μKLAIM is more expressive than Dπ 15
4.6 Further Impossibility Results 16

5 On the Variety of Ambient-like Languages **17**
5.1 Subjective vs Objective moves in MA 18
5.2 Adding passwords to SA 20
5.3 Shared vs Localized Channels in BA 23
5.4 Alternative Mobility Primitives in BA: SBA and NBA 25

6 Conclusions and Related Work **27**

A Properties of the encoding of π_a-calculus in MA **29**

1 Introduction

In the last ten 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 fortunate research line (network-aware programming, WAN computing, global computing, ...) that is now a well-established field for many computer scientists around the world.

In this scenario, the term *mobility* has become the reference keyword to denote several possible dynamic evolutions of systems, ranging from name mobility to mobile computation and mobile computing. *Name mobility* has been coined for the π-calculus [37], where a collection of concurrent processes communicate through named channels and the communicated objects are channel names as well; 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. The term *mobile computation* has been used to denote those languages where the net structure (seen as a collection of network nodes) is visible to the processes running in the system and is exploited by the processes to move across the net, i.e. to migrate from one node to another. Finally, the term *mobile computing* has been used to denote languages in which network nodes can move, together with the processes and data they host.

Different kinds of mobility stress different features of the system modeled. For example, with name mobility, the scope of a name shrinks and widens during computations; thus, one desirable property is to control when and how processes can access a certain channel. With mobile computations, the process allocation in the nodes of the net varies in time; hence, one may want to control where a process migrate and which resources of the new node it exploits. Finally, with mobile computing, the position of nodes within the net changes; so, a primary need is to control node movements to ensure, e.g., that some desired structural properties of the net are respected. All these checks have lead to more and more sophisticated type systems ([44, 27, 23, 14, 10, 9, 30, 5, 33], just to cite a very few samples); truly speaking, the development of powerful type theories has longly been the primary research topic on calculi for mobility.

More recently, calculi for mobility have also been the workbench of orthogonal research lines, like the development of good implementations of new programming paradigms [45, 17, 19, 3, 41] and of easy-to-handle proof-techniques for behavioural properties of systems [32, 22, 30, 6, 34]. From the practical side, we would need real-life applications where the distinctive features of such formalisms are essential. From the theoretical side, what is still lacking is an exhaustive comparative analysis of all these proposals, from a linguistic perspective; in particular, few formal results have been proved about the inter-relationships between the different languages and paradigms.

In this paper, we approach this problem by comparing some mainstream calculi for mobility: asynchronous π-calculus (written π_a-calculus) [25, 4], distributed π-calculus (written Dπ) [23], a distributed version of Linda (called μKlaim) [15] and Mobile/Boxed/Safe ambients (written MA/BA/SA) [11, 5, 30]. Our results formally prove some claims informally appeared in literature and prove in a different ways 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, neither 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 clear hierarchy, based on their expressiveness.

To this aim, the first crucial decision we had to take was the criterion to evaluate the expressive power of the languages considered. A too liberal criterion would lead to poorly informative results: most (if not all) of the languages would satisfy it. But also a too stringent criterion would be fruitless: (almost) none of the languages would satisfy it. A good compromise seems to be the notion of *relative expressive power*: this criterion relies on the possibility/impossibility of translating one language in 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* (the encoding of a compound term must be expressed in terms of the encoding of its components) and *faithfulness* (the encoding of a term must have exactly the same functionalities as the original term). There are different ways to formalize these notions; mainly for the second one, a number of different proposals have been considered in literature (e.g., sensitiveness to barbs/divergence/deadlock, operational correspondence, full abstraction, ...). Here, we consider the proposal of [21] and consider only the encodings that satisfy the following properties: *compositionality*, *name invariance* (i.e., the encodings of two source processes that differ only in their free names must only differ in the associated free names), *operational correspondence* (i.e., computations of a source term must correspond to computations in the encoded term, and vice versa), *divergence sensitiveness* (i.e., non-terminating processes must be translated into non-terminating processes, and vice versa) and *success sensitiveness* (i.e., successfully terminated terms must be translated into successfully terminated terms, and vice versa). We think that these criteria form a valid proposal for language comparison; indeed, all the best known encodings respect them (so our notion is consistent with the common understanding of the community), but there still exist encodings in 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.

In the first part of the paper, we compare π_a-calculus, Dπ, μKlaim, MA, BA and SA; our results are summarized in Figure 1. There, we write $\mathcal{L}_1 \longrightarrow \mathcal{L}_2$ if $\mathcal{L}_1$ can be encoded in $\mathcal{L}_2$ but not vice versa. We say that $\mathcal{L}_2$ is *more expressive than* $\mathcal{L}_1$ if $\mathcal{L}_1 \longrightarrow \ldots \longrightarrow \mathcal{L}_2$; $\mathcal{L}_1$ and $\mathcal{L}_2$ are *incomparable* if neither $\mathcal{L}_1$ is more expressive than $\mathcal{L}_2$ nor vice versa.

Some of our results are expectable: for example, we confirm that π_a-calculus is the minimal common denominator of calculi for mobility, since it can be encoded in all the languages considered. Some other results, though expectable, turned out very difficult to prove. For example, to encode π_a-calculus in MA we had to develop quite a complex encoding since one of our criteria is operational correspondence: what we propose is, to the best of our knowledge, the first encoding that does not introduce 'spurious' computations in the encoding of a π_a-calculus process). Ruling out computations that are not present in the source process is a sensible task when dealing with MA, because of the high possi-

$$
\begin{tikzcd}
\mu\textsc{Klaim} & \mathrm{SA} & \\
\mathrm{D}\pi \arrow[u] & \mathrm{MA} \arrow[u] \arrow[r, "??", "/"{anchor=center, sloped}] & \mathrm{BA} \\
 & \pi_a\text{-calculus} \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}
$$

Figure 1: The Main Hierarchy of Calculi for Mobility

bility of interferences between MA processes. A simpler encoding of π_a-calculus is possible, e.g., in SA (see [30]), because the latter language is “more controlled” than MA. Another issue that turned out surprisingly difficult to prove is the encodability of MA in BA, that we believe *not to* hold; this is a conjecture that this paper leaves open.

In the second part of the paper, we throughly compare several dialects of ambient-based calculi; in particular, we consider objective mobility actions in MA, the introduction of passwords in SA and some possible alternative mobility and communication primitives for BA. In some cases, we discover that the dialect proposed is comparable, in terms of expressive power, with the language it comes from; in other cases, we discover that the dialect and its original language are incomparable, i.e. no relative encoding exists. In the latter case, 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. 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 feature of the original language gets lost, thus making also the converse encoding impossible.

This paper is organized as follows. In Section 2, we formally present syntax and operational semantics of the six languages depicted in Figure 1. In Section 3, we recall from [21] the properties that encodings should satisfy. In Section 4, we formally build up the hierarchy of Figure 1: for every pair of languages, we give a formal proof of encodability/non-encodability. In Section 5, we enrich the hierarchy of Figure 1 with further languages that are variants of ambient-like languages. Finally, in Section 6 we conclude the paper by also mentioning some related work.

2 The Process 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:

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

  As usual, $\mathbf{0}$ is the terminated process, whereas $\surd$ denotes success (see the discussion on Property 5 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 recursive processes; all what we are going to prove does 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*, 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 and '$\equiv$' denotes structural equivalence (used to equate different ways of writing the same process). Of course, the operational axioms, the evaluation contexts and strutural equivalence are peculiar to every language. As usual, $\Longmapsto$ denotes the reflexive and transitive closure of $\longmapsto$.

- $\simeq$ is a behavioural equivalence/preorder, needed to describe the abstract behaviour of a process; usually, $\simeq$ is a congruence at least with respect to parallel composition.

We now present the syntax and reduction-based operational semantics of the specific calculi considered in this paper. 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-calculus; channels, localities and variables in Dπ; localities and variables in μKlaim; ambients and variables in MA, SA and BA).

2.1 The asynchronous π-calculus (π_a-calculus)

We consider the asynchronous version of the π-calculus, as defined in [4]. 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 good implementation [45]. 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.

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 described above 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 [23]; mainly, we elided typing information from the syntax. The main syntactic entity are *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-calculus 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 nets 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 [23], this check was carried out, among other tasks, 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 micro KLAIM (μKLAIM)

μKLAIM [15] is a core calculus at the basis of the KLAIM language [13], a distributed version of LINDA [18]. Its syntax is similar to that of Dπ with the difference that now nodes hosts *components*, i.e. processes or tuples of data, denoted by $\langle \ldots \rangle$; indeed, similarly to LINDA, communication in μKLAIM is not channel-based but it relies on the notion of *(distributed) tuple spaces*.

$$N ::= \mathbf{0} \;\big|\; l : C \;\big|\; N|N \;\big|\; (\nu l)N \qquad\qquad C ::= \langle \widetilde{u} \rangle \;\big|\; P \;\big|\; C_1|C_2$$

$$P ::= \ldots \;\big|\; \mathbf{in}(\widetilde{T})@u.P \;\big|\; \mathbf{rd}(\widetilde{T})@u.P \;\big|\; \mathbf{out}(\widetilde{u})@u.P \;\big|\; \mathbf{eval}(P)@u.P \qquad T ::= u \;\big|\; \ulcorner u \urcorner$$

where we denote by $\widetilde{\_}$ a (possibly empty) sequence of elements of kind $\_$.

Actions in μKLAIM access possibly remote data spaces by producing data (viz. action **out**), consuming data (viz. action **in**) or reading data (viz. action **rd**), and spawn processes at a possibly remote locality (viz. action **eval**). When accessing data (via **in** or **read**), it is possible to either specify a name that must be present in the datum accessed (via parameters of kind $\ulcorner u \urcorner$) or read a new name that will be replaced in the continuation process (via parameters of kind u, that play the same rôle as input variables of π_a-calculus and Dπ).

Names in μKLAIM can be bound in four ways: either by $(\nu l)N$ and $(\nu l)P$, that bind l in N and P, or by $\mathbf{in}(\ldots, x, \ldots)@u.P$ and $\mathbf{rd}(\ldots, x, \ldots)@u.P$, that bind x in P (in this case, x is a *formal* input parameter); free names are defined accordingly. In particular, notice that parameters of the form $\ulcorner u \urcorner$ do not bind u in the continuation: they are *actual* input parameters that must be exactly matched when retrieving a datum (this corresponds to a LINDA-like pattern matching).

Evaluation contexts and structural equivalence are defined like for Dπ, with C in place of P everywhere. The reduction axioms are:

$$\begin{array}{ll}
l : \mathbf{in}(\widetilde{T})@l'.P \mid l' : \langle \widetilde{l} \rangle \longmapsto l : P\sigma \mid l' : \mathbf{0} & \text{if } \mathrm{M}(\widetilde{T}; \widetilde{l}) = \sigma \\
l : \mathbf{rd}(\widetilde{T})@l'.P \mid l' : \langle \widetilde{l} \rangle \longmapsto l : P\sigma \mid l' : \langle \widetilde{l} \rangle & \text{if } \mathrm{M}(\widetilde{T}; \widetilde{l}) = \sigma \\
l : \mathbf{out}(\widetilde{l})@l'.P \mid l' : \mathbf{0} \longmapsto l : P \mid l' : \langle \widetilde{l} \rangle & \\
l : \mathbf{eval}(P')@l'.P \mid l' : \mathbf{0} \longmapsto l : P \mid l' : P' &
\end{array}$$

In the first two axioms, $P\sigma$ denotes the capture-avoiding application of substitution σ to P; σ results from the pattern-matching function, $\mathrm{M}(\widetilde{T}; \widetilde{l})$, defined as follows:

$$\mathrm{M}(x; l) = \{^l/_x\} \qquad\qquad \mathrm{M}(\,;\,) = \mathrm{M}(\ulcorner l \urcorner; l) = \epsilon \qquad\qquad \frac{\mathrm{M}(T; l) = \sigma_1 \qquad \mathrm{M}(\widetilde{T}; \widetilde{l}) = \sigma_2}{\mathrm{M}(T, \widetilde{T}; l, \widetilde{l}) = \sigma_1 \uplus \sigma_2}$$

with 'ϵ' being the empty substitution and '$\uplus$' denoting the union of partial functions with disjoint domains.

Actions **in** and **read** try to access a remote datum $\langle \widetilde{l} \rangle$ matching the parameters $\widetilde{T}$ argument of the actions; if such a datum exists, the first action removes it, whereas the second action leaves it in the remote locality; if no matching datum exists, both actions are suspended. Intuitively, $\widetilde{l}$ matches against $\widetilde{T}$ if they have the same number of fields and corresponding fields match, where x matches any name l whereas $\ulcorner l \urcorner$ matches only l.

2.4 Mobile Ambients (MA)

We consider the Ambient calculus as presented in [11].

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

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-calculus 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:

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

MA, like all the following Ambient-like languages, strongly relies on a type system to avoid inconsistent processes like, e.g., $m.P$ or $in\_n[P]$; these two processes can arise after the (ill-typed) communications $(x).x.P \mid \langle m\rangle$ and $(x).x[P] \mid \langle in\_n\rangle$. For MA, like for SA and BA, we shall always consider the sub-language formed by all the well-typed processes, as defined in [10, 30, 5].

2.5 Safe Ambients (SA)

We consider the Safe Ambient calculus as presented in [30]. SA extends MA by adding *co-actions*, though which ambient movements/openings must be authorised 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:

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

2.6 Boxed Ambients (BA)

We consider the Boxed Ambient calculus as presented in [5]. BA evolves MA by removing the *open* action that is considered too powerful and, hence, potentially dangerous. 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 $\hat{}$) or towards an enclosed ambient n (if tagged with direction n).

$$P \;::=\; \ldots \;\big|\; (x)^\eta.P \;\big|\; \langle M\rangle^\eta.P \;\big|\; M.P \;\big|\; u[P]$$

$$M \;::=\; u \;\big|\; in\_u \;\big|\; out\_u \;\big|\; M.M \qquad\qquad \eta ::= \star \;\big|\; \hat{} \;\big|\; u$$

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

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

3 Properties of Encodings

A *translation* of $\mathcal{L}_1 = (\mathcal{P}_1, \longmapsto_1, \simeq_1)$ into $\mathcal{L}_2 = (\mathcal{P}_2, \longmapsto_2, \simeq_2)$, written $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$, is a function from $\mathcal{P}_1$ to $\mathcal{P}_2$. We shall call *encoding* any translation that 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., $\mathcal{L}_2$).

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 replaced by the k holes $\_{}_1, \ldots, \_{}_k$.

Property 1. *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* compositional *if, for every k-ary* $\mathcal{L}_1$*-operator* op *and finite subset of names N, there exists a k-ary 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. However, it is possible that an encoding fixes some names to play a precise rôle or it can map a single name into a tuple of names. In general, every encoding assumes a *renaming policy* $\varphi_{[\![\,]\!]} : \mathcal{N} \longrightarrow \mathcal{N}^k$ that is a function such that $\forall u, v \in \mathcal{N}$ with $u \neq v$, it holds that $\varphi_{[\![\,]\!]}(u) \cap \varphi_{[\![\,]\!]}(v) = \emptyset$ (where $\varphi_{[\![\,]\!]}(\cdot)$ is simply considered a set here). We extend the application of a substitution to sequences of names in the expected way.

Property 2. *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *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 can fuse two distinct names, and this matters because compositionality also depends on the free names occurring in the encoded terms. For more discussion, see [21].

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. *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* operationally corresponding *if*

- *for every* S *and* S' *such that* $S \Longmapsto_1 S'$*, it holds that* $[\![S]\!] \Longmapsto_2 \simeq_2 [\![S']\!]$*;*
- *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 $\longmapsto^{\omega}$.

Property 4. *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* divergence reflecting *whenever* $[\![S]\!] \longmapsto_2^{\omega}$ *implies that* $S \longmapsto_1^{\omega}$*, 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 sensible semantic theories [16, 47]. To formulate our property in a simple way, we follow the approach in [47] and assume that all the languages contain the same success process $\surd$; then, we define the predicate $\Downarrow_{SUCC}$, meaning reducibility (in some modality, e.g. may/must/fair-must) to a process containing a top-level unguarded occurrence of $\surd$. Clearly, different modalities in general lead to different results; in this paper, proofs will be carried out in a 'may' modality, but all our results could be adapted to other modalities. Finally, for the sake of coherence, we require the notion of success be caught by the semantic theory underlying the calculi, viz. $\simeq$; in particular, we assume that $\simeq$ never relates two processes P and Q such that $P \Downarrow_{SUCC}$ and $Q \not\Downarrow_{SUCC}$.

Property 5. *A translation* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *is* success sensitive *if, for every* S*, it holds that* $S \Downarrow_{SUCC}$ *iff* $[\![S]\!] \Downarrow_{SUCC}$.

3.1 Derived Properties

In [21] we have shown that some separation result can be proved in the general framework we have just presented. However, to carry out more proofs, we have to slightly specilise 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 $(\mathcal{E}(T), T)$, for $\mathcal{E}(T) \simeq_2 T$.

All these assumptions are discussed and justified at length in [21]. By relying on them, we can prove a number of auxiliary results that will be useful in carrying out the main proofs of the paper.

Proposition 3.1. *Let* $[\![\cdot]\!]$ *be an encoding; then,* $S \not\longmapsto_1$ *implies that* $[\![S]\!] \not\longmapsto_2$.

Proposition 3.2. *Let* $[\![\cdot]\!]$ *be an encoding; if there exist two source terms* S_1 *and* S_2 *such that* $S_1 \mid S_2 \Downarrow_{SUCC}$, $S_1 \not\Downarrow_{SUCC}$ *and* $S_2 \not\Downarrow_{SUCC}$, *then* $[\![S_1 \mid S_2]\!] \longmapsto_2$.

Proposition 3.3. *Let* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *be an encoding. If there exists two source terms* S_1 *and* S_2 *that do not reduce but such that* $[\![S_1 \mid S_2]\!] \longmapsto$, *then*

1. *if* $\mathcal{L}_2 \in \{\pi_a$*-calculus,* $\mathrm{D}\pi\}$, *it can only be that* $[\![S_1]\!] \mid [\![S_2]\!] \longmapsto$;

2. *if* $\mathcal{L}_2 \in \{\mathrm{MA, BA, SA}\}$, *it can only be that* $C_1([\![S_1]\!]) \mid C_2([\![S_2]\!]) \longmapsto$, *where* $C_|^{fn(S_1,S_2)}(\_1;\_2)$, *i.e. the context used to compositionally translate* $S_1 | S_2$, *is structurally equivalent to* $\mathcal{E}(C_1(\_1) \mid C_2(\_2))$ *for some evaluation context* $\mathcal{E}(\cdot)$ *and two contexts* $C_1(\cdot)$ *and* $C_2(\cdot)$ *that are either empty (viz.,* $\cdot$ *) or a single top-level ambient containing a top-level hole (viz.,* $m[\,\cdot\,]$*).*

Theorem 3.4. *Assume that there is a* $\mathcal{L}_1$*-process* S *such that* $S \not\longmapsto_1$, $S \not\Downarrow_{SUCC}$ *and* $S \mid S \Downarrow_{SUCC}$*; moreover, assume that every* $\mathcal{L}_2$*-process* T *that does not reduce is such that* $T \mid T \not\longmapsto_2$. *Then, there cannot exist any encoding* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.

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 $\mathcal{L}$. For example, the matching degree of Mobile Ambients is 1, whereas the matching degree of $\mathrm{D}\pi$ is 2.

Theorem 3.5. *If* $\mathrm{MD}(\mathcal{L}_1) > \mathrm{MD}(\mathcal{L}_2)$, *then there exists no encoding* $[\![\cdot]\!] : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.

Another derived property, not needed for the results in [21], is the following one.

Proposition 3.6. *Let* $[\![\cdot]\!] : \mathcal{L}_1 \to \mathcal{L}_2$ *be a translation 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 Hierarchy, bottom-up

For every pair of languages, we see whether one is more expressive than the other, or if they are incomparable. In the first case, we provide an 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 encoding of one in the other exists.

We now give the crucial results underlying the hierarchy in Figure 1. The remaining pairs of languages can be compared either by transitivity of the encodability relation (for the encodings we are going to develop, it holds that the composition of two encodings is still an encoding), or by contradiction with one of the impossibility results we are going to prove.

4.1 Technical Preliminaries

To carry out proofs, we found it convenient to exploit the labelled transition systems developed for some of the languages studied. For space limitations, we cannot give here a full account on this topic; thus, we informally present only the technicalities strictly needed in our proofs and refer the interested reader to [34, 30, 6] for full details and for formal proofs.

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

$$\begin{array}{ll} 1.\ P_1 \longmapsto & 3.\ P_1 \xrightarrow{enter\_n} \text{ and } P_2 \xrightarrow{amb\_n} \\ 2.\ P_1 \xrightarrow{\langle - \rangle} \text{ and } P_2 \xrightarrow{(M)} & 4.\ P_1 \xrightarrow{open\_n} \text{ and } P_2 \xrightarrow{amb\_n} \end{array}$$

where $P \xrightarrow{\langle - \rangle}$ *means that* P *has some top-level datum,* $P \xrightarrow{(M)}$ *means that* P *has a top-level input action, ready to receive any message* M*,* $P \xrightarrow{amb\_n}$ *means that* P *has a top-level ambient named* n*,* $P \xrightarrow{enter\_n}$ *means that* P *has a top-level ambient containing a top-level prefix* in_n *and* $P \xrightarrow{open\_n}$ *means that* P *has a top-level prefix* $open\_n$*.*

Proposition 4.2 (Labeled actions for SA)**.** *In* SA*, it holds that* $P_1 \mid P_2 \longmapsto$ *if and only if one of the following conditions hold (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* $P_2 \xrightarrow{?open\_n}$

where $P \xrightarrow{\mu}$ *, for* $\mu \in \{?enter\_n, ?open\_n\}$*, means that* P *has a top-level ambient named* n *containing a top-level prefix* $\overline{in}\_n$ *or* $\overline{open}\_n$*.*

Proposition 4.3 (Labeled actions for BA)**.** *In* BA*, it holds that* $P_1 \mid P_2 \longmapsto$ *if and only if one of the following conditions hold (possibly with* P_1 *and* P_2 *swapped):*

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

$$\begin{array}{ll} 4.\ P_1 \xrightarrow{\langle - \rangle^\star} \text{ and } P_2 \xrightarrow{up(M)} & 6.\ P_1 \xrightarrow{\langle - \rangle^n} \text{ and } P_2 \xrightarrow{n(M)} \\ 5.\ P_1 \xrightarrow{(M)^\star} \text{ and } P_2 \xrightarrow{up\langle - \rangle} & 7.\ P_1 \xrightarrow{(M)^n} \text{ and } P_2 \xrightarrow{n\langle - \rangle} \end{array}$$

where $P \xrightarrow{\mu}$ *, for* $\mu \in \{\langle - \rangle^\star, (M)^\star, \langle - \rangle^n, (M)^n\}$*, means that P has a top-level action* $\langle M \rangle^\star$*,* $(x)^\star$*,* $\langle M \rangle^n$ *or* $(x)^n$*;* $P \xrightarrow{\mu}$ *, for* $\mu \in \{up\langle - \rangle, up(M)\}$*, means that P has a top-level ambient containing the top-level action* $\langle M \rangle^{\hat{}}$ *or* $(x)^{\hat{}}$*;* $P \xrightarrow{\mu}$ *, for* $\mu \in \{n\langle - \rangle, n(M)\}$*, means that P has a top-level ambient named n containing the top-level action* $\langle M \rangle^\star$ *or* $(x)^\star$*.*

4.2 Dπ and BA are more expressive than π_a-calculus

Clearly, π_a-calculus can be trivially encoded in Dπ: it suffices to locate the π_a-calculus process in a reserved locality hosting all the channels needed. On the contrary, Dπ cannot be encoded in π_a-calculus, as a corollary of the non-encodability of Dπ in BA (see Theorem 4.8 later on) and of the encodability of π_a-calculus in BA [5]. The latter result is proved by the encoding defined as a homomorphism w.r.t. all the operators, except for

$$\begin{array}{rcll} [\![u(x).P]\!] & \triangleq & (x)^u.[\![P]\!] & \\ [\![\bar{u}\langle v \rangle]\!] & \triangleq & (\nu k)(u[\langle v \rangle^\star.in\_k] \mid k[\mathbf{0}]) & \qquad \text{for } k \text{ fresh} \end{array}$$

Also the (choice-free) synchronous π-calculus can be encoded in BA: it suffices to exploit the encoding of the (choice-free) synchronous π-calculus in π_a-calculus developed in [4]. The fact that BA cannot be encoded in π_a-calculus is proved in the following result.

Theorem 4.4. *There exists no encoding of* BA *in* π_a*-calculus.*

Proof. Corollary of Theorem 3.4:

- On one hand, notice that, if T is a π_a-calculus-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, $T \longmapsto_2$; hence, every π_a-calculus-process T that does not reduce is such that $T \mid T \not\longmapsto_2$.
- On the other hand, we can find in BA a process S that does not reduce and does not report success, but such that $S \mid S$ reports success: it suffices to let S be $(\nu p)(open\_p.\surd \mid n[in\_n.p[out\_n.out\_n]])$. ◻

4.3 MA is more expressive than π_a-calculus

First, notice that MA cannot be encoded in π_a-calculus, as proved in Theorem 4.4 (the proof of such result scales well to MA too). We are left with proving that π_a-calculus can be encoded in MA; this is not a trivial task, if we want to satisfy all the properties in Section 3. Indeed, in several papers [11, 10, 9] there are attempts to encode π_a-calculus in MA, but none of them satisfies operational completeness. To the best of our knowledge, the encoding we are going to present is the first one that fully satisfies operational correspondence.

The encoding relies on a renaming policy that maps every name a to a triple of pairwise different names (a_1, a_2, a_3); it is a homomorphism w.r.t. all the operators, except for restrictions, inputs and outputs, that are translated as follows:

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

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

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

The encoding just presented satisfies all the properties of Section 3. The interested reader can find the (non-trivial) details of this proof in Appendix A.

4.4 SA is more expressive than MA

In [30] 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 does not exactly enjoy all the properties listed in Section 3. 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.

This problem can be fixed in two ways. The first way consists in accepting a weaker formulation of operational correspondence, that only holds up to strong barbed equivalence restricted to translated contexts (written $\simeq^{tr}$). To this aim, 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 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

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

is a barbed bisimulation. We distinguish whether the hole is immediately contained in an ambient u or not. In the first case, $C(\cdot) \equiv \mathcal{D}(u[\cdot \mid P])$, 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, $C(P_u) \downarrow$ iff $C(\mathbf{0}) \downarrow$. 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 $(C'(P_u), C'(\mathbf{0})) \in \Re$, 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.

A second way to fix the problem of the translation given in [30] is to consider a *family* of encodings $[\![\cdot]\!]_N$, for $N \subset \mathcal{N}$, with the idea that a MA process P can be encoded via $[\![\cdot]\!]_N$ only if $fn(P) \subseteq N$. For every N, $[\![\cdot]\!]_N$ is a homomorphism for all operators, except for

$$[\![\mathbf{0}]\!]_N \triangleq P_N \qquad [\![u[P]]\!]_N \triangleq u[P_N \mid [\![P]\!]_N]$$

$$[\![(\nu n)P]\!]_N \triangleq (\nu n)[\![P]\!]_{N\cup\{n\}} \qquad [\![(x).P]\!]_N \triangleq (x).[\![P]\!]_{N\cup\{x\}}$$

where $P_N \triangleq \prod_{n\in N} P_n$. By exploiting the equivalence $!P \simeq !P \mid !P$, it is easy to check that now operational correspondence holds up to $\simeq$. It is however worth noting that name invariance must be used with some care: for $[\![\cdot]\!]_N$, it makes only sense to use substitutions whose domain and range are contained in N.

We now prove that SA cannot be encoded in MA.

Theorem 4.5. *There exists no encoding of* SA *in* MA.

Proof. By contradiction. Consider the pair of SA processes $P \triangleq n[in\_n.\langle m\rangle]$ and $Q \triangleq n[\overline{in}\_n.(m[out\_n.\overline{open}\_m.\surd] \mid \overline{out}\_n)] \mid open\_m$, 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 $C_2([\![Q]\!]) \xrightarrow{\alpha}$, for $\alpha \in \{enter\_n', open\_n'\}$
2. or $C_2([\![Q]\!]) \xrightarrow{amb\_n'}$ and $C_1([\![P]\!]) \xrightarrow{\alpha}$, for $\alpha \in \{enter\_n', open\_n'\}$.

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 $[\![Q]\!] \xrightarrow{(M)}$, otherwise, by Property 2, $[\![m[in\_m.\langle n\rangle] \mid Q]\!]$ 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 $[\![n[out\_n.\langle m\rangle] \mid Q]\!] \longmapsto$ (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 such names). Thus, we have that $[\![P]\!] \xrightarrow{amb\_n'}$; but also this leads to a contradiction. Indeed, by Property 1, it holds that $[\![P]\!] \triangleq C_{n[\,]}^{\{n,m\}}([\![in\_n.\langle m\rangle]\!])$; so, the ambient named n' can be exhibited either by $C_{n[\,]}^{\{n,m\}}(\cdot)$ or by $[\![in\_n.\langle m\rangle]\!]$ (and, hence, $C_{n[\,]}^{\{n,m\}}(\cdot)$ has a top-level hole). In both cases, we can contradict Proposition 3.1: in the first case, we would have that $[\![n[out\_n.\langle m\rangle]]\!] \xrightarrow{amb\_n'}$ and so $[\![n[out\_n.\langle m\rangle] \mid Q]\!] \longmapsto$; in the second case, we would have that $[\![in\_n.\langle m\rangle \mid Q]\!] \longmapsto$. □

4.5 μKlaim is more expressive than Dπ

First, we prove that μKlaim cannot be encoded in Dπ.

Theorem 4.6. *There exists no encoding of* μKlaim *in* Dπ.

Proof. Corollary of Theorem 3.5, since $\textsc{Md}(\mu\textsc{Klaim}) = \infty$ whereas $\textsc{Md}(\mathrm{D}\pi) = 2$. □

It is possible to encode $\mathrm{D}\pi$ in μKlaim: indeed, channel-based communications are easy to simulate via data spaces and pattern matching, as already proved in [20]. The encoding acts homomorphically on all the operators, except for

$$\begin{aligned}
[\![\, l : P \,]\!] &\triangleq l : \mathtt{expand}_l([\![\, P \,]\!]_l) \\
[\![\, go_u.P \,]\!]_w &\triangleq \mathbf{eval}([\![\, P \,]\!]_u)@u.\mathbf{0} \\
[\![\, u(\widetilde{x}).P \,]\!]_w &\triangleq \mathbf{in}(\ulcorner u \urcorner, \widetilde{x}, y)@w.\mathbf{out}()@y.[\![\, P \,]\!]_w \qquad \text{for } y \text{ fresh} \\
[\![\, \bar{u}\langle \widetilde{v} \rangle.P \,]\!]_w &\triangleq (\nu k)\mathbf{out}(u, \widetilde{v}, k)@w.\mathbf{in}()@k.[\![\, P \,]\!]_w \qquad \text{for } k \text{ fresh}
\end{aligned}$$

where function $\mathtt{expand}_l$ turns all the top-level processes prefixed with a $\mathbf{out}(\widetilde{l})@l$ prefix into a datum $\langle \widetilde{l} \rangle$ at l's dataspace in parallel with the continuation process (this is needed to respect Proposition 3.1, e.g. in $[\![\, l : \bar{a}\langle b \rangle \,]\!]$). We leave to the interested reader the easy task of proving that this encoding enjoys all the properties listed in Section 3.

4.6 Further Impossibility Results

Proposition 4.7. *There exists no encoding of* MA *and* BA *in* μKlaim.

Proof. This is a corollary of Theorem 3.4 and the proof is similar to Theorem 4.4. □

Proposition 4.8. *There exists no encoding of* $\mathrm{D}\pi$ *in* SA *nor in* BA.

Proof. Corollary of Theorem 3.5, since $\textsc{Md}(\mathrm{D}\pi) = 2$ whereas $\textsc{Md}(\mathrm{MA}) = \textsc{Md}(\mathrm{SA}) = \textsc{Md}(\mathrm{BA}) = \textsc{Md}(\pi_a) = 1$. □

Theorem 4.9. *There exists no encoding of* BA *in* SA.

Proof. 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 $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{enter\_n'}$ and $C_2([\![\, n[\langle b \rangle^\star] \,]\!]) \xrightarrow{?enter\_n'}$
2. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{?enter\_n'}$ and $C_2([\![\, n[\langle b \rangle^\star] \,]\!]) \xrightarrow{enter\_n'}$
3. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{open\_n'}$ and $C_2([\![\, n[\langle b \rangle^\star] \,]\!]) \xrightarrow{?open\_n'}$
4. or $C_1([\![\, (x)^n.\surd \,]\!]) \xrightarrow{?open\_n'}$ and $C_2([\![\, n[\langle b \rangle^\star] \,]\!]) \xrightarrow{open\_n'}$.

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 $n' \in \varphi_{[\![\,]\!]}(n)$.

However, we now prove that all the cases depicted above lead to contradict Proposition 3.1. Let $C_2([\![\, n[\langle b \rangle^\star] \,]\!]) \xrightarrow{\alpha}$, for $\alpha \in \{?enter\_n', enter\_n', ?open\_n', open\_n'\}$. 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 α is produced by $C^{\{b\}}_{n[\,]}(\cdot)$, also $[\![\, n[\langle b \rangle^{\hat{}}] \,]\!]$ would exhibit label α; if the production of α involves $[\![\, \langle b \rangle^\star \,]\!]$, we would have that $n' \in fn([\![\, \langle b \rangle^\star \,]\!])$, in contradiction with Proposition 3.6. So, assume that $C_2(\cdot)$ is not empty; this rules out case 4 above and imposes that $[\![\, n[\langle b\rangle^\star]\,]\!] \xrightarrow{\alpha'}$, for $\alpha' \in \{\overline{in}\_n', in\_n', \overline{open}\_n'\}$. 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 α' without contradicting Proposition 3.1. $\square$

To complete the hierarchy of Figure 1, it suffices to prove that there exists no encoding of MA in BA. Surprisingly, we have not been able to prove such an expectable result; thus, we leave it open as a conjecture.

Conjecture 1. *There exists no encoding of* MA *in* BA.

However, similarly to Theorem 4.5, we can prove that BA cannot encode SA.

Theorem 4.10. *There exists no encoding of* SA *in* BA.

Proof. By contradiction. First, consider the pair of SA processes $P \triangleq n[in\_n.\langle m\rangle]$ and $Q \triangleq n[\overline{in}\_n.(m[out\_n.\overline{open}\_m.\surd] \mid \overline{out}\_n)] \mid open\_m$, for $n \neq m$; by Propositions 3.2 and 3.3, it must be that $C_1([\![\, P\,]\!]) \xrightarrow{\alpha}$ and $C_2([\![\, Q\,]\!]) \xrightarrow{\alpha'}$ where, by Proposition 4.3, it can only be that

1. $\alpha = amb\_n'$ and $\alpha' \in \{enter\_n', open\_n'\}$, or vice versa;
2. $\alpha = \langle -\rangle^{n'}$ and $\alpha' = n'(M)$, or vice versa;
3. $\alpha = (M)^{n'}$ and $\alpha' = n'\langle -\rangle$, or vice versa.

In all cases, $n' \in \varphi_{[\![\ ]\!]}(n)$. We now prove that the three cases above all lead to a contradiction: the first case is formally identical to the proof of Theorem 4.5; the second and the third case are similar, so we only work out case 2. First, notice that $C_1(\cdot)$ must be empty; so, α is produced either by $C^{\{n,m\}}_{n[\ ]}(\cdot)$ or by $[\![\, P\,]\!]$. In both cases, we can contradict Proposition 3.1: in the first case, it suffices to note that $[\![\, n[in\_m.\langle n\rangle]\,]\!] \xrightarrow{\alpha}$ and so $[\![\, n[in\_m.\langle n\rangle] \mid Q\,]\!] \longmapsto$; in the second case, we would have that $[\![\, in\_n.\langle m\rangle \mid Q\,]\!] \longmapsto$. $\square$

Thus, there are only two possibilities for resolving the '??' in Figure 1:

$$\begin{tikzcd}
 & SA & & BA \\
D\pi & & MA \arrow[ul] \arrow[ur] & \\
 & \pi_a \arrow[ul] \arrow[ur] & &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & SA & \\
D\pi & MA \arrow[u] & BA \\
 & \pi_a \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

according to whether MA is encodable in BA or not. We strongly believe that the right one should hold, even if we still have not been able to prove it.

5 On the Variety of Ambient-like Languages

The languages MA, SA and BA are just a small set of representatives among the set of ambient-like languages. A lot of small variations on these three mainstream languages appeared in literature. We want to mention here some of these variations and try to compare them with the dialects presented so far.

To prove some of the following results, we need a further property for our encodings:

Property 6 (Adequacy)**.** *An encoding* $[\![\cdot]\!]$ *is* adequate *if* $\Psi \equiv \Psi'$ *implies that* $[\![\Psi]\!] \simeq [\![\Psi']\!]$.

This property seems us quite acceptable, since the purpose of structural equivalence is relating different ways of writing the same process; thus, it is natural to require that the encoding of structurally equivalent processes behave in the same way. We could have asked for structural equivalence of the encoded terms, but, because of compositionality, this would have led to a too demanding property. It has to be said that Property 6 is quite close in spirit to the notion of *full abstraction*, whereas the proposal in [21] was defined as an alternative to such a notion. Thus, we would really like to avoid the use of Property 6; this leaves space for improving our results. Indeed, we believe that the impossibility results we are going to prove via Property 6 should hold also without it, but we have still not been able to prove them.

5.1 Subjective vs Objective moves in MA

One of the first variations of MA was already proposed in the seminal paper [11]. The idea was that the movement, instead of being *subjective* (the moving ambient decides where and when moving), could be *objective* (the moving ambient is moved from the outside). We recall here the original semantics proposed in [11]. In the *objective ambient calculus* (MA_o), actions in_n and out_n are replaced by $mv\, in\_n$ and $mv\, out\_n$, whose semantics is

$$\begin{array}{rcl} mv\, in\_n.P_1 \mid n[P_2] & \longmapsto & n[P_1 \mid P_2] \\ n[mv\, out\_n.P_1 \mid P_2] & \longmapsto & P_1 \mid n[P_2] \end{array}$$

Theorem 5.1. MA *is more expressive than* MA_o*: there exists an encoding of* MA_o *in* MA*; there exists no encoding of* MA *in* MA_o*.*

Proof. Consider the encoding of MA_o in MA provided in [11]:

$$\begin{array}{rcll} [\![n[P]]\!] & \triangleq & n'[[\![P]\!] \mid !open\_\mathtt{in}] \mid !open\_\mathtt{out} & \\ [\![mv\, in\_n.P]\!] & \triangleq & (\nu k)k[in\_n'.\mathtt{in}[out\_k.open\_k.[\![P]\!]]] & \text{for } k \text{ fresh} \\ [\![mv\, out\_n.P]\!] & \triangleq & (\nu k)k[out\_n'.\mathtt{out}[out\_k.open\_k.[\![P]\!]]] & \text{for } k \text{ fresh} \end{array}$$

where $\mathtt{in}$ and $\mathtt{out}$ are reserved names, and $n' = \varphi_{[\![\ ]\!]}(n)$. We leave to the reader the easy task of checking that this encoding satisfies all the properties listed in Section 3.

The fact that MA cannot be encoded in MA_o is a corollary of Theorem 3.4. □

Another variation of MA with objective moves is the so called *Push and Pull ambient calculus* (PAC) [43]. Now, actions in_n and out_n are replaced by $pull\_n$ and $push\_n$, whose semantics is

$$\begin{array}{rcl} n[P_1] \mid m[pull\_n.P_2 \mid P_3] & \longmapsto & m[n[P_1] \mid P_2 \mid P_3] \\ m[n[P_1] \mid push\_n.P_2 \mid P_3] & \longmapsto & n[P_1] \mid m[P_2 \mid P_3] \end{array}$$

Theorem 5.2. MA *and* PAC *are incomparable: there exists no encoding of* PAC *in* MA *and of* MA *in* PAC *that satisfy Property 6.*

Proof. For the first claim, consider the PAC process $P \mid Q$, for $P \triangleq n[pull\_m.(\langle n\rangle \mid push\_p \mid p[\surd])]$, $Q \triangleq m[\mathbf{0}] \mid open\_p$ and $n \neq m$. By Proposition 3.2, its encoding must reduce and, by Propositions 3.3 and 4.1, this can happen in one of the following ways:

- $C_1(\llbracket P \rrbracket) \xrightarrow{\langle - \rangle}$ and $C_2(\llbracket Q \rrbracket) \xrightarrow{(M)}$ (or vice versa): this is not possible otherwise, by Property 2, we would have that $\llbracket P\sigma \mid Q \rrbracket \longmapsto$, for σ the permutation swapping n and m.

- $C_1(\llbracket P \rrbracket) \xrightarrow{enter\_k}$ and $C_2(\llbracket Q \rrbracket) \xrightarrow{amb\_k}$: if $C_1(\cdot)$ was not empty, it must be that $\llbracket P \rrbracket \xrightarrow{in\_k}$; this is not possible, because otherwise either $\llbracket pull\_m.(\langle n\rangle \mid push\_p \mid p[\surd]) \mid Q \rrbracket \longmapsto$ or $\llbracket n[pull\_n.(\langle m\rangle \mid push\_p \mid p[\surd])] \mid Q \rrbracket \longmapsto$, according to whether $\llbracket pull\_m.(\langle n\rangle \mid push\_p \mid p[\surd]) \rrbracket \xrightarrow{in\_k}$ or $C^{\{n,m\}}_{n[\,]}(\cdot) \xrightarrow{in\_k}$. So, it must be that $\llbracket P \rrbracket \xrightarrow{enter\_k}$; we now prove that this implies that either $\llbracket \cdot \rrbracket$ violates Proposition 3.1 or that $\llbracket\, !P \rrbracket \xrightarrow{enter\_k}{}^{\omega}$ (and so $\llbracket\, !P \mid Q \rrbracket$ diverges, in violation with Property 4). By Proposition 3.3, we know that $C^{fn(P)}_{|}(\_1; \_2) \equiv \mathcal{E}(\_1 \mid \_2)$: indeed, we have just shown that $C_1(\cdot)$ must be empty and also $C_2(\cdot)$ must be empty, otherwise $\llbracket P \mid (\langle m\rangle \mid open\_m) \rrbracket \longmapsto$.

  If the hole in $\mathcal{E}(\cdot)$ is contained in (at least) one ambient, then either $\mathcal{E}(\cdot) \xrightarrow{enter\_k}$ (and in this case we would have that $(m[\langle n\rangle \mid \langle p\rangle] \mid m[\langle n\rangle \mid \langle p\rangle]) \mid Q \longmapsto$) or $\llbracket P \rrbracket \xrightarrow{in\_k}$, because $\llbracket (P \mid P) \mid Q \rrbracket$ must reduce. It is now easy to prove that every possible way to produce $\llbracket P \rrbracket \xrightarrow{in\_k}$ leads to contradict Proposition 3.1; so, the hole in $\mathcal{E}(\cdot)$ cannot fall in any ambient, i.e. $\mathcal{E}(\cdot) \equiv (\nu\widetilde{n})(\,\cdot \mid P)$.

  If $k \notin bn(\mathcal{E}(\cdot))$, then we can use Property 6 to state that $\llbracket\, !P \rrbracket \simeq \llbracket P \mid\, !P \rrbracket \equiv \mathcal{E}(\llbracket P \rrbracket \mid \llbracket\, !P \rrbracket) \xrightarrow{enter\_k} \mathcal{E}(K \mid \llbracket\, !P \rrbracket)$, for $\llbracket P \rrbracket \xrightarrow{enter\_k} K$.[1] But then $\mathcal{E}(K \mid \llbracket\, !P \rrbracket) \simeq \mathcal{E}(K \mid \llbracket P \mid\, !P \rrbracket) \equiv \mathcal{E}(K \mid \mathcal{E}(\llbracket P \rrbracket; \llbracket\, !P \rrbracket)) \xrightarrow{enter\_k} \mathcal{E}(K \mid \mathcal{E}(K \mid \llbracket\, !P \rrbracket))$, and so on; hence, $\llbracket\, !P \rrbracket \xrightarrow{enter\_k}{}^{\omega}$. We now prove that $k \in bn(\mathcal{E}(\cdot))$ implies that there must exists a pair of complementary actions α and $\bar{\alpha}$ such that: (*i*) $\llbracket P \rrbracket \xrightarrow{\alpha}$; (*ii*) $\llbracket Q \rrbracket \xrightarrow{\bar{\alpha}}$; and (*iii*) either α is of kind $\langle - \rangle/(M)/amb\_h/open\_h$ or it is of kind $enter\_h$ for $h \notin bn(\mathcal{E}(\cdot))$. It it was not the case, then $\llbracket P \mid ((\nu b)in\_b.P \mid Q) \rrbracket$, that is structurally equivalent to $\mathcal{E}(\llbracket P \rrbracket \mid \mathcal{E}(\llbracket ((\nu b)in\_b.P \mid Q) \rrbracket))$, would not reduce, in contradiction with Proposition 3.2. Now, points (*i*) – (*iii*) allow us to conclude: if α is of kind $\langle - \rangle/(M)/amb\_h/open\_h$, we fall in a different case of this Theorem and, hence, $\llbracket \cdot \rrbracket$ would violate Proposition 3.1; if it is of kind $enter\_h$, with $h \notin bn(\mathcal{E}(\cdot))$, we conclude that $\llbracket\, !P \rrbracket \xrightarrow{enter\_h}{}^{\omega}$.

- $C_1(\llbracket P \rrbracket) \xrightarrow{open\_k}$ and $C_2(\llbracket Q \rrbracket) \xrightarrow{amb\_k}$: like in the previous case, $C_1(\cdot)$ must be empty. If $\llbracket pull\_m.(\langle n\rangle \mid push\_p \mid p[\surd]) \rrbracket \xrightarrow{open\_k}$ then $\llbracket pull\_m.(\langle n\rangle \mid push\_p \mid p[\surd]) \mid Q \rrbracket \longmapsto$; if $C^{\{n,m\}}_{n[\,]}(\cdot) \xrightarrow{open\_k}$ then $\llbracket n[pull\_n.(\langle m\rangle \mid push\_p \mid p[\surd])] \mid Q \rrbracket \longmapsto$.

- $C_1(\llbracket P \rrbracket) \xrightarrow{amb\_k}$ and $C_2(\llbracket Q \rrbracket) \xrightarrow{\alpha}$, for $\alpha \in \{enter\_k, open\_k\}$: we work like in the previous case, with action amb_k in place of $open\_k$.

[1] To be precise, K is not a process but it is what in [34] is called a *concretion*; however, for our purposes, such a notion is not necessary.

The second claim can be proved in a very similar way, by letting $P \triangleq n[in\_m.(\langle n\rangle \mid p[out\_n.out\_m.\surd])]$. Just notice that action $enter\_k$ must now be replaced by action $catch\_k$ (that in PAC signals the presence of a top-level ambient containing a top-level prefix $pull\_k$) and that $C_1([\![P]\!]) \xrightarrow{catch\_k}$ implies that $[\![n[!\, in\_m.(\langle n\rangle \mid p[out\_n.out\_m.\surd])] \mid !\, Q]\!]$ diverges. $\square$

Notice that the form of objective mobility in PAC is much more liberal than that in MA_o: in the latter, at every moment at most one movement for every ambient can happen, since the moving ambient is blocked by the *mv in*/*mv out* prefix. On the other hand, in PAC the same ambient can undergo different movements, because of execution of different parallel actions naming the same ambient. Not incidentally, MA can encode MA_o, whereas MA cannot encode PAC.

5.2 Adding passwords to SA

In [32], SA has been enriched with passwords: 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 action becomes $\overline{open}\_(m,p)$ and only the ambients knowing the password p can open m. The introduction of passwords was needed in [32] mainly to co-inductively characterize barbed equivalence in a SA-like language; here we prove that passwords enhance the expressive power of the language.

Let SA_p be the language defined by the syntax of SA, with

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

and with the reductions rules of SA extended by also matching passwords.

Theorem 5.3. SA_p *is more expressive than* SA*: there exists an encoding of* SA *in* SA_p*; there exists no encoding of* SA_p *in* SA.

Proof. SA is trivially encodable in SA_p as follows:

$$\begin{array}{rclcrcl} [\![in\_n]\!] & \triangleq & in\_(n,n) & \quad & [\![\overline{in}\_n]\!] & \triangleq & \overline{in}\_(n,n) \\ [\![out\_n]\!] & \triangleq & out\_(n,n) & & [\![\overline{out}\_n]\!] & \triangleq & \overline{out}\_(n,n) \\ [\![open\_n]\!] & \triangleq & open\_(n,n) & & [\![\overline{open}\_n]\!] & \triangleq & \overline{open}\_(n,n) \end{array}$$

The converse is a corollary of Theorem 3.5. $\square$

The language proposed in [32] (called SAP) differs from SA_p in the semantics of the *out* action: in SAP, the co-action is not in the ambient left (like in SA and SA_p) but is in the receiving ambient. Formally, the axiom to exit an ambient now becomes:

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

We now prove that this slight modification makes SAP incomparable with both SA and SA_p; to this aim, it suffices to prove the following two results. Notice that we have introduced SA_p to stress that the two ways of placing the $\overline{out}$ primitive are incomparable.

Theorem 5.4. *There exists no encoding of* SAP *in* SA_p.

Proof. Consider the processes $P_1 \triangleq m[n[out\_(m,p)]]$ and $P_2 \triangleq \overline{out\_}(m,p).\surd$, for n, m and p pairwise distinct; $[\![P_1]\!]$ and $[\![P_2]\!]$ must interact and can do so in four ways:[2]

1. either $C_1([\![P_1]\!]) \xrightarrow{?enter\_h,k}$ and $C_2([\![P_2]\!]) \xrightarrow{enter\_h,k}$,
2. or $C_1([\![P_1]\!]) \xrightarrow{enter\_h,k}$ and $C_2([\![P_2]\!]) \xrightarrow{?enter\_h,k}$,
3. or $C_1([\![P_1]\!]) \xrightarrow{?open\_h,k}$ and $C_2([\![P_2]\!]) \xrightarrow{open\_h,k}$,
4. or $C_1([\![P_1]\!]) \xrightarrow{open\_h,k}$ and $C_2([\![P_2]\!]) \xrightarrow{?open\_h,k}$.

In all cases, by Property 2, we have that $h \in \varphi_{[\![\ ]\!]}(m)$ and $k \in \varphi_{[\![\ ]\!]}(p)$, or vice versa. We now show that all these cases lead to a contradiction; to this aim, notice that, by Property 1, it holds that $[\![P_1]\!] \triangleq C^{\{n,m,p\}}_{m[\ ]}([\![n[out\_(m,p)]]\!])$.

1.,3. Let $\alpha = ?enter\_h,k$ in case 1 and $\alpha = ?open\_h,k$ in case 3. Assume that $C_1(\cdot)$ is not empty; it cannot be that $C_1(\cdot) \xrightarrow{\alpha}$, otherwise $[\![n[m[out\_(n,p)]] \mid P_2]\!] \longmapsto$. Hence $[\![P_1]\!] \xrightarrow{\alpha'}$, for $\alpha' = \overline{in\_}(h,k)$ in case 1 and $\alpha' = \overline{open\_}(h,k)$ in case 3. We now prove that this can be used to violate Proposition 3.1.

   (a) It cannot be that $C^{\{n,m,p\}}_{m[\ ]}(\cdot) \xrightarrow{\alpha'}$, otherwise $[\![m[m[out\_(n,p)]] \mid P_2]\!] \longmapsto$.

   (b) It cannot be that $[\![n[out\_(m,p)]]\!] \xrightarrow{\alpha'}$, otherwise $[\![n[out\_(m,p)] \mid P_2]\!] \longmapsto$.

   Hence, $C_1(\cdot)$ is empty and $[\![P_1]\!] \xrightarrow{\alpha}$; this can happen in three possible ways, all of them contradicting Proposition 3.1. The first two possibilities are formally identical to sub-cases (a) and (b) above (with α in place of α'); now, it is also possible that $C^{\{n,m,p\}}_{m[\ ]}(\cdot) \equiv (\nu\widetilde{n})(h[\,\cdot \mid Q_1] \mid Q_2)$ and $[\![n[out\_(m,p)]]\!] \xrightarrow{\alpha'}$, for $\alpha' = \overline{in\_}(h,k)$ in case 1 and $\alpha' = \overline{open\_}(h,k)$ in case 3. However, recall that $[\![n[out\_(m,p)]]\!] \triangleq C^{\{m,p\}}_{n[\ ]}([\![out\_(m,p)]\!])$; so, it cannot be that $C^{\{m,p\}}_{n[\ ]}(\cdot) \xrightarrow{\alpha'}$, otherwise $[\![n[out\_(p,m)]]\!] \xrightarrow{\alpha'}$, nor that $[\![out\_(m,p)]\!] \xrightarrow{\alpha'}$, otherwise $[\![m[out\_(m,p)]]\!] \xrightarrow{\alpha'}$.

2. This case is similar to the previous one, with $\alpha = enter\_h,k$ and $\alpha' = in\_h,k$.

4. Like before, $C_1(\cdot)$ must be empty. Then, it cannot be that $C^{\{n,m,p\}}_{m[\ ]}(\cdot) \xrightarrow{open\_h,k}$, otherwise $[\![m[n[out\_(p,m)]] \mid P_2]\!] \longmapsto$, nor that $[\![n[out\_(m,p)]]\!] \xrightarrow{open\_h,k}$, otherwise $[\![n[out\_(m,p)] \mid P_2]\!] \longmapsto$. $\square$

Theorem 5.5. *There exists no encoding of* SA *in* SAP.

[2] For SA_p, it suffices to extend Proposition 4.2 in the obvious way, i.e. by letting the label also contain the specified password.

Proof. Consider $P \triangleq m[n[out\_m.\overline{open}\_n] \mid \overline{out}\_m]$ and $Q \triangleq open\_n.\surd$, for $n \neq m$. We know that $[\![P \mid Q]\!] \longmapsto$, and this can happen in six ways:[3]

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, otherwise $[\![m[out\_m.\overline{open}\_n \mid \overline{out}\_m] \mid Q]\!] \longmapsto$, and $[\![P]\!] \xrightarrow{\alpha}$, for $\alpha \in \{open\_h,k\ ,\ ?exit\_h,k\}$. 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 $C^{\{n,m\}}_{m[\,]}(\cdot) \xrightarrow{\alpha}$, otherwise $[\![m[] \mid Q]\!] \longmapsto$, nor that $[\![P']\!] \xrightarrow{\alpha}$, 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 $[\![P]\!] \xrightarrow{\alpha'}$, for $\alpha' \in \{in\_h,k\ ,\ \overline{in}\_h,k\ ,\ \overline{open}\_h,k\ ,\ out\_h,k\}$. Like before, we can prove that there is no way for $[\![P]\!]$ to perform α' without contradicting Proposition 3.1. Hence, it must be that $C_1(\cdot)$ is empty. Again, $[\![P]\!] \triangleq C^{\{n,m\}}_{m[\,]}([\![P']\!])$ and $[\![P]\!] \xrightarrow{\alpha}$, for $\alpha \in \{enter\_h,k\ ,\ ?enter\_h,k\ ,\ ?open\_h,k\ ,\ exit\_h,k\}$. If $C^{\{n,m\}}_{m[\,]}(\cdot) \xrightarrow{\alpha}$ or $[\![P']\!] \xrightarrow{\alpha}$, we can work like for cases 3 and 6 above. So, it must be that $C^{\{n,m\}}_{m[\,]}(\cdot) \equiv (\nu\widetilde{p})(a[\,\cdot \mid R_1] \mid R_2)$ and $[\![P']\!] \xrightarrow{\alpha'}$, for $\alpha' \in \{in\_h,k\ ,\ \overline{in}\_h,k\ ,\ \overline{open}\_h,k\ ,\ out\_h,k\}$. Furthermore, by Property 1, $[\![P']\!] \equiv C^{\{n,m\}}_{\mid}([\![n[out\_m.\overline{open}\_n]]\!]; [\![\overline{out}\_m]\!])$; thus, $[\![P']\!] \xrightarrow{\alpha'}$ can happen in three ways:

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

All these cases lead to contradict Proposition 3.1: in the first two cases, it is easy to prove that also $[\![m[n[\mathbf{0}] \mid \overline{out}\_m] \mid Q]\!] \longmapsto$; in the third case, we would have that $[\![m[out\_m.\overline{open}\_n \mid \overline{out}\_m] \mid Q]\!] \longmapsto$. □

[3] For SAP, Proposition 4.2 must be extended by letting the label also contain the specified password and by adding the pair of complementary actions $exit\_h,k$ and $?exit\_h,k$: the former one signals the presence, within a top-level ambient h, of some ambient that want to exit from h by exhibiting password k; the latter one signals the presence of a top-level $\overline{out}\_(h,k)$ action.

To conclude, notice that SA_p and SAP are more expressive than Dπ: the latter cannot encode SA_p because it cannot encode SA (as a corollary of Proposition 4.7); Dπ cannot encode SAP since, by working like in Proposition 4.7, we can easily prove that SAP cannot be encoded in μKLAIM. On the contrary, Dπ can be encoded both in SA_p and in SAP: this is possible because, thanks to passwords, both SA_p and SAP can atomically match two names, viz. the name of the channel where the Dπ processes communicate and the locality hosting them.

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 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 go). Formally, the encoding acts homomorphically on all the operators, except for

$$
\begin{array}{rcll}
\llbracket\, l : P \,\rrbracket & \triangleq & \llbracket\, P \,\rrbracket_{l'} & \\
\llbracket\, \bar{u}\langle v\rangle.P \,\rrbracket_w & \triangleq & (\nu k)(w[\overline{in_{\_}}(w,u').open_{\_}(\mathsf{p},\mathsf{p}). & \\
 & & \qquad (\langle v'\rangle \mid \mathsf{go}[\overline{open_{\_}}(\mathsf{go},\mathsf{go}).\overline{open_{\_}}(w,k)])\,] & \text{for } k \text{ fresh} \\
 & & \quad \mid open_{\_}(w,k).\llbracket\, P \,\rrbracket_w) & \\
\llbracket\, u(x).P \,\rrbracket_w & \triangleq & \mathsf{p}[in_{\_}(w,u').\overline{open_{\_}}(\mathsf{p},\mathsf{p}).(x').open_{\_}(\mathsf{go},\mathsf{go}).\llbracket\, P \,\rrbracket_w] & \\
\llbracket\, go_{\_}u.P \,\rrbracket_w & \triangleq & (\nu k)(k[\overline{open_{\_}}(k,k)] \mid open_{\_}(k,k).\llbracket\, P \,\rrbracket_{u'}) & \text{for } k \text{ fresh}
\end{array}
$$

where p and go are reserved names and, consequently, l', u', v' and x' are the renamings of l, u, v and x, respectively. It can be proved that this encoding enjoys all the Properties listed in Section 3.

Thus, we have proved the following hierarchy of languages:

$$
\begin{tikzcd}
SA_p & SAP \\
SA \arrow[u] & D\pi \arrow[ul] \arrow[u]
\end{tikzcd}
$$

5.3 Shared vs Localized Channels in BA

Parent-child communications can be modeled in (at least) two ways: the first one exploits *shared channels* (i.e., communications can happen either within the same ambient or via a channel shared by the parent and its child); the second one exploits *localized channels* (i.e., communications can happen either within the same ambient or via a channel owned by either the parent or the child). Both these approaches have been adopted in some presentations of BA; we now formally compare them.

Formally, BA_s is the calculus derived from BA by letting the four reduction rules for remote communications be replaced by:

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

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

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

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

where message M can be consumed by three different input actions placed in different ambients. However, as we now prove, the two forms of communication are incomparable.

Theorem 5.6. BA_s *and* BA *are incomparable: there exists no encoding of* BA_s *in* BA *and there exists no encoding of* BA *in* BA_s *that satisfies Property 6.*

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[4]:

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 $C_j([\![P_j]\!]) \xrightarrow{n'(M)}$, for $\{i, j\} = \{1, 2\}$;

c) $C_i([\![P_i]\!]) \xrightarrow{(M)^{n'}}$ 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), we can prove, like in previous proofs, that both $C_1(\cdot)$ and $C_2(\cdot)$ must be empty. Moreover, it could only be $i = 1$ and $j = 2$, with $C^{\{b\}}_{(x)^{n'}}(\cdot) \xrightarrow{amb\_n'}$, $C^{\{b\}}_{n[\,]}(\cdot) \equiv (\nu\widetilde{h})(h[\,\cdot \mid Q_1] \mid Q_2)$ 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 α (that is $\langle - \rangle^{n'}$ in case (b) and $(M)^{n'}$ in case (c)) cannot be produced: indeed, if $C_1(\cdot) \xrightarrow{\alpha}$, then $C_1(b[\langle n\rangle^\star]) \xrightarrow{\alpha}$; if $C^{\{b\}}_{n[\,]}(\cdot) \xrightarrow{\alpha}$, then $C_1(n[\langle b\rangle^{\hat{}}]) \xrightarrow{\alpha}$; finally, $[\![\langle b\rangle^{\hat{}}]\!] \xrightarrow{\alpha}$ is not possible because of Proposition 3.6.

Hence, it must be that $C_2([\![P_2]\!]) \xrightarrow{\alpha}$, for $\alpha \in \{n'(M), n'\langle - \rangle\}$. Again, the only way to respect Proposition 3.1 is when $C_2(\cdot)$ is empty, $C^{\{b\}}_{n[\,]}(\cdot) \equiv (\nu\widetilde{n})(n'[\,\cdot \mid Q_1] \mid Q_2)$ and $[\![\langle b\rangle^\star]\!] \xrightarrow{\alpha_1}$, for $\alpha_1 \in \{(M)^{\hat{}}, \langle - \rangle^{\hat{}}\}$.

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 $[\![(x)^\star]\!] \xrightarrow{\alpha_2}$, for $\alpha_2 \in \{(M)^{\hat{}}, \langle - \rangle^{\hat{}}\}$. Moreover, α_2 must be of a different kind from α_1: indeed, it they were both inputs (outputs), then we would have that $[\![P_3 \mid n[\langle b\rangle^\star]]\!] \longmapsto$.

Now, consider processes $P_5 \triangleq (x)^\star.\surd$ and $P_6 \triangleq n[\langle b\rangle^{\hat{}}]$. The possible interactions between their encodings are $C_1([\![P_5]\!]) \xrightarrow{\alpha}$ and $[\![P_6]\!] \xrightarrow{\bar{\alpha}}$, for $\alpha \in \{amb\_m, \langle - \rangle^m, (M)^m\}$ and, correspondingly, $\bar{\alpha} \in \{enter\_m, m(M), m\langle - \rangle\}$. Indeed, $C_2(\cdot)$ must be empty and $[\![P_6]\!]$ cannot perform α. Moreover, it must be that $C^{\{b\}}_{n[\,]}(\cdot) \equiv (\nu\widetilde{k})(k[\,\cdot \mid R_1] \mid R_2)$ and $[\![\langle b\rangle^{\hat{}}]\!] \xrightarrow{\alpha_3}$, for $\alpha_3 \in \{in\_m, (M)^{\hat{}}, \langle - \rangle^{\hat{}}\}$ respectively. However, this allows us to conclude that $[\![\cdot]\!]$ is not an encoding that respects Property 6: if $\alpha_3 = in\_m$, we can conclude that $[\![P_5 \mid\, !\,P_6]\!]$ diverges, by a reasoning similar to the one in the proof of Theorem 5.2; if $\alpha_3 \in \{(M)^{\hat{}}, \langle - \rangle^{\hat{}}\}$, we have that either $[\![P_1 \mid P_6]\!] \longmapsto$ or $[\![P_2 \mid P_6]\!] \longmapsto$, according to whether α_3 is of the same kind as α_1 or of α_2.

[4] For BA_s, Proposition 4.3 must be updated as follows: (*i*) ignore points 4 and 5; (*ii*) let labels $n(M)$ and $n\langle - \rangle$ mean that there is a top-level ambient n with a top-level action $(x)^{\hat{}}$ or $\langle M\rangle^{\hat{}}$.

For the non-encodability of BA_s in BA, we work in a similar way. First, consider the BA_s processes $P_1 \triangleq (x)^n.\surd$ and $P_2 \triangleq n[\langle b\rangle^\wedge]$. Like in the non-encodability of BA in BA_s, we have that $[\![\langle b\rangle^\wedge]\!] \xrightarrow{\alpha_1}$, for $\alpha_1 \in \{(M)^\star, \langle -\rangle^\star\}$. Second, consider $P_3 \triangleq \langle b\rangle^n.\surd$ and $P_2 \triangleq n[(x)^\wedge]$; again, we have that $[\![(x)^\wedge]\!] \xrightarrow{\alpha_2}$, for $\alpha_2 \in \{(M)^\star, \langle -\rangle^\star\}$. We are now ready to violate Proposition 3.1 (so, in this case Property 6 is not needed): if α_1 and α_2 are of the same kind, then $[\![P_1 \mid P_4]\!] \longmapsto$; otherwise, $[\![(x)^\wedge \mid \langle b\rangle^\wedge]\!] \longmapsto$. $\square$

5.4 Alternative Mobility Primitives in BA: SBA and NBA

Safe Boxed Ambient (SBA) A first extension of BA is SBA (*Safe* BA, [33]): it is BA extended with co-actions to better control ambient movements, in the same spirit as SA. However, SBA co-actions can either allow any ambient enter/exit a given ambient n (and this is similar to SA), or can selectively allow movements (this resembles SAP, though no password appears in SBA). Formally, the reductions for ambient movements are:

$$n[in\_m.P_1 \mid P_2] \mid m[\overline{in}\_\delta.P_3 \mid P_4] \;\longmapsto\; m[n[P_1 \mid P_2] \mid P_3 \mid P_4]$$

$$m[n[out\_m.P_1 \mid P_2] \mid P_3] \mid \overline{out}\_\delta.P_4 \;\longmapsto\; n[P_1 \mid P_2] \mid m[P_3] \mid P_4$$

for $\delta \in \{*, n\}$. Also notice that the $\overline{out}$ action is placed outside the ambient left, like in SAP.

Remark 5.1. *For* SBA*, Proposition 4.3 must be adapted as follows:*

(*i*) *case 3 now involves labels $n : enter\_m$ and $m : ?enter\_n$, where the first label means that there is a (possibly restricted) top-level ambient n containing a top-level prefix in_m and the second label means that there is a top-level ambient m containing a top-level prefix $\overline{in}\_*$ or $\overline{in}\_n$;*

(*ii*) *introduce case 8, that holds if $P_1 \xrightarrow{n:exit\_m}$ and $P_2 \xrightarrow{?n:exit\_m}$, where the first label means that there is a top-level ambient m containing a (possibly restricted) top-level ambient n containing a top-level prefix out_m and the second label means that there is a top-level prefix $\overline{out}\_*$ or $\overline{out}\_n$;*

(*iii*) *leave all the remaining cases exactly as in Proposition 4.3.*

It is quite easy to prove that of SBA is more expressive than BA.

Theorem 5.7. SBA *is more expressive than* BA*: there is an encoding of* BA *in* SBA*, whereas there is no encoding of* SBA *in* BA*.*

Proof. It is easy to prove that SBA_* can encode BA: it suffices to translate every operator homomorphically, except for $[\![u[P]]\!] \triangleq !\,\overline{out}\_* \mid u[!\,\overline{in}\_* \mid [\![P]\!]]$ and $[\![\mathbf{0}]\!] \triangleq !\,\overline{out}\_*$. The converse is a corollary of Theorem 3.5. $\square$

New Boxed Ambient (NBA) [6] 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. As we have shown in Theorem 5.6, located channels cannot be encoded in shared channels nor vice versa; thus, to compare NBA with BA and SBA, we consider the variant of NBA with localised channels that we call NBA_l. Formally, its distinctive reduction rules are:

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

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

Remark 5.2. *For* NBA_l*, Proposition 4.3 must be adapted as follows:*

(i) *case 3 now involves labels* $n : enter\_m, p$ *and* $m :?enter\_n, p$*, where the first label means that there is a (possibly restricted) top-level ambient n containing a top-level prefix* $in\_(m, p)$ *and the second label means that there is a top-level ambient m containing a top-level prefix* $\overline{in\_}(x, p)$ *and n has been used to replace x in the process prefixed by the action;*

(ii) *introduce case 8, that holds if* $P_1 \xrightarrow{n:exit\_m,p}$ *and* $P_2 \xrightarrow{?n:exit\_m,p}$*, where the first label means that there is a top-level ambient m containing a (possibly restricted) top-level ambient n containing a top-level prefix* $out\_(m, p)$ *and the second label means that there is a top-level prefix* $\overline{out\_}(x, p)$ *and n has been used to replace x in the process prefixed by the action;*

(iii) *leave all the remaining cases exactly as in Proposition 4.3.*

We now prove that NBA_l is more expressive than BA.

Theorem 5.8. NBA_l *is more expressive than* BA*: there is an encoding of* BA *in* NBA_l*, whereas there is no encoding of* NBA_l *in* BA.

Proof. NBA_l can encode BA: it suffices to translate every operator homomorphically, except for

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

$$[\![\, in\_u.P \,]\!] \triangleq in\_(u,p).[\![\, P \,]\!] \qquad [\![\, out\_u.P \,]\!] \triangleq out\_(u,p).[\![\, P \,]\!]$$

for some predefined and fixed (constant) password p. The converse is a corollary of Theorem 3.5. □

The hierarchy of BA-derived languages We have shown that both SBA and NBA_l are more expressive than BA; it remains to understand the relationships between NBA_l and SBA. We now prove that the two languages are incomparable.

Theorem 5.9. *There is no encoding of* NBA_l *in* SBA.

Proof. Consider the processes $P \triangleq n[in\_(m,p).\langle q\rangle^\star]$ and $Q \triangleq m[\overline{in\_}(x,p).\langle\ \rangle^*] \mid (\ )^m.\surd$, for n, m, p and q pairwise distinct. Their encodings must interact: $C_1([\![\, P \,]\!]) \xrightarrow{\mu}$ and $C_2([\![\, Q \,]\!]) \xrightarrow{\mu'}$, for some μ and μ'. By Property 2, it must be that $fn(\mu) = fn(\mu') = \{m', p'\}$, for $m' \in \varphi_{[\![\ ]\!]}(m)$ and $p' \in \varphi_{[\![\ ]\!]}(p)$; hence, since $m' \neq p'$, it must be that $\mu = h : enter\_k$ and $\mu' = k :?enter\_h$ (or vice versa, that is handled similarly), for $\{h, k\} = \{m', p'\}$; alternatively, we could have $\mu = h : exit\_k$ and $\mu' = k :?exit\_h$ (or vice versa), but the reasoning would be similar.

First, notice that $C_1(\cdot)$ must be empty, otherwise either $C_1([\![\, n[in\_(p,m).\langle q\rangle^\star] \,]\!]) \xrightarrow{\mu}$ or $C_1([\![\, in\_(p,m).\langle q\rangle^\star.\langle n\rangle^\star \,]\!]) \xrightarrow{\mu}$, according to whether $C^{\{m,p,q\}}_{n[\,]}(\cdot) \xrightarrow{in\_k}$ or $[\![\, in\_(m,p).\langle q\rangle^\star \,]\!] \xrightarrow{in\_k}$. Hence, $[\![\, P \,]\!] \xrightarrow{\mu}$; this can happen in three ways:

- $C^{\{m,p,q\}}_{n[\,]}(\cdot) \xrightarrow{\mu}$, but then $[\![\, n[in\_(p,m).\langle q\rangle^\star] \mid Q \,]\!] \longmapsto$;
- $[\![\, in\_(m,p).\langle q\rangle^\star \,]\!] \xrightarrow{\mu}$, but then $[\![\, in\_(m,p).\langle q\rangle^\star \mid Q \,]\!] \longmapsto$;

- $C^{\{m,p,q\}}_{n[\,]}(\cdot) \equiv (\nu\widetilde{n})(h[\,\cdot \mid Q_1] \mid Q_2)$ and $[\![\, in\_(m,p).\langle q\rangle^{\star} \,]\!] \xrightarrow{in\_k}$. In this case, let σ be the permutation that swaps m with q, if $k = p'$, and that swaps p with q, otherwise. Then, $[\![\, P\sigma \,]\!] \xrightarrow{\mu}$ and so $[\![\, P\sigma \mid Q \,]\!] \longmapsto$, in contradiction with Proposition 3.1. $\square$

Theorem 5.10. *There is no encoding of* SBA *in* NBA_l.

Proof. Consider the processes $P \triangleq n[in\_m]$ and $Q \triangleq m[\overline{in}\_n.\langle\ \rangle^*] \mid (\ )^m.\surd$, for $n \neq m$. Their encodings must interact: $C_1([\![\, P \,]\!]) \xrightarrow{\mu}$ and $C_2([\![\, Q \,]\!]) \xrightarrow{\mu'}$, for some μ and μ'. By Property 2, it must be that $fn(\mu) = fn(\mu') = \{m', n'\}$, for $m' \in \varphi_{[\![\ ]\!]}(m)$ and $n' \in \varphi_{[\![\ ]\!]}(n)$; hence, it must be that $\mu = h : enter\_(k,p)$ and $\mu' = k : ?enter\_(h,p)$ (or vice versa, that is handled similarly), for $\{k,p\} = \{m',n'\}$; alternatively, we could have $\mu = h : exit\_(k,p)$ and $\mu' = k : ?exit\_(h,p)$ (or vice versa), but the reasoning would be similar.

First, $C_1(\cdot)$ must be empty, otherwise $C_1([\![\, n[\langle m\rangle^{\star}] \,]\!]) \xrightarrow{\mu}$; indeed, because of Proposition 3.6, it cannot be that $[\![\, in\_m \,]\!] \xrightarrow{in\_(k,p)}$, since $\{k,p\} \cap \varphi_{[\![\ ]\!]}(n) \neq \emptyset$ but $n \notin fn(in\_m)$. Hence, $[\![\, n[in\_m] \,]\!] \xrightarrow{\mu}$; this can happen in three ways:

- $C^{\{m\}}_{n[\,]}(\cdot) \xrightarrow{\mu}$, but then $[\![\, n[\langle m\rangle^{\star}] \mid Q \,]\!] \longmapsto$;
- $[\![\, in\_m \,]\!] \xrightarrow{\mu}$, but then $[\![\, in\_m \mid Q \,]\!] \longmapsto$;
- $C^{\{m\}}_{n[\,]}(\cdot) \equiv (\nu\widetilde{n})(h[\,\cdot \mid Q_1] \mid Q_2)$ and $[\![\, in\_m \,]\!] \xrightarrow{in\_(k,p)}$: again, because of Proposition 3.6, the latter fact is not possible. $\square$

To sum up, we have the following hierarchy for BA-derived calculi:

$$\begin{tikzcd} \mathrm{NBA}_l & & \mathrm{SBA} \\ & \mathrm{BA} \arrow[ul] \arrow[ur] & \end{tikzcd}$$

6 Conclusions and Related Work

We have comparatively studied several mainstream calculi for mobility and some of their variants, namely the asynchronous π-calculus, a distributed π-calculus, a distributed version of Linda, Mobile Ambients (and two dialects with objective moves), 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 expressive power. To this aim, we have exploited the criteria presented and discussed in [21], but we believe that they should also hold under different 'reasonable' encodability criteria.

In our opinion, the most important of our positive results is the encodability of π_a-calculus in MA: indeed, to the best of our knowledge, no such encoding has even been presented before ours (in particular, none of the encodings of π_a-calculus in MA satisfied operational soundness). It has to be said that our encoding is quite complex (the encoding of a single communication in π_a-calculus requires 14 reduction steps in MA) because some ingenuity is needed to handle the possible interferences that can arise between the encoding of different actions on the same channel. Notice that the encoding of π_a-calculus in SA [30] is simpler (just 5 reductions to mimic a single communication), since co-actions can be exploited to reduce such interferences; this is a further evidence of SA's expressive power. Moreover, an equally good encoding (though sensibly more complex) holds also in SA without the communication primitives [49]; we believe that such a result is not possible in MA. Finally, we also want to remark that the encoding of π_a-calculus in BA [5] is even simpler: thanks to parent-child communications, just 2 reductions are needed to mimic a single communication. These remarks can be used to argue that co-actions and, even more, remote communications are more suitable to implement channel-based communications in ambient-like languages. Of course, to make this claim formal, we should prove that no more efficient encoding of π_a-calculus in MA is possible; we leave this aspect for future work.

It is surprising that some expected separation results were so difficult to prove. A paradigmatic sample of this fact is Conjecture 1: we have not been able to prove such (expectable) result. Indeed, remote communications should not be enough to reasonably implement the *open* primitive of MA.

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 expectable 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). Moreover, objective moves, that in [11] are defined 'dangerous' (because they can be used to entrap an ambient in a restricted ambient and leave it there for ever), turned out to be less expressive than the subjectives moves of MA. 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 (see, e.g., the various encodings of π_a-calculus in MA presented in [11, 10, 9]).

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, MA_o reduces the expressiveness of MA, whereas SA and NBA/SBA enhance the expressiveness of MA and BA, 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 PAC vs MA, 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, [49] 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; our encoding of π_a-calculus in MA confirms this intuition, since communications in π_a-calculus are translated by exploiting communications in MA. Second, [29] 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 [8] entail that Dπ cannot be encoded in π_a-calculus, under properties similar to ours; notably, they need homomorphism w.r.t. parallel composition whereas we just rely on compositionality. Fourth, [42, 43] are inspired by Palamidessi's work on electoral systems [40] 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-calculus 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 ([7, 31, 40, 20, 15], just to cite some samples). These works are quite orthogonal to ours, since they compare different sub-calculi of the same language, whereas we aimed at comparing of different programming paradigms.

Acknowledgments. Thanks to Iain Phillips that introduced me to [42, 43]; thanks also to Rosario Pugliese, Ivano Salvo and Maria Grazia Vigliotti that read a preliminary version of this work.

A Properties of the encoding of π_a-calculus in MA

All the properties of Section 3 are easy to prove, except for Properties 3 and 4. To carry out the proofs, we found it useful to assign a number to the actions of the encoding, to refer them easily later on. Moreover, for the sake of simplicity, we assume triadic communications in MA, so that a datum $\langle b_1, b_2, b_3\rangle$ can be consumed by an action (x_1, x_2, x_3) in just one reduction step.

$$[\![\, \bar{a}\langle b\rangle \,]\!] \;\triangleq\; a_1[a_2[\overset{\textcircled{3}}{open\_a_3}.\overset{\textcircled{4'}}{\langle b_1, b_2, b_3\rangle}]]$$

$$[\![\, a(x).P \,]\!] \;\triangleq\; \overset{\textcircled{1}}{open\_a_1}.(\nu p, q)(\overset{\textcircled{9}}{open\_p} \mid a_3[\overset{\textcircled{2}}{in\_a_2}.\overset{\textcircled{7}}{open\_\mathtt{rest}} \mid \overset{\textcircled{4''}}{(x_1, x_2, x_3)}.\overset{\textcircled{5}}{in\_q}.p[\overset{\textcircled{8}}{out\_q}.[\![\, P \,]\!]]]$$
$$\mid q[\overset{\textcircled{6}}{open\_a_2}.\mathtt{rest}[!\,\mathtt{rest}[\overset{\textcircled{10}}{in\_a_3}.\overset{\textcircled{11}}{out\_q}.in\_a_2.open\_\mathtt{rest}]]])$$

In what follows, we denote with $\textcircled{4}$ the simultaneous execution of actions $\textcircled{4'}$ and $\textcircled{4''}$. Actions $\textcircled{1}/\ldots/\textcircled{9}$ are used to mimic a communication in the source term; moreover, note that action $\textcircled{7}$ is not needed, if no interference arises. However, in the presence of interferences between the encoding of different communications along the same channel, action $\textcircled{7}$ becomes fundamental. In such a case, some actions (viz, $\textcircled{7}$, $\textcircled{10}$ and $\textcircled{11}$) are needed to restore the interfering a_3 ambients at top-level, ready to complete their task. However, the corresponding computations are *spurious*, in the sense that they do not correspond to original reductions in π_a-calculus and are only performed to remedy some interference.

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

- it is of kind $\textcircled{2}$, but leads an a_3 ambient within an a_2 ambient that has already been entered by (at least) another a_3 ambient;
- it is of kind $\textcircled{7}$ and is executed within an a_3 ambient;
- it is of kind $\textcircled{10}$ or $\textcircled{11}$.

In the first two cases above, we denote the step with $\textcircled{2s}$ and $\textcircled{7s}$, to emphasize its spurious nature and distinguish it from a step performed to mimic a reduction in π_a-calculus.

To ease reading, let us denote with $\mathrm{PR}_a^{\textcircled{k}, s_2, s_{10}, s_7}$ the process arising from $[\![\, \bar{a}\langle b\rangle \mid a(x).P \,]\!]$, for some b, x and P, after the execution of the non-spurious action $\textcircled{k}$ and that contains: s_2 ambients named a_3

that have executed only a ⓶ₛ action; s_{10} ambients named a_3 that have also executed a ⑩ action; s_7 ambients named a_3 that have also executed a ⑦ₛ action. Moreover, we denote with PR_a the encoding of $\bar{a}\langle b\rangle$, for some b, with its enclosing a_1 ambient dissolved. Finally, $\mathrm{PR}_a^{s_2,s_{10},s_7}$ denotes the process

$$(\nu q)(q[\ !\mathtt{rest}[in\_a_3.out\_q.in\_a_2.open\_\mathtt{rest}] \mid \prod_{i=1}^{s_2} a_3[open\_\mathtt{rest} \mid (x_1,x_2,x_3).\cdots]) \\ \mid \prod_{i=1}^{s_{10}} a_3[open\_\mathtt{rest} \mid (x_1,x_2,x_3).\cdots \mid \mathtt{rest}[out\_q.\cdots]]) \\ \mid \prod_{i=1}^{s_7} a_3[(x_1,x_2,x_3).\cdots \mid out\_q.\cdots]\ )$$

We now give a simple proposition that describes some syntactic and operational properties of the processes we have just defined; the proof directly follows from the definition of the processes.

Proposition A.1.

1. $\mathrm{PR}_a^{①,s_2,s_{10},s_7}$ *is such that* $s_2 = s_{10} = s_7 = 0$*; moreover, it is structurally equivalent to a process of the form* $\mathrm{PR}_a \mid a_3[in\_a_2.open\_\mathtt{rest} \mid (x_1,x_2,x_3).\cdots]$*; finally, it can evolve by either performing a* ② *and becoming* $\mathrm{PR}_a^{②,0,0,0}$*, or performing a* ⓶ₛ *and becoming* PR_a*, with its* a_3 *ambient that enters in a sibling* a_2 *ambient that has already been entered by (at least) another* a_3*.*

2. $\mathrm{PR}_a^{ⓚ,s_2,s_{10},s_7}$*, for* $k \in \{2,3,4\}$*, is such that* $s_{10} = s_7 = 0$*; moreover, it can evolve by either performing a* ⓚ₊₁ *and becoming* $\mathrm{PR}_a^{ⓚ₊₁,s_2,0,0}$*, or undergoing to a* ⓶ₛ *and becoming* $\mathrm{PR}_a^{ⓚ,s_2+1,0,0}$*.*

3. $\mathrm{PR}_a^{ⓚ,s_2,s_{10},s_7}$*, for* $k \in \{5,6\}$*, is such that* $s_{10} = s_7 = 0$*; moreover, it can only evolve by performing a* ⓚ₊₁ *and becoming* $\mathrm{PR}_a^{ⓚ₊₁,s_2,0,0}$*.*

4. $\mathrm{PR}_a^{ⓚ,s_2,s_{10},s_7}$*, for* $k \in \{7,8\}$ *can evolve by either performing a* ⓚ₊₁ *and becoming* $\mathrm{PR}_a^{ⓚ₊₁,s_2,s_{10},s_7}$*, or performing a* ⑩ *and becoming* $\mathrm{PR}_a^{ⓚ,s_2-1,s_{10}+1,s_7}$ *(provided that* $s_2 > 0$*), or performing a* ⑦ₛ *and becoming* $\mathrm{PR}_a^{ⓚ,s_2,s_{10}-1,s_7+1}$ *(provided that* $s_{10} > 0$*), or performing a* ⑪ *and becoming* $\mathrm{PR}_a^{ⓚ,s_2,s_{10},s_7-1} \mid a_3[in\_a_2.open\_\mathtt{rest} \mid (x_1,x_2,x_3.\cdots)]$ *(provided that* $s_7 > 0$*).*

5. $\mathrm{PR}_a^{⑨,s_2,s_{10},s_7}$ *is structurally equivalent to a process of the form* $\mathrm{PR}_a^{s_2,s_{10},s_7} \mid [\![P\{b/x\}]\!]$*, for some b, x and P; moreover, it can evolve by either performing a* ⑩ *and becoming* $\mathrm{PR}_a^{⑨,s_2-1,s_{10}+1,s_7}$ *(provided that* $s_2 > 0$*), or performing a* ⑦ₛ *and becoming* $\mathrm{PR}_a^{⑨,s_2,s_{10}-1,s_7+1}$ *(provided that* $s_{10} > 0$*), or performing a* ⑪ *and becoming* $\mathrm{PR}_a^{⑨,s_2,s_{10},s_7-1} \mid a_3[in\_a_2.open\_\mathtt{rest} \mid (x_1,x_2,x_3.\cdots)]$ *(provided that* $s_7 > 0$*).*

6. $\mathrm{PR}_a^{0,0,0} \simeq \mathbf{0}$*, where '*$\simeq$*' denotes strong barbed equivalence.*

Operational completeness (i.e. the first item of Property 3) is now a trivial corollary of the previous proposition. To prove operational soundness (i.e. the second item of Property 3), it suffices to prove the following lemma. There and in what follows, we denote with $n^a_{ⓚ}$, for $k \in \{1,\ldots,11,2s,7s\}$, the number of actions of kind ⓚ originated from the encoding of a communication along a in a given sequence of n reductions; $n_{ⓚ}$ stands for $\sum_{a\in\mathcal{N}} n^a_{ⓚ}$.

Lemma A.2. *Let P be a π_a-calculus process and Q be a* MA *process such that* $[\![P]\!] \longmapsto^n Q$*, for* $n = \sum_{k=1}^{11} n_{ⓚ} + n_{ⓢ} + n_{ⓢ}$*. Then,*

$$Q \equiv (\nu \widetilde{m}_1, \widetilde{m}_2, \widetilde{m}_3)\Big([\![R]\!] \mid \prod_{a \in \mathcal{N}} \Big(\prod_{k=1}^{n^a_{②ⓢ}+n^a_{⑩}+n^a_{⑦ⓢ}} \mathrm{PR}_a \mid \prod_{k=1}^{n^a_{①}} \mathrm{PR}_a^{①,0,0,0} \mid \prod_{k=2}^{8} \prod_{i=1}^{n^a_{ⓚ}} \mathrm{PR}_a^{ⓚ, n^a_{2s_{k_i}}, n^a_{10_{k_i}}, n^a_{7s_{k_i}}} \mid \prod_{i=1}^{n^a_{⑨}} \mathrm{PR}_a^{n^a_{2s_{9_i}}, n^a_{10_{9_i}}, n^a_{7s_{9_i}}} \Big) \Big)$$

for some $\widetilde{m}_1, \widetilde{m}_2, \widetilde{m}_3$ of the same length, for some π_a-calculus process R, for $n^a_{10_{k_i}} = n^a_{7s_{k_i}} = 0$ ($k < 7$) and for $n_{②ⓢ} = \sum_{a \in \mathcal{N}} \sum_{k=2}^{9} \sum_{i=1}^{n^a_{ⓚ}} n^a_{2s_{k_i}}$, $n_{⑩} = \sum_{a \in \mathcal{N}} \sum_{k=7}^{9} \sum_{i=1}^{n^a_{ⓚ}} n^a_{10_{k_i}}$, $n_{⑦ⓢ} = \sum_{a \in \mathcal{N}} \sum_{k=7}^{9} \sum_{i=1}^{n^a_{ⓚ}} n^a_{7s_{k_i}}$.

Proof. By induction on n. The base step is trivial; the inductive step relies on Proposition A.1. □

Theorem A.3 (Operational soundness)**.** *Let P be a π_a-calculus process and Q be a* MA *process such that* $[\![P]\!] \longmapsto^n Q$*. Then,* $P \longmapsto^{n_{①}} P'$*, for some π_a-calculus process P' such that* $Q \Longmapsto\simeq [\![P']\!]$.

Proof. By induction on n. The base step is trivial; the inductive step relies on Lemma A.2. □

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

Lemma A.4. *Let* $[\![P]\!] \longmapsto^n$*; then the number of spurious actions (i.e.,* $n_{②ⓢ} + n_{⑩} + n_{⑦ⓢ} + n_{⑪}$*) is at most* $2 \cdot (n_{①})^2 + n_{①}$.

Proof. The worst case is when all the $n_{①}$ actions are on the same channel, say a, and can be obtained as follows. Put all the $n_{①}$ a_3 ambients in the same a_2 ambient; this introduces $n_{①} - 1$ spurious actions of kind ②ⓢ and the corresponding $3 \cdot (n_{①} - 1)$ actions (of kind ⑩, ⑦ⓢ and ⑪) to remedy this choice. Then, put all the remaining $n_{①} - 1$ a_3 ambients in the same a_2 ambient; this introduces $4 \cdot (n_{①} - 2)$ spurious actions. And so on. Thus, the overall number of spurious actions is at most

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

□

Theorem A.5 (Divergence freedom)**.** *If* $[\![P]\!] \longmapsto^{\omega}$*, then* $P \longmapsto^{\omega}$.

Proof. Let $[\![P]\!] \longmapsto^n$ and observe that $n > 0$ implies that $n_{①} > 0$. Moreover, for every $k \in \{2, \ldots, 9\}$, it holds that $n_{ⓚ} \le n_{①}$; thus, by Lemma A.4, $n \to \infty$ implies that $n_{①} \to \infty$. So, by Theorem A.3, we easily conclude. □

References

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

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

[3] L. Bettini, R. De Nicola, and R. Pugliese. KLAVA: a Java Package for Distributed and Mobile Applications. *Software – Practice and Experience*, 32:1365–1394, 2002.

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

[5] 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.

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

[7] 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.

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

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

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

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

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

[13] R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: a Kernel Language for Agents Interaction and Mobility. *IEEE Trans. on Software Engineering*, 24(5):315–330, 1998.

[14] R. De Nicola, G. Ferrari, R. Pugliese, and B. Veneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, 2000.

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

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

[17] 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.

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

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

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

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

[22] 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.

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

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

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

[26] K. Honda and N. Yoshida. On Reduction-Based Process Semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.

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

[28] N. Kobayashi. A Partially Deadlock-Free Typed Process Calculus. *ACM Trans. on Progr. Lang. and Systems*, 20(2):436–482, 1998. An extended abstract appeared in *LICS'97*.

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

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

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

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

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

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

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

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

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

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

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

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

[41] 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.

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

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

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

[45] 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.

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

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

[48] E. Shapiro. Separating concurrent languages with categories of language embeddings. In *Proc. of STOC*, pages 198-208. ACM, 1991.

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