Parametric model checking for Mobile Ambients

Dino Distefano

Department of Computer Science
Queen Mary University of London
ddino@dcs.qmul.ac.uk

Abstract. In this paper we propose an new abstract finite model of Mobile Ambients able to express some interesting security properties. This model can be used for analysing these properties by means of model checking techniques. The precision of the analysis can be increased by modifying certain parameters of the model increasingly avoiding thereby the occurrences of false counterexamples.

1 Introduction

The calculus of Mobile Ambients (MA) is meant to model *wide area* computations. Introduced in [2], MA has as main characteristic to allow active processes to move between different sites.

A wide range of work has been recently carried out on the analysis of mobile ambients [1, 8, 12, 13, 18], mostly based on static-analysis techniques and abstract interpretation [7].

In this paper we propose a *parametric finite abstract* model to analyse properties of mobile ambients processes by model checking — as an alternative to static analysis and abstract interpretation. Our model is based on techniques introduced first in [10]. Such techniques provide a general framework for modelling and verifying systems whose computation involves manipulation of pointer structures. The model we define here is suitable for verifying some security properties of systems, such as *secrecy*. It has the following features: *(i)* It provides a *safe approximation* of the concrete transition system of processes. *(ii)* It models *finitely* (by means of abstraction) processes that are in principle infinite due to replication (e.i., $!P$). *(iii)* The model depends on two (numeric) *parameters* that can be increased to tune its precision in case false counterexamples are returned by the model checking algorithm.

The analysis we propose is based on the following strategy. Our models, called HABA, are special Büchi automata with some characteristic proper of history-dependent automata [17]. HABA are used to represent the behaviour of an ambient process P. Security properties are expressed in the temporal logic *NTL* (introduced in [10]) which is interpreted over HABA runs. Then, the model checking algorithm defined in [9, 11] can be used to verify the validity of the property against the model.

The contribution of our approach w.r.t. existing analysis lays on its ability to deal finitely with replication. The model distinguishes between P and $!P$ at several level of precisions (due to parametricity). Existing techniques can cope with replication only to a limited extent. They are designed only for abstraction $\{0, 1, \omega\}$ (i.e., none, one, many). Our abstraction goes beyond this range. The other advantage of our approach is that the model introduced here provide a general and completely automated framework for the verification of properties of MA. This means that the model is *not* limited to some specific safety properties (like static analysis techniques). Many temporal properties expressible by NTL-formulae can be automatically checked on the abstract model giving us the possibility to infer safe answer on ambient processes.

Related work. Our model takes inspiration from the following works. The paper [18] proposes an algorithm detecting process firewalls that are not protective. The technique is based on a control flow analysis and does not distinguish between a process P and $!P$. This technique is enhanced in [12] where the precision of the analysis is improved by the use of information about the multiplicity of the number of ambients occurring within another ambient. The distinction is within the range $\{0, 1, \omega\}$. Another refinement of the analysis proposed in [18], for the special case

of Safe Ambients [15], is introduced in [8]. However, the analysis proposed — as the one in [18] — *does not* distinguish between different copies of the same ambient. An abstract interpretation framework for MA is proposed in [13]. Based on [12] and [8] the analysis given in this paper considers some information about multiplicity of the ambients and contextual information. Again, based on [12], the paper [1] defines a more accurate analysis for capturing boundary crossing. Also in this work no information on multiplicities is provided.

A parallel stream of work considers model checking for mobile ambients using spatial logics [5] and in particular ambient logic[3]. In [6] the authors identify a fragment of mobile ambients (where replication is replaced by recursion) verifiable by model-checking. For this fragment, a model-checking algorithm for the ambient logic is proposed. The paper [4] introduces a spatial logic for synchronous π-calculus and investigate its power. A model-checking algorithm is then presented for a class of bounded processes. Our contribution stands somehow between these two independent streams of work in that it applies model checking in a static analysis oriented fashion.

Organisation of the paper. This paper is organised as follows: Section 2 reviews some background on the ambient calculus. Section 3 gives an overview of *NTL* and HABA. Section 4 defines an operational semantics for MA using HABA. Section 5 provides some concluding remarks.

2 An Overview of Mobile Ambients

We consider the pure *Mobile Ambients* calculus [2] without communication primitives.

Definition 2.1. *Let $\mathcal{N}$ be a denumerable set of* names *(ranged over by a, b, n, m). The set of processes over $\mathcal{N}$ is defined according to the following grammar:*

$$N ::= \mathsf{in}\ n \mid \mathsf{out}\ n \mid \mathsf{open}\ n \qquad \textit{(capabilities)}$$

$$P, Q ::= \mathbf{0} \mid (\nu n)P \mid P|Q \mid\ !P \mid n[P] \mid N.P \qquad \textit{(processes)}$$

For a process P we write $n(P)$, $fn(P)$, $bn(P)$ for the set of names, free names and bound names, respectively. In a process there can be multiple ambients with the same name. $\mathbf{0}$ does not perform any action. The restriction $(\nu n)P$ creates a new name called n that is private in the scope of P. $P \mid Q$ is the standard parallel composition of processes P and Q. Replication $!P$ represents an arbitrary number of copies of P and it is used to introduce recursion as well as iteration. $n[P]$ represents an ambient with name n enclosing a running process P. Ambients can be arbitrarily nested. Capabilities provide ambients with the possibility to *interact* with other ambients. In particular, $\mathsf{in}\ n$ has the effect to move the ambient that performs itinto a sibling ambient called n (if there exists one). Symmetrically, by $\mathsf{out}\ n$ an ambient nested inside n, moves outside $\mathsf{open}\ n$ dissolves an ambient n nested inside the one performing this capability.

Operational semantics The standard semantics of Mobile Ambients is given in [2] on the basis of a structural congruence between processes (denoted by $\equiv$) and a reduction relation. Structural congruence provides us with a partitioning of processes into equivalence classes (of structurally congruent processes) within which processes are equivalent up to syntactic restructuring. Table 1 defines the structural congruence. Moreover, processes are identified up to α-conversion, i.e., $(\nu n)P = (\nu m)P\{n/m\}$ if $m \notin fn(P)$. Note that: $n[P]|n[Q] \not\equiv n[P|Q]$ that is, multiple copies of an ambient n have distinct identities. Moreover,

$$!(\nu n)P \not\equiv (\nu n)!P \tag{1}$$

that is, the replication operator combined with restriction creates an infinite number of new names. This is exemplified as follows.

Example 1. Let $P_1 = !(\nu n)(n[\mathsf{in}\ n])$ and $P_2 = (\nu n)!(n[\mathsf{in}\ n])$. Process P_1 creates an unbounded number of new local names, i.e., it expands as:

$$n'[\mathsf{in}\ n']|n''[\mathsf{in}\ n'']|n'''[\mathsf{in}\ n''']|\cdots$$

where $n', n'', n''', \ldots$ cannot interact with each other since these names are local. Hence, P_1 cannot perform any action. On the contrary, P_2 expands as

$$n[\mathsf{in}\ n]|n[\mathsf{in}\ n]|n[\mathsf{in}\ n]|\cdots$$

that is, an infinite number of copies of the same ambient n is created. Every copy can interact with any other one. Hence, in P_2 the instances of n can move, producing thus, every kind of possible nesting. □

Our analysis adopts some simplifications on processes, in particular on those of the form $!(\nu n)P$.

$P \equiv P$	(Refl)	$P\|Q \equiv Q\|P$	(Par Comm)
$P \equiv Q \Rightarrow Q \equiv P$	(Symm)	$(P\|Q)\|R \equiv P\|(Q\|R)$	(Par Assoc)
$P \equiv Q \wedge Q \equiv R \Rightarrow P \equiv R$	(Trans)	$!P \equiv P\|!P$	(Repl Par)
		$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(Res Res)
$P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$	(Res)	$(\nu n)P\|Q \equiv P\|(\nu n)Q$ if $n \notin fn(P)$	(Res Par)
$P \equiv Q \Rightarrow P\|R \equiv Q\|R$	(Par)	$(\nu n)m[P] \equiv m[(\nu n)P]$ if $n \neq m$	(Res Amb)
$P \equiv Q \Rightarrow !P \equiv !Q$	(Repl)		
$P \equiv Q \Rightarrow n[P] \equiv n[Q]$	(Amb)	$P\|\mathbf{0} \equiv P$	(Zero Par)
$P \equiv Q \Rightarrow M.P \equiv M.Q$	(Amb)	$(\nu n)\mathbf{0} \equiv \mathbf{0}$	(Zero Res)
		$!\mathbf{0} \equiv \mathbf{0}$	(Zero Repl)

Table 1. Structural Congruence for Mobile ambients.

The reduction relation $\rightarrow$ is defined by the rules listed in Table 2. The first three rules define the effect of capabilities in one step-reductions. The reduction is then propagated within name restriction, ambient nesting, and parallel composition by the next three rules. The last rule allows the use of structural congruence during reduction. As usual, $\rightarrow^*$ stands for the reflexive and transitive closure of $\rightarrow$.

$$n[\mathsf{in}\ m.P|Q]|m[R] \rightarrow m[n[P|Q]|R] \quad \text{(Red In)}$$

$$m[n[\mathsf{out}\ m.P|Q]|R] \rightarrow n[P|Q]|m[R] \quad \text{(Red Out)}$$

$$\mathsf{open}\ n.P|n[Q] \rightarrow P|Q \quad \text{(Red Open)} \qquad \frac{P' \equiv P \quad P \rightarrow Q \quad Q \equiv Q'}{P' \rightarrow Q'} \quad \text{(Red } \equiv\text{)}$$

$$\frac{P \rightarrow Q}{n[P] \rightarrow n[Q]} \quad \text{(Red Amb)} \qquad \frac{P \rightarrow Q}{P|R \rightarrow Q|R} \quad \text{(Red Par)} \qquad \frac{P \rightarrow Q}{(\nu n)P \rightarrow (\nu n)Q} \quad \text{(Red Res)}$$

Table 2. Reduction rules for Mobile ambients.

3 An overview on *NTL* and HABA

In this section we summarise the framework for modelling and model cheking systems with pointers introduced in [10, 11].

Navigation Temporal Logic. Let LVAR be a countable set of logical variables ranged over by x, y, z, and *Ent* be a countable set of entities ranged over by e, e', e_1 etc. $\bot \notin \mathit{Ent}$ is used to represent "undefined"; we denote $E^\bot = E \cup \{\bot\}$ for arbitrary $E \subseteq \mathit{Ent}$. Navigation Temporal Logic (*NTL*) is a linear temporal logic where quantification ranges over logical variables that can denote entities, or may be undefined. The syntax of navigation expressions is defined by the grammar:

$$\alpha \;::=\; \mathit{nil} \;\Big|\; x \;\Big|\; \alpha\!\uparrow$$

where *nil* denotes the null reference, x denotes the entity (or nil) that is the value of x, and $\alpha\!\uparrow$ denotes the entity referred to by (the entity denoted by) α (if any). Let $x\!\uparrow^0 = x$ and $x\!\uparrow^{n+1} = (x\!\uparrow^n)\!\uparrow$ for natural n. The syntax of *NTL* is:

$$\Phi \;::=\; \alpha = \alpha \;\Big|\; \alpha \;\mathsf{new} \;\Big|\; \alpha \rightsquigarrow \alpha \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \exists x.\, \Phi \;\Big|\; \mathsf{X}\Phi \;\Big|\; \Phi \,\mathsf{U}\, \Phi \;\;.$$

The basic proposition α new states that the entity (referred to by) α is fresh, $\alpha = \beta$ states that α and β are aliases, and $\alpha \rightsquigarrow \beta$ expresses that (the entity denoted by) β is reachable from (the entity denoted by) α. The boolean connectives, quantification, and the linear temporal connectives X (next) and U (until) have the usual temporal interpretation. We denote $\alpha \neq \beta$ for $\neg(\alpha = \beta)$, $\alpha \not\rightsquigarrow \beta$ for $\neg(\alpha \rightsquigarrow \beta)$ and $\forall x.\, \Phi$ for $\neg(\exists x.\, \neg\Phi)$. The other boolean connectives and temporal operators $\Diamond$ (eventually) and $\Box$ (always) are standard [19]. For example, $\Diamond(\exists x.\, x \neq v \;\wedge\; x \rightsquigarrow v \;\wedge\; v \rightsquigarrow x)$ expresses that eventually v will point to a non-empty cycle.

Formulae are interpreted over infinite sequences of triples, called *allocation sequence*,

$$(E_0, \mu_0, \mathcal{C}_0)(E_1, \mu_1, \mathcal{C}_1)(E_2, \mu_2, \mathcal{C}_2)\ldots$$

where for all $i \geqslant 0$, $E_i \subseteq \mathit{Ent}$ and $\mu_i : E_i^\bot \to E_i^\bot$ such that $\mu_i(\bot) = \bot$; μ_i encodes the pointer structure of E_i[1]. $\mathcal{C}_i$ is a function on E_i such that $\mathcal{C}_i(e) \in \mathbb{M} = \{1, \ldots, M\} \cup \{*\}$ for some fixed constant $M > 0$. The number $\mathcal{C}_i(e)$ is called the *cardinality* of e. Entity e for which $\mathcal{C}_i(e) = m \leqslant M$ represents a chain of m "concrete" entities; if $\mathcal{C}_i(e) = *$, e represents a chain that is longer than M. In the latter case, the entity is called *unbounded*. (Such entities are similar to summary nodes [20], with the specific property that they always abstract from chains.)

Configurations and morphisms. States in our automata are triples $(E, \mu, \mathcal{C})$, called *configurations*. In the following *Conf* denote the set of all configurations ranged over by γ and γ'. Configurations at different abstraction levels are related by morphisms, defined as follows. Let $\mathcal{C}(\{e_1, \ldots, e_n\}) = \mathcal{C}(e_1) \oplus \ldots \oplus \mathcal{C}(e_n)$ where $n \oplus m = n{+}m$ if $n{+}m \leqslant M$ and $*$ otherwise.

Definition 3.1. *For $\gamma, \gamma' \in \mathit{Conf}$, surjective function $h : E \to E'$ is a* morphism *if:*

1. *for all $e \in E'$, $h^{-1}(e)$ is a pure chain and $\mathcal{C}'(e) = \mathcal{C}(h^{-1}(e))$*
2. *$e \prec' e' \;\Rightarrow\; \mathit{last}(h^{-1}(e)) \prec \mathit{first}(h^{-1}(e'))$*
3. *$e \prec e' \;\Rightarrow\; h(e) \preceq' h(e')$ where $\preceq'$ denotes the reflexive closure of $\prec'$.*

According to the first condition only pure chains may be abstracted to a single entity while keeping the cardinalities invariant. The last two conditions enforce the preservation of the pointer structure under h. By means of a morphism the abstract shape of the pointer dependencies represented by the two related configurations is maintained. The identity function *id* is a morphism (from a configuration to itself) and morphisms are closed under composition, i.e., the set of configurations equipped with morphisms forms a category. Configurations γ and γ' are isomorphic, denoted $\gamma \cong \gamma'$, iff there exist morphisms from γ to γ' and from γ' to γ such that their composition is the identity function.

[1] In the following we often use the notatiion $e \prec e'$ for $\mu(e) = e'$.

Automata-based models. Morphisms relate configurations that model the pointer structure at distinct abstraction levels. They do not model the dynamic evolution of such structures. To reflect the execution of pointer-manipulating statements, such as either the creation or deletion of entities or the change of pointers *reallocations* are used.

Definition 3.2. *For $\gamma, \gamma' \in Conf$, $\lambda : (E^{\perp} \times E'^{\perp}) \to \mathbb{M}$ is a* reallocation *if:*

1. *(a)* $\mathcal{C}(e) = \bigoplus \lambda(e, e')$ *and (b)* $\mathcal{C}'(e') = \bigoplus \lambda(e, e')$
2. *(a) for all* $e \in E$, $|\{e' \mid \lambda(e, e') = *\}| \leqslant 1$ *and (b)* $\{e' \mid \lambda(\perp, e') = *\} = \varnothing$
3. *(a) for all* $e \in E$, $\{e' \mid \lambda(e, e') \neq 0\}$ *and*
   (b) for all $e' \in E'$, $\{e \mid \lambda(e, e') \neq 0\}$ *are chains.*

We write $\gamma \overset{\lambda}{\rightsquigarrow} \gamma'$ *if there is a reallocation (named* λ*) from* γ *to* γ'*.*

The special entity $\perp$ is used to model birth and death: $\lambda(\perp, e) \neq 0$ denotes the birth of (some instances of) e whereas $\lambda(e, \perp) \neq 0$ denotes the death of (some instances of) e. Intuitively speaking, reallocation λ redistributes cardinalities on E to E' such that (1a) the total cardinality allocated by λ to $e \in E$ equals $\mathcal{C}(e)$ and (1b) the total cardinality assigned to $e' \in E'$ equals $\mathcal{C}'(e')$. Moreover, (2a) for each entity e unbounded cardinalities (i.e., equal to $*$) are assigned only once (according to (1b) to an unbounded entity in E'), and (2b) no unbounded entities can be born. The last condition is self-explanatory. Note that the identity function *id* is a reallocation. The concept of reallocation can be considered as a generalization of the idea of identity change as, for instance, present in history-dependent automata [17]: besides the possible change of identity of entities, it allows for the evolution of pointer structures.

To model the dynamic evolution of a system manipulating (abstract) linked lists, we use a generalization of Büchi automata where each state is a configuration and transitions exist between states only if these states can be related by means of a reallocation reflecting the possible change in the pointer structure.

Definition 3.3. *A* high-level allocation Büchi automaton *(HABA)* $\mathcal{H}$ *is a tuple* $\langle X, C, \to, I, \mathcal{F} \rangle$ *with:*

- $X \subseteq$ LVar*, a finite set of logical variables;*
- $C \subseteq Conf$*, a set of configurations (also called* states*);*
- $\to \subseteq C \times (Ent \times Ent \times \mathbb{M}) \times C$*, a* transition relation*, s.t.* $c \to_{\lambda} c' \Rightarrow c \overset{\lambda}{\rightsquigarrow} c'$*;*
- $I : C \rightharpoonup 2^{Ent} \times (X \rightharpoonup Ent)$*, an* initialization *function such that for all* c *with* $I(c) = (N, \theta)$ *we have* $N \subseteq E$ *and* $\theta : X \rightharpoonup E$*.*
- $\mathcal{F} \subseteq 2^{C}$ *a set of sets of* accept *states.*

HABA can be used to model the behaviour of systems at different levels of abstraction: In particular, when all entities in any state are concrete (i.e., $\mathcal{C}(e) = 1$ for all e), a concrete model is obtained that is very close to the actual system behaviour.

Model Checking NTL In [11] a model checking algorithm which establishes whether a formula ϕ is satisfiable on a given (finite) HABA $\mathcal{H}$ was developed. The model checking algorithm is based on the construction of a tableau graph for $G_{\mathcal{H}}(\phi)$ as in [16]. As usual in model checking of infinite-state systems in the presence of abstraction the algorithm is sound but not complete in the sense that it might return *false negatives*, This means that if the algorithm fails to show $\mathcal{H} \models \Phi$ then it cannot be concluded that Φ is *not* satisfiable (by some run of $\mathcal{H}$). However, since such a failure is always accompanied by a "prospective" counterexample of Φ, further analysis or testing may be used to come to a more precise conclusion.

4 Abstract models for mobile ambients

We give an abstract semantics for MA that captures essential information useful for proving security properties. The information we try to retrieve are along the line of [8, 12, 13, 18]. We start with some motivating examples.

Motivating examples In wide area systems security is an important issue since trustworthy ambients may operate inside and together with untrustworthy ambients. Malicious ambients can acquire information contained in other ambients by opening them. Properties such as *secrecy* of data are preserved if no untrustworthy ambient can ever open a trustworthy one.

Example 2. In [8] the following system is considered. Ambient m wants to send a message to ambient b. Messages are delivered by enclosing them in a wrapper ambient that moves inside the receiver which acquires the information by opening it. For secret messages we want to be sure that they can be opened only by the receiver b.

$$SYS_1 = m[mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]]] \mid b[\mathsf{open}\ msg] \mid \mathsf{open}\ msg.$$

Figure 1 shows a pictorial view of the initial configuration of SYS_1. Data D is secret, *mail* is the pilot ambient that goes out of m to reach b. The outer-most ambient, which we denote by @, attempts to access the secret by an $\mathsf{open}\ msg$. Once inside b, the wrapper *mail* is opened and b reads the secret D. For the process SYS_1 we want to guarantee that the property (UA): *"no untrusted ambients can access D".*

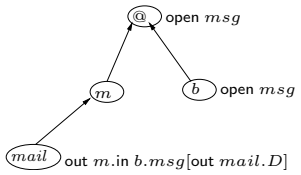

Fig. 1. Initial configuration of SYS_1.

Example 3. The following example was presented in [1] where multilevel security for Mobile Ambients is investigated. *Boundary* ambients are introduced to protect high-level information. The restriction is that high-level data must be contained either in boundary ambients or in low level ambients not escaping boundaries. The authors consider the following system:

$$SYS_2 = m[send[\mathsf{out}\ m.\mathsf{in}\ b \mid hdata[\mathsf{in}\ filter]]] \mid b[\mathsf{open}\ send] \mid filter[\mathsf{in}\ send] \mid \mathsf{open}\ filter.$$

Boundary ambients are b and *send*. The security property we want this system to satisfy is (BA)" *"hdata is always within boundary ambients or if hdata is within the low level ambient filter then filter is in a boundary ambient".*

The aim of this paper is to develop finite-state models for mobile ambients in order to verify properties such as (UA) and (BA) by model checking.

4.1 HABA modelling approach

Basic idea of the model. Along the line of [8, 12, 13, 18], the essential information we want to retrieve from a mobile ambient process P is:

which ambients may end up in which other ambient

To model the structure of a process P we introduce a classification among the entities in use. For any ambient a occurring in P we have:

- A special entity a^{ho} (called a's *host*) that is used to record, at any point in the computation, the ambients (hosted) directly inside *any* copy of ambient a. It is fixed, i.e., during the computation its position within the topology of the process does not change.

- A special entity a^{is} (called the *inactive site* of a). It is the repository where the copies of a are placed when this ambient is inactive. Informally speaking, inactive means that a cannot execute any action and it is not yet visible to other ambients (a complete exposition of this concept is postponed till Section 4.5). As a^{ho}, also a^{is} is fixed and it does not move during the computation.
- Concrete/multiple/unbounded entities —distinct from a^{ho} and a^{is}— that are used to represent instances of the ambient a which execute capabilities. Each of this entities (if concrete) can move according to the capabilities of the particular copy of a it represents. If there exists several instances of a, some of them may be represented by multiple/unbounded entities. In this case, before one single instance in the multiple/unbounded entity performs a capability it is materialised.

Example 4. Figure 3 depicts how process SYS_1 in Figure 1 is represented in our model. Outgoing references define the son/father relation μ. Notation $e{:}n$ says that e denotes an ambient with name n. The host of an ambient, say a, keeps track of the ambients directly contained in *any* copy of a. Thus, ambients m and b are inside the outer-most ambient @, whereas *mail* is inside m. Ambients b and *mail* are empty. Hosts entities are depicted as squares and inactive sites as patterned squares in order to distinguish them from entities that can move around during the computation (depicted as circles). *msg* is *inactive* since in the beginning it cannot execute any action because it is guarded by $\mathsf{out}\ m.\mathsf{in}\ b$. Only when both $\mathsf{out}\ m$ and $\mathsf{in}\ b$ have been consumed, *msg* is enabled to execute its actions (it becomes *active*). Inactive ambients are modelled by letting the copies of the ambient lead to the their inactive site. Only if an ambient is inactive its inactive site has some entity leading to it. Figure 8 (left) shows the use of the unbounded entity e_2 to model more than M copies of the ambient n inside @.

4.2 Process indexing and preliminary notation

We assume that the names occurring bound inside restriction are all distinct from each other and from the free names. This can always be achieved by α-conversion.

In order to achieve a more precise analysis we label the ambients occurring in the process P by distinguished indexes. Let J be a countable set of indexes and $\tilde{\mathcal{N}} = \{n_j \mid n \in \mathcal{N}, j \in J\}$ be the set of indexed names. A *indexed process* $\tilde{P}$ is a process such that for every ambient construct $n[Q]$ occuring in $\tilde{P}$ we have $n \in \tilde{\mathcal{N}}$. However, names in capabilities or in name restriction are not indexed. Let $noi(P) : \mathbf{Proc} \to \mathbf{Proc}$ be function that given an indexed process $\tilde{P}$, returns the (non-indexed) process P obtained from $\tilde{P}$ by stripping out every index from the ambient names occurring in $\tilde{P}$. The process $\tilde{P}$ is an indexed version of P, if $noi(\tilde{P}) = P$. Moreover, $idx(\tilde{P}) = \{j \in J \mid n_j \text{ occurs in } \tilde{P}\}$ is the set of indexes occurring in $\tilde{P}$.

Definition 4.1. *Process $\tilde{P}$ is* well-indexed *if and only if*

- $P = \mathbf{0}$;
- $P = n_i[Q]$ *and* Q *is well-indexed and* $i \notin idx(Q)$;
- $P = Q \mid Q'$, *and* Q, Q' *are well-indexed and* $idx(Q) \cap idx(Q') = \varnothing$;
- $P = !Q$ *and* Q *is well-indexed;*
- $P = M.Q$ *and* Q *is well-indexed;*
- $P = (\nu n)Q$ *and* Q *is well-indexed.*

Example 5. A well-indexed instance of $P = a[\mathsf{in}\ b.a[\mathsf{out}\ b]] \mid b[\mathbf{0}]$ is $\tilde{P} = a_1[\mathsf{in}\ b.a_2[\mathsf{out}\ b]] \mid b_3[\mathbf{0}]$. We use indexing in order to distinguish between copies of identically named ambients that have different behaviour, as for example the two instances of a in $\tilde{P}$ above. For simplicity (without loss of generality) we assume that process indexing is done always after α-conversion.

In the following we assume that every process has been well-indexed. This can be considered as a preprocessing step before the extraction of the model.

Preliminary notation We assume the existence of a global function $\mathsf{A} : \mathit{Ent} \to n(\tilde{P})$ that associates to every entity e a (indexed) name of the ambients in well-indexed process $\tilde{P}$ represented by e. The function A provides a partitioning of Ent. For $e \in \mathit{Ent}$, we write $e{:}n$ as a shorthand for $\mathsf{A}(e) = n$.[2] Moreover, we write $e{:}a \in E$ is a shorthand for $e \in E \wedge \mathsf{A}(e) = a$. We consider two special sets of entities $E_P^{\mathsf{is}} \cap E_P^{\mathsf{ho}} = \varnothing$, where:

- $E_P^{\mathsf{is}} = \{n^{\mathsf{is}} \in \mathit{Ent} \mid n \in n(P)\}$ is the set of inactive sites.
- $E_P^{\mathsf{ho}} = \{n^{\mathsf{ho}} \mid n^{\mathsf{is}} \in n(P)\}$, is the set of host entities.

In every state of the model and for every ambient n we have n^{is} point to n^{ho}; moreover we assume $\mathsf{A}(n^{\mathsf{is}}) = \mathsf{A}(n^{\mathsf{ho}}) = n$. Note that for an indexed process $\tilde{P}$, there exist distinct instances of n_i^{is}, n_j^{is} and n_j^{ho}, n_j^{ho} where $i \neq j$ for different occurrences of the ambient n in P. Every HABA state modelling mobile ambients is of the form:

$$q = \langle \gamma, \mathsf{P} \rangle \in \textsc{States}$$

where $\textsc{States} = \textsc{Conf} \times (\mathit{Ent} \rightharpoonup 2^{\mathbf{Proc}})$. The first component $\gamma = (E, \mu, \mathcal{C}) \in \textsc{Conf}$ is the standard configuration of HABA states as defined in Definition 3.3. For γ we write E^{fix} for its set of fixed entities $E^{\mathsf{fix}} = E \cap (E_P^{\mathsf{ho}} \cup E_P^{\mathsf{is}})$ and E^{c} for its set of entities representing the copies of ambients that can move, i.e., $E^{\mathsf{c}} = E \backslash E^{\mathsf{fix}}$. The second component $\mathsf{P} : \mathit{Ent} \rightharpoonup 2^{\mathbf{Proc}}$ is introduced for the special case of mobile ambients. $\mathsf{P}(e)$ associates to the entity e the set of processes that e must execute. The typical state we obtain is shown in figures where the component $\mathsf{P}(e)$ is depicted close to e. It is not written if it is the empty process. For example in Figure 3, e_2 has to execute $\{\mathsf{open}\ \mathit{msg}\}$, whereas e_1 only $\{\mathbf{0}\}$.

4.3 Pre-initial and initial state: an overview

Pre-initial state. The *pre-initial state* is an artificial state that we add to the model to identify for every entity which ambient it represents. The pre-initial state of a process P is built in such a way that every entity representing a copy of the ambient n leads to the inactive site $n^{\mathsf{is}} \in E_P^{\mathsf{is}}$. The structure of the graph does not reflect the initial topology described by P. When we specify formulae in the logic we will exploit the fact that an entity e in the pre-initial state leads to n^{is} to express the fact that e is an entity representing a copy of the ambient n.

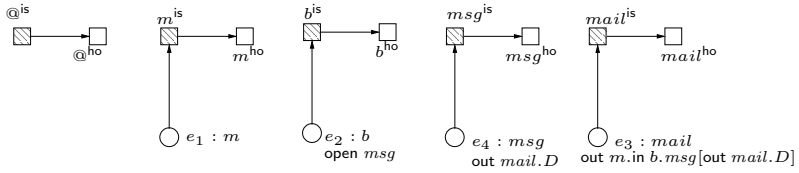

Fig. 2. Pre-initial state of the process SYS_1 of Example 2.

Example 6. Figure 2 depicts the pre-initial state of the process SYS_1 in Example 2. The important point to note for the moment in this figure is that every copy of an ambient leads to the corresponding entity in E_P^{is}. For example e_1 that stands for m leads to m^{is}. Although e_1 has the label m (i.e., $\mathsf{A}(e_1) = m$), this information cannot be exploited in the logic. However, in *NTL* we can refer to m^{is} by a logical variable x_m. Hence, due to the pre-initial state, we can use the formula $\exists x : x \rightsquigarrow x_m$ in order to identify x as a copy representing the ambient m. □

[2] In the following we often make use of this notation in some contexts if it is necessary to know the ambient of an entity. For example, in a function with parameter e, we write $f(e{:}a)$ if it is essential to know that e represents a copy of the ambient a.

Example 7. The security property (UA) of Example 2 is violated if and only if the following *NTL* formula is satisfied

$$\Phi_{\mathsf{UA}} \equiv \exists x : x \rightsquigarrow x_{msg} \wedge \Diamond(x \not\rightsquigarrow x_{msg} \ \wedge \ x\uparrow \neq x_{mail}\uparrow \ \wedge \ \ x\uparrow \neq x_b\uparrow).$$

Φ_{UA} states that *msg* eventually will be included inside an ambient different from *mail* and b (which are the only trustworthy ones) violating therefore the security property (UA). The property (BA) of Example 3 is violated if and only if:

$$\begin{aligned}\Psi_{\mathsf{BA}} \equiv \exists x : x \rightsquigarrow x_{hdata} \wedge \exists y : y \rightsquigarrow x_{filter} \wedge \Diamond(x \not\rightsquigarrow x_{hdata}\ x\uparrow \neq x_{send}\uparrow \ \wedge \\ x\uparrow \neq x_b\uparrow \ \wedge \ (x\uparrow = \mathit{filter} \ \Rightarrow y \not\rightsquigarrow x_{filter} \ \wedge \ y\uparrow \neq x_b\uparrow \ \wedge \ y\uparrow \neq x_{send}\uparrow))\end{aligned}$$

Ψ_{BA} states that eventually *hdata* escapes the boundary ambients b and *send*, and if it is inside *filter* this is not protected by one of the two boundaries.

Hence, if $\mathcal{H}_{SYS_1}$ and $\mathcal{H}_{SYS_2}$ are the HABA modelling SYS_1 and SYS_2, the security properties are guaranteed to hold if we verify $\mathcal{H}_{SYS_1} \not\models \Phi_{\mathsf{UA}}$ and $\mathcal{H}_{SYS_2} \not\models \Psi_{BA}$. That can be automatically checked using the model checking algorithm defined in [11, 9].

Initial state. The *initial-state* models the son/father relation (i.e. the topology) described by the process in terms of entities and references between them. For example, Figure 3 shows the initial state of the process SYS_1 of Example 2. Finally, note that the ambient @ does not have a real instance (it is modelled only by $@^{\mathsf{is}}$ and $@^{\mathsf{ho}}$), therefore we use $@^{\mathsf{is}}$ for the execution of open and !.

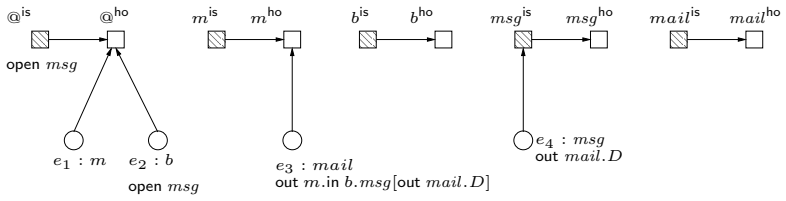

Fig. 3. Initial state of the process SYS_1 of Example 2.

4.4 On morphisms and canonical form for mobile ambients

Assumption on morphisms. The employment of E_P^{is} as variables to name ambients in *NTL*-formulae entails some assumptions on morphisms and reallocations. In particular, throughout this paper we consider only morphisms that satisfy the following conditions:

$$h : \gamma_1 \to \gamma_2 \ \Rightarrow \ h \restriction E_P^{\mathsf{is}} = id_{E_P^{\mathsf{is}}} \tag{2}$$

$$\gamma_1 \overset{\lambda}{\hookrightarrow} \gamma_2 \ \Rightarrow \ \forall e \in E_P^{\mathsf{is}} : \lambda(e, e) = 1 \tag{3}$$

$$h : \gamma_q \to \gamma_{q'} \ \Rightarrow \ \forall e \in E_{\gamma_q} : (\mathsf{P}_q(e) = \mathsf{P}_{q'}(h(e)) \ \wedge \ \mathsf{A}(e) = \mathsf{A}(h(e))) \tag{4}$$

They force the correspondence of the program variables in configurations related by morphisms or reallocations. Condition (4) is typical for the ambient calculus and simply forces morphisms to map an entity e only onto another entity e' representing the same ambient and executing the same process.

Definition 4.2 (L-safety, L-compactness, L-canonicity). *Let $L > 0$ and γ a configuration with $E_P^{\mathsf{ho}} \subseteq E_\gamma$.*

- *γ is* L-safe *if* $\forall e \in E_P^{\mathsf{ho}} : (\forall e' : d(e', e) \leqslant L \Rightarrow \mathcal{C}_\gamma(e') = 1)$.

– γ *is* L-compact *if* $\forall e \in E_\gamma : (\mathit{indegree}(e) > 1 \ \vee \ \exists e' \in E_P^{\mathsf{ho}} : d(e, e') \leqslant L + 1)$.
– γ *is* L-canonical *(or in* L*-normal form) if* γ *is* L*-safe and* γ *is* L*-compact.*

Unformally, a state is L-safe if only concrete entities are closer than $L + 1$ pointer dereferences a from a host. Moreover, a state is L-compact if there are no pure chains longer than $L + 1$. We use only states in canonical form. This has several advantages (see [11]): it is possible to determine precisely which entities are involved in a pointer update; moreover, for every configuration γ the canonical form exists and it is unique (denoted by $\mathsf{cf}(\gamma)$). Finally, using only states in canonical form ensures to obtain always a finite-state HABA.

Dealing with multiple instances of an ambient. We represent multiple copies of the same ambient by multiple/unbounded entities. Due to canonical form, multiple/unbounded entities are not direct children of hosts: there are L concrete entities in between. However, both the multiple/unbounded entity and the preceding concrete entities represent different copies of the *same* ambient. In other words, all these entities are assumed to be at the same level, i.e., inside the same ambient. This is according to our initial aim to collect information about what is contained at top level for every ambient whereas we abstract the inner levels. With this model we are able to distinguish that inside an ambient, say a, there are no instances of the ambient b; or there are precisely i instances of b with $1 \leq i \leq L + M$; or there are more than $L + M$ instances of b. Since L and M are parameters of the model they can be properly tuned to accomplish a more precise model.

Example 8. Assuming $M = 1$, in the ambient a depicted in Figure 4, there are *exactly* two instances of the ambient n and *any* number of copies strictly greater than 4 of the ambient b. Between copies of the same ambient, we depicted dashed horizontal arrows to stress that, at the conceptual level, these arrows do not describe a son/father relation as the solid vertical ones.

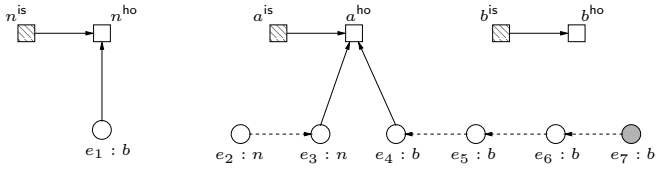

Fig. 4. Representing multiple instances of the same ambient.

4.5 Coding processes into HABA configurations

In this section we define a function that codes a given process P into a HABA state. This function returns: *(i)* a configuration γ that models the ambients topology delineated by P; and *(ii)* a function P that associates to every entity in the configuration the set of capabilities it must perform. We first need a few auxiliary definitions.

Configuration union. For a configuration γ, let $\ell^a_\gamma(e)$ be the longest pure chain of a copies in γ leading to (and including) e. That is

$$\ell^a_\gamma(e) = \{e' \in E^{\mathsf{c}}_\gamma \mid e' {:} a, \ e \in \mu^*(e')\} \cup \{e\} \tag{5}$$

For example in Figure 4, we have $\ell^n_\gamma(a^{\mathsf{ho}}) = \{e_2, e_3, a^{\mathsf{ho}}\}$ and $\ell^b_\gamma(a^{\mathsf{ho}}) = \{e_7, e_6, e_5, e_4, a^{\mathsf{ho}}\}$ and $\ell^n_\gamma(b^{\mathsf{ho}}) = \{e_1, b^{\mathsf{ho}}\}$. For configurations γ, γ' with distinct sets of non-fixed entities i.e., such that

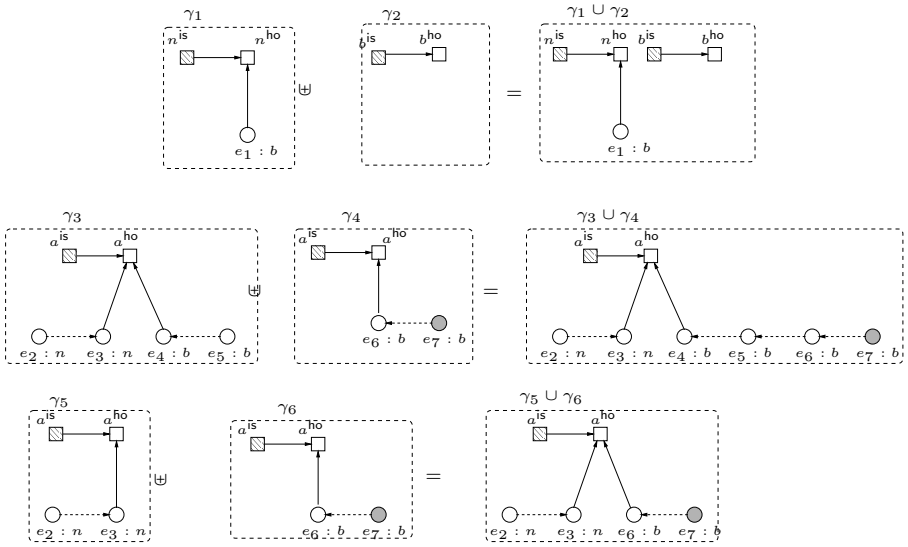

Fig. 5. Example of configuration unions.

$E^{\mathsf{c}}_{\gamma} \cap E^{\mathsf{c}}_{\gamma'} = \varnothing$ we define the union configuration $\gamma \uplus \gamma' = (E, \mu, \mathcal{C})$ where:

$$E = E_{\gamma} \cup E_{\gamma'}$$

$$\mu(e) = \begin{cases} \mu_{\gamma}(e) & \text{if } e \in E_{\gamma} \\ \mu_{\gamma'}(e) & \text{if } e \in E^{\mathsf{fix}}_{\gamma'} \\ \mathit{first}(\ell^{a}_{\gamma}(\mu_{\gamma'}(e))) & \text{if } e{:}a \in E^{\mathsf{c}}_{\gamma'} \wedge \mu_{\gamma'}(e) \in E^{\mathsf{ho}}_{\gamma'} \\ \mu_{\gamma'}(e) & \text{otherwise} \end{cases}$$

$$\mathcal{C}(e) = \begin{cases} \mathcal{C}_{\gamma}(e) & \text{if } e \in E_{\gamma} \\ \mathcal{C}_{\gamma'}(e) & \text{if } e \in E_{\gamma'} \backslash E_{\gamma} \end{cases}$$

The definition of E is straightforward. $\mathcal{C}$ is well defined since $\mathcal{C}_{\gamma}$ and $\mathcal{C}_{\gamma'}$ are equal to the $\mathbf{1}$[3] on fixed entities which may be the only intersection of their domains. For $e{:}a \in E_{\gamma'}$, $\mu(e)$ assigns the first entity in the queue of copies of a. Some examples of configuration unions are depicted in Figure 5. If both configurations have copies of an ambient, say b, inside the same ambient, say a, the union appends the copies of the second configuration to those of the first one. In the figure, this is exemplified in $\gamma_3 \uplus \gamma_4$. To accomplish union of states we need also to define a notion of union for the component P of the state. This is done point-wise as follows: let $q = \langle \gamma, \mathsf{P} \rangle$ and $q' = \langle \gamma', \mathsf{P}' \rangle$ be two states and $e \in E_{\gamma} \cup E_{\gamma'}$, then

$$(\mathsf{P} \uplus \mathsf{P}')(e) = \begin{cases} \mathsf{P}(e) \cup \mathsf{P}'(e) & \text{if } e \in E_{\gamma} \cap E_{\gamma'} \\ \mathsf{P}(e) & \text{if } e \in E_{\gamma} \backslash E_{\gamma'} \\ \mathsf{P}'(e) & \text{if } e \in E_{\gamma'} \backslash E_{\gamma} \end{cases} \tag{6}$$

Finally the union of states is defined component-wise: $\langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}' \rangle = \langle \gamma \uplus \gamma', \mathsf{P} \uplus \mathsf{P}' \rangle$.

Subprocesses executed by ambients. Given a process P the function $\rho : \mathbf{Proc} \to 2^{\mathbf{Proc}}$ returns the set of sub-processes that the ambient containing P must execute. It is defined by:

$$\begin{array}{ll} \rho(\mathbf{0}) = \varnothing & \rho(M.Q) = \{M.Q\} \\ \rho(Q \mid Q') = \rho(Q) \cup \rho(Q') & \rho(m[Q]) = \varnothing \\ \rho(!Q) = \{!Q\} & \rho((\nu n)Q) = \rho(Q) \end{array}$$

Processes belonging to nested ambients are not returned. Note that because of the assumption on the bound names we can delete restriction.

[3] This is the cardinality function that assigns 1 to every entity in its domain.

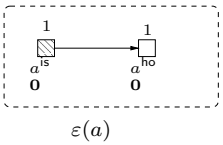

Fig. 6. Graphical representation of $\varepsilon(a)$.

Example 9. Consider our running example SYS_1. We have:

$$\rho(\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]) = \{\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D])\}.$$

This will be used to define the process that must be executed by the entity representing the instance of *mail*. Moreover, since m contains only $mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]])$ the copy of m does not execute anything, in fact $\rho(mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]]) = \varnothing$.

Enabled ambients. An *enabled* ambient is an ambient which is ready to perform some action. $enab(P)$ denotes the set of all enabled ambients in P. It is defined as:

$$\begin{array}{ll} enab(\mathbf{0}) = \varnothing & enab(M.Q) = \varnothing \\ enab(Q \mid Q') = enab(Q) \cup enab(Q') & enab(a[Q]) = \{a\} \cup enab(Q) \\ enab(!Q) = enab(Q) & enab((\nu n)Q) = enab(Q). \end{array}$$

For example, in SYS_1 of Example 2 the set $enab(SYS_1) = \{m, mail, b\}$. Note that ambient *msg* is not enabled (yet). The classification of enabled and non-enabled ambients entails a corresponding representation on the HABA configurations of the process. We have the following (semantic) notion that corresponds to (the syntactic one of) being enabled.

Definition 4.3. *In state q, the ambient n* is active *if $\nexists e \in E_{\gamma_q} : e \prec_{\gamma_q} n^{\mathsf{is}}$.*

If n is not active it is called *inactive*. The operational rules use this notion in order to distinguish between ambients that may perform a capability from those that cannot.

Constructing the state. We use the following abbreviation for a configuration composed only by two entities.

$$(e_1, k_1, P_1) \prec (e_2, k_2, P_2) = \langle \{e_1, e_2\}, \{e_1 \prec e_2\}, \{e_1 \mapsto k_1,\ e_2 \mapsto k_2\}, \{e_1 \mapsto P_1,\ e_2 \mapsto P_2\} \rangle$$

The very basic building block for the construction of the state is the representation of that (sub)part related to the fixed entities of an ambient a, i.e., a^{is}, a^{ho}. Let $a \in n(P)$:[4]

$$\varepsilon(a) = (a^{\mathsf{ho}}, 1, \mathbf{0}) \prec (a^{\mathsf{is}}, 1, \mathbf{0})$$

$\varepsilon(a)$ indicates the *empty (fixed) state for ambient a* and its graphical representation is shown in Figure 6. The function $\Omega(a, P, k, \mathsf{act})$ returns a HABA state representing the process P contained inside the ambient a. The parameter k is used for dealing with cardinalities. The parameter act is a boolean that instructs Ω to construct the P configuration with the active or with the inactive representation of its ambients.

[4] From now on when convenient we write a state as a four tuple $\langle E, \mu, \mathcal{C}, \mathsf{P} \rangle$, where it is clear that the first three correspond to the configuration.

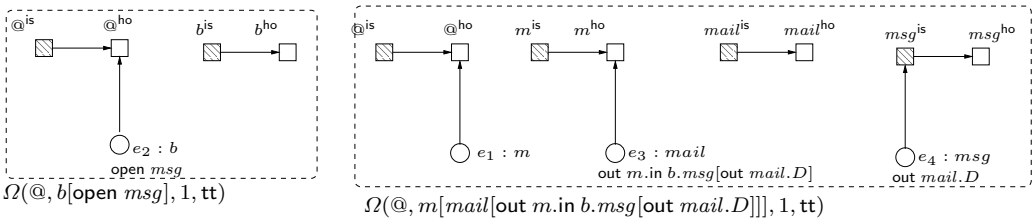

Fig. 7. HABA states (up to isomorphism) returned by $\Omega(@, b[\mathsf{open}\ msg], 1, \mathsf{tt})$ and $\Omega(@, m[mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]]], 1, \mathsf{tt})$.

Formally, $\Omega : \mathcal{N} \times \mathbf{Proc} \times \mathbb{M}^* \times \mathbb{B} \to \textsc{States}$ is given by:

$$\Omega(a, \mathbf{0}, k, \mathsf{act}) = \varepsilon(a)$$

$$\Omega(a, m[Q], k, \mathsf{act}) = \varepsilon(a) \uplus \Omega(m, Q, k, \mathsf{act}) \uplus \begin{cases} (e, k, \rho(Q)) \prec (a^{\mathsf{ho}}, 1, \mathbf{0}), \text{ if } \mathsf{act} \\ (e, k, \rho(Q)) \prec (m^{\mathsf{is}}, 1, \mathbf{0}) \text{ otherwise} \end{cases}$$

where $e{:}m$ is fresh

$$\Omega(a, Q_1|Q_2, k, \mathsf{act}) = \Omega(a, Q_1, k, \mathsf{act}) \uplus \Omega(a, Q_2, k, \mathsf{act})$$
$$\Omega(a, (\nu n)Q, k, \mathsf{act}) = \Omega(a, Q, k, \mathsf{act})$$
$$\Omega(a, !Q, k, \mathsf{act}) = \Omega(a, Q, *, \mathsf{act})$$
$$\Omega(a, N.Q, k, \mathsf{act}) = \varepsilon(a) \uplus \Omega(a, Q, k, \mathsf{ff})$$

The representation of the empty process inside a is given by the empty active state for a. The representation of $m[Q]$ in a comprehends the empty state for a, the sub-state of Q inside m and a configuration with a non fixed entity e standing for the copy of m in a. Depending on the parameter act, this representation can be either the active or the inactive one. Fixed entities have always cardinality 1, whereas entities representing copies, like e, are concrete if their ambient does not occur within the scope of replication. This is recorded in the parameter k which is assigned to e as cardinality. Moreover e has associated the set of capabilities $\rho(Q)$ to be executed. The representation of $Q_1|Q_2$ is given by the union of the representation of Q_1 and Q_2. When encountered, $\Omega(a, !Q, k)$ makes a recursive call changing the cardinality from k to $*$ This assigns cardinality $*$ to every non-fixed entity within the scope $\Omega(a, Q, *)$. Finally, the representation of $N.Q$ inside a has the empty state for a and the *inactive* representation for the process Q — which contains only non-enabled ambients since it is guarded by N. Therefore the recursive call $\Omega(a, Q, k, \mathsf{ff})$ is done with the explicit value $\mathsf{act} = \mathsf{ff}$.

Lemma 1. *For all $m \in n(P)$ and Q subprocess of P: if $\Omega(m, Q, k, \mathit{tt}) = \langle \gamma, \mathsf{P} \rangle$ then m is active.*

Proof. Straightforward by induction of the structure of Q.

Lemma 2. *For every process P, $a \in enab(P) \ \Rightarrow\ a$ is active in $\Omega(@, P, 1, \mathit{tt})$.*

Proof. Straightforward by induction of the structure of P and by the previous lemma.

Example 10. Figure 7 shows the HABA state representation for $\Omega(@, b[\mathsf{open}\ msg], 1.\mathsf{tt})$ as well as $\Omega(@, m[mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]]], 1, \mathsf{tt})$. In the latter, note the different representation between active ambients ($@$, m, $mail$) and inactive (msg). Composing the two states (by the union operation) together with $\Omega(@, \mathsf{open}\ msg, 1, \mathsf{tt})$ we obtain:

$$\Omega(@, b[\mathsf{open}\ msg], 1, \mathsf{tt}) \uplus \Omega(@, \mathsf{open}\ msg, 1, \mathsf{tt}) \uplus$$
$$\Omega(@, m[mail[\mathsf{out}\ m.\mathsf{in}\ b.msg[\mathsf{out}\ mail.D]]], 1, \mathsf{tt}) = \Omega(@, SYS_1, 1, \mathsf{tt}).$$

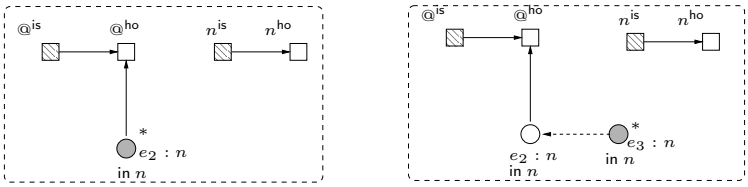

Fig. 8. Left: HABA state (up to isomorphism) returned by $\Omega(@, !n[\mathsf{in}\ n], 1, \mathsf{tt})$. Right: Its 1-canonical representation.

The state $\Omega(@, SYS_1, 1, \mathsf{tt})$ is depicted in Figure 3[5].

As different example, the left part of Figure 8 shows a state involving replication. By definition we have $\Omega(@, !n[\mathsf{in}\ n], 1, \mathsf{tt}) = \Omega(@, n[\mathsf{in}\ n], *, \mathsf{tt})$ therefore, the entity e_2 modelling the copies of n, becomes unbounded.

Finally, the next definition give the encoding of a process in a L-canonical HABA state. $\Omega(@, P, 1, \mathsf{tt})$ does not necessarily return a canonical state. That happens only in some circumstances involving unbounded entities.

Definition 4.4 (Process encoding). *The* process encoding *function* $\mathcal{D}$: **Proc** $\rightarrow$ States *is defined by:*

$$\mathcal{D}(P) = \langle \mathsf{cf}(\gamma), \mathsf{P}[@^{\mathsf{is}} \mapsto \rho(P)] \rangle$$

where $\Omega(@, P, 1, tt) = \langle \gamma, \mathsf{P} \rangle$.

The existence of a unique canonical form is proved in [11, 9][6]. For example, the state in the left part of Figure 8 is not L-canonical for any $L > 0$. The canonical form (shown for the case $L = 1$ on the right side of the same figure), automatically provides us with the correct representation needed for the simulation of P's behaviour. The definition $\mathcal{D}$ assigns to the inactive site $@^{\mathsf{is}}$ the set $\rho(P)$ containing the capabilities to be executed by @.

Pre-initial and initial state construction. For any process P, its pre-initial state is given by:

$$q_{pre} = \Omega(@, P, 1, \mathsf{ff}) \tag{7}$$

References among entities are defined — by the application of $\Omega(@, P, 1, \mathsf{ff})$ — in order to follow the (inactive) scheme introduced informally in Section 4.1 (in particular see Figure 2). From the previous definition we have that in the pre-initial state every ambient in P is inactive. Given the mapping $\mathcal{D}$, the definition of initial state is straightforward :

$$q_{in} = \mathcal{D}(P). \tag{8}$$

4.6 Configuration link manipulations

In the definition of the operational model, the computation of a process P corresponds to specific pointer manipulations that mimic the movements of P-ambients.

[5] Except for the capability of $@^{\mathsf{is}}$ that is be dealt with in a special way, see Definition 4.4.

[6] In these papers, the existence of the canoinical form is proved for a more general class of L-safe configurations. Here we don't need to consider L-safe configurations because of the special form of the configuration. For the well-indexed hypothesis of the processes there cannot be chains of entities with more than one unbounded entity.

Activating ambients. Active ambients in a HABA state modelling P correspond to enabled ambients in P. If an ambient is enabled it means that one of its instances can execute capabilities. During the computation, if a executes the capability N of the process $N.Q$, some ambients in Q may become enabled (since Q is no longer guarded by N). In the HABA state these ambients must be activated in order to have a consistent representation of the process. Here we define a function $act_{Q,a}(\gamma)$ that modifies the pointer structure of γ in such a way that every enabled ambient of the process Q rooted in a becomes active.

$$act_{Q,a}(\gamma) = (\gamma \backslash \gamma_{\Omega(a,Q,1,\mathsf{ff})}) \uplus \gamma_{\Omega(a,Q,1,\mathsf{tt})}.$$

The activation process consists of replacing the part of the configuration related to Q with inactive representation (i.e., $\Omega(a, Q, 1, \mathsf{ff})$) by the configuration where the enabled ambients of Q are active, i.e., $\Omega(a, Q, 1, \mathsf{tt})$.

Proposition 1. $\forall m \in n(Q) : (m \in enab(Q) \;\Leftrightarrow\; m \textit{ is active in } \Omega(a, Q, 1, tt))$.

Proof. Straightforward by induction on the structure of Q.

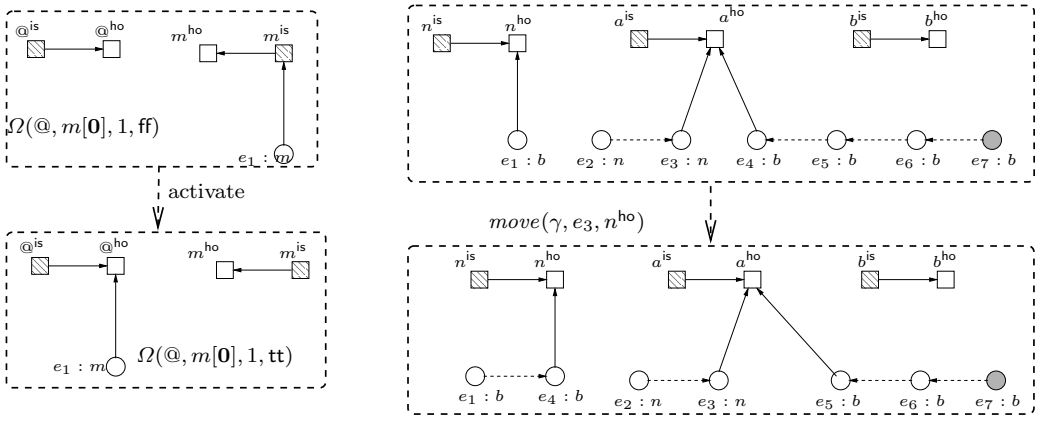

Fig. 9. Left: Rearrangements of pointers performed by $act_{m[\mathbf{0}],@}(\gamma)$ activating m. **Right:** Rearrangements of pointers for $move(\gamma, e_4, n^{\mathsf{ho}})$.

Figure 9 (left part) depicts the activation of the ambient m by the outer-most ambient @. It corresponds to $act_{m[\mathbf{0}],@}(\gamma)$.

Moving ambients. The execution of in and out involves moving entities from one host entity to another. In our setting, this activity requires some re-adjustments of pointers. For example, several instances of an ambient b contained in a form a queue (cf. Figure 4). If another copy $e{:}b$ enters a, it is enqueued in the first position of the queue. Symmetrically, if a copy of b moves out a (in Figure 4 the entity e_4) it must be dequeued and enqueued in the target ambient. We define the function $move(\gamma, e, \hat{e})$ that, in order to move e inside $\hat{e}$, manipulates γ according to the consistency requirements just described. $move : \textsc{Conf} \times Ent \times Ent \to \textsc{Conf}$ is defined by:[7]

$$move(\gamma, e, \hat{e}) = (E_\gamma, \mu_\gamma[e \mapsto \hat{e},\ \mu_\gamma^{-1}(e) \mapsto \mu_\gamma(e),\ e' \mapsto e], \mathcal{C}_\gamma)$$

where $\mu_\gamma(e') = \hat{e}$, $\mathsf{A}(e') = \mathsf{A}(e)$. The entity e moves inside $\hat{e}$ (the first pointer update of μ_γ). The second and the third update concern the consistency discussed above. In particular, if there is a queue starting with an entity $\mu^{-1}(e)$ pointing to e after the transition, $\mu^{-1}(e)$ points to $\mu(e)$ (i.e.,

[7] We write $f[x \mapsto y]$ the update of f at x, i.e., for function that is like f except on the argument x where we have $f[x \mapsto y](x) = y$.

second link update). Moreover, $\hat{e}$ may already have a queue (with first element e') representing copies of $\mathsf{A}(e)$. In this case, e is inserted in the first position (third link update)[8]. Figure 9 (right part) shows how the configuration changes when e_4 moves inside n^{ho}.

State update for the In/Out. There are more modifications to carry out during the execution of capabilities in and out than those performed by *move*. More specifically, assume there is an entity e that executes $N.Q$ where N is either in or out. The execution of N requires three kinds of tasks:

(i) pointer rearrangements moving e from its current location to the target location and those due for the consistency of the configuration. As we have seen, these updates are performed by *move*.
(ii) rearrangements due to the activation the ambients that in Q become enabled.
(iii) the set of capabilities $\mathsf{P}(e)$ should be updated in order to record that e has executed N and that it must continue with Q.

We define the function $\mathrm{IOUp}(q, N.Q, e, \hat{e})$ that manipulates the state q according to (i)–(iii) when e moves inside $\hat{e}$ because of the execution of N and continue with Q. $\mathrm{IOUp} : (\textsc{States} \times \mathbf{Proc} \times \mathit{Ent} \times \mathit{Ent}) \to \textsc{States}$ is defined by:

$$\mathrm{IOUp}(q, N.Q, e, \hat{e}) = \langle act_{Q,\mathsf{A}(e)} \circ move(\gamma_q, e, \hat{e}), \quad \mathsf{P}[e \mapsto \mathsf{P}(e) \backslash \{N.Q\} \cup \rho(Q)] \rangle$$

The configuration of the state is obtained applying first the rearrangements to move e inside $\hat{e}$ (this is done by *move*). After the manipulation of pointers activation takes place. The component $\mathsf{P}(e)$ is obtained deleting $N.Q$ and adding the subprocesses to be executed in Q, i.e., $\rho(Q)$.

Dissolving ambients. The operation $diss(\gamma, a^{\mathsf{ho}}, e{:}b)$ modifies γ so that the ambient a opens the ambient b represented by e. Pointers are updated in such a way a acquires b's inner ambients. More precisely, γ is updated first of all in order to link to a^{ho} some possible other copies of b contained in a, i.e., $\mu_\gamma^{-1}(e)$. Secondly, a acquires the ambients nested inside the copy of b (given by e), that is $\mu_\gamma^{-1}(b^{\mathsf{ho}})$. The function $diss : (\textsc{Conf} \times \mathit{Ent} \times \mathit{Ent}) \to \textsc{Conf}$ is defined by

$$diss(\gamma, a^{\mathsf{ho}}, e{:}b) = (E_\gamma \backslash \{e\},\ \mu_\gamma[\mu_\gamma^{-1}(e) \mapsto a^{\mathsf{ho}},\ \mu_\gamma^{-1}(b^{\mathsf{ho}}) \mapsto a^{\mathsf{ho}}],\ \mathcal{C}_\gamma \restriction E_\gamma \backslash \{e\}).$$

State update for open. The update of the state when open is executed involves a rearrangement of links done by *diss*, the activation of the ambients that become enabled and the update of the set of subprocesses that remain to be done by the entity executing open. $\mathrm{OpenUp} : (\textsc{States} \times \mathit{Ent} \times \mathit{Ent} \times \mathbf{Proc}) \to \textsc{States})$ is defined by:

$$\mathrm{OpenUp}(q, N.Q, e'{:}a, e) = \langle act_{Q,a} \circ diss(\gamma_q, a^{\mathsf{ho}}, e), \quad \mathsf{P}[e' \mapsto \mathsf{P}(e') \backslash \{N.Q\} \cup \rho(Q) \cup \mathsf{P}(e)] \rangle$$

Note that e' is the entity executing open. Since it is a copy of ambient a, its host is passed as parameter of *diss*. However, e' takes the processes $\mathsf{P}(e)$ that were supposed to be executed by e.

4.7 A HABA semantics of mobile ambients

We can now define the HABA $\mathcal{H}_P$ meant as an abstract model for the process P.

Definition 4.5. *Let P be a well-indexed process. The abstract semantics of P is the HABA $\mathcal{H}_P = \langle X_P, S, \to, I, \mathcal{F} \rangle$ where*

- $X_P = \{x_n \mid n \in n(P)\} \cup \{x_@\}$;
- $S \subseteq \textsc{States}$ *such that $q_{pre}, q_{in} \in S$, where for state $\langle \gamma, \mathsf{P} \rangle \in S$, the component $\mathsf{P}(e)$ is the set of processes that must be executed by e.*
- *let $\mathcal{R} \subseteq S \times (\mathit{Ent} \times \mathit{Ent} \to \mathbb{M}) \times S$ be the smallest relation satisfying the rules in Table 3. Then $\to = \mathcal{R} \cup \{(q_{pre}, \lambda_{pre}, q_{in})\} \cup \{(q, id, q) \mid \neg \exists q', \lambda : (q, \lambda, q') \in \mathcal{R}\}$;*

[8] Note that e' may not exist in which case this update does not take place.

– $\mathrm{dom}(I) = \{q_{pre}\}$ *and* $I(q_{pre}) = \langle \varnothing, \vartheta \rangle$ *where* $\vartheta(x_n) = n^{\mathsf{is}}$ $(n \in n(P))$.
– $\mathcal{F} = \{\{q \in S \mid 9\exists q', \lambda : q \rightarrow_\lambda q') \Rightarrow q = q'\}\}$.

The set X_P contains a logical variable for each ambient name occurring in P plus $x_@$ that is used to refer to the outer-most ambient. The transition relation $\rightarrow$ contains those transitions defined by means of the rules in Table 3 together with a transition from the pre-initial state to the intial state with reallocation $\lambda_{pre} = h_{\mathsf{cf}}^{-1}(\gamma_{\Omega(@,P,1,\mathsf{tt})})$ and an "artificial" self-loop for each deadlocked state in $\mathcal{R}$. The set of accept states $\mathcal{F}$ is defined as the set of states which only have a self-loop. Every computation reaches an accept state that is visited infinitely often, fulfilling therefore, the generalised Büchi acceptance condition. The set of initial states contains only the pre-initial state and the interpretation ϑ that allows us to refer to ambient names in *NTL* formulae.

In $\dfrac{e \in E^{\mathsf{m}}, \quad \mathsf{in}\ b.Q \in \mathsf{P}_q(e), \quad b_i \in \mathit{siblings}(e)}{q \rightarrow_\lambda \mathsf{cf}(\gamma''), \mathsf{P}'}$

where $\langle \gamma', \mathsf{P}' \rangle = \mathrm{IOUp}(q, \mathsf{in}\ b.Q, e, b_i^{\mathsf{ho}})$ and $\gamma'' \in \mathsf{SExp}(\gamma')$

Out $\dfrac{e \in E^{\mathsf{m}}, \quad \mathsf{out}\ b.Q \in \mathsf{P}(e), \quad e \prec b_i^{\mathsf{ho}} \quad a \in \mathit{parents}(b_i)}{q \rightarrow_\lambda \mathsf{cf}(\gamma''), \mathsf{P}'}$

where $\langle \gamma', \mathsf{P}' \rangle = \mathrm{IOUp}(q, \mathsf{out}\ b.Q, e, a^{\mathsf{ho}})$ and $\gamma'' \in \mathsf{SExp}(\gamma')$

Open $\dfrac{e \in E^{\mathsf{m}}_@, \quad \mathsf{open}\ b.Q \in \mathsf{P}(e), \quad e' : b_i \in \mathit{son}(\mathsf{A}(e))}{q \rightarrow_\lambda \mathsf{cf}(\gamma''), \mathsf{P}'}$

where $\langle \gamma', \mathsf{P}' \rangle = \mathrm{OpenUp}(q, \mathsf{open}\ b.Q, e, e')$ and $\gamma'' \in \mathsf{SExp}(\gamma')$

Bang $\dfrac{e \in E^{\mathsf{m}}_@, \quad !Q \in \mathsf{P}(e)}{q \rightarrow_\lambda \mathsf{cf}(\gamma'), \mathsf{P}'}$

where $\mathsf{P}' = \mathsf{P}_q[e \mapsto \mathsf{P}_q(e) \cup \rho(Q)]$ and $\gamma' \in \mathsf{SExp}(\mathit{act}_{Q,\mathsf{A}(e)}(\gamma_q))$

Table 3. Operational rules for Mobile ambients.

Operational rules. The execution of a capability $N.Q$, in a given state q, applies the following pattern: γ_q is first modified in order to achieve the needed link rearrangements that depends from the capability executed and P_q is updated. This is performed by IOUp (for in and out) or OpenUp (for open). Applying such pointer updating corresponds to an *id* reallocation from γ to γ'. Because of the rearrangements of the links, γ' may be not canonical. Therefore, we consider its safe expansion, i.e. $\mathsf{SExp}(\gamma')$ [9] and for each of its element γ'' we take the canonical form $\mathsf{cf}(\gamma'')$. The reallocation is defined as: $\lambda = h_{\mathsf{cf}} \circ h^{-1}(\gamma')$ where the morphism h is determined by the safe expansion of γ' and h_{cf} is the morphism giving the canonical form of $h^{-1}(\gamma')$. Again [11] shows that is a good definition of reallocation.

For a configuration γ, the transition relation uses some auxiliary concepts. The entities (in the source state q) that are allowed to move, called *mobile entities*, are

$$E^{\mathsf{m}} = \{e \in E_q^{\mathsf{c}} \mid \mathsf{A}(e) \text{ is active}, \ \ \mu_q(e) \in E_P^{\mathsf{ho}}\}.$$

[9] The safe expansion of a (possibly unsafe) configuration γ' is a finite set of L-safe configurations γ'' which are related to γ' by a morphism (i.e., they represent the same topological structure). Formally: $\mathsf{SExp}(\gamma') = \{\gamma'' \mid \gamma''$ is L-safe and $h : \gamma'' \rightarrow \gamma'\}$. See [11] for an exhaustive treatment.

Only concrete non-fixed entities modelling an active ambient and directly pointing a host can move. Other concrete entities do not move. For example in the state represented in Figure 4, we have $E^{\mathsf{m}} = \{e_1, e_3, e_4\}$. Entities such as e_2 and e_5 can only move when they are shifted to the beginning of the queue, i.e., when e_3 and e_4 have moved, respectively. In the rules $E^{\mathsf{m}}_{@} = E^{\mathsf{m}} \cup \{@^{\mathsf{is}}\}$, moreover, we use:

$$\begin{aligned} \mathit{siblings}(e) &= \{b_i \mid \exists e' : b_i \neq e \wedge (\mu_\gamma(e') = \mu_\gamma(e) \vee \mu_\gamma(e') = e)\} \\ \mathit{son}(a) &= \{e' \in E_q \backslash E^{\mathsf{is}}_P \mid e' \prec_q a^{\mathsf{ho}}\} \\ \mathit{parents}(b_i) &= \{a \mid \exists e' : b_i \in \mathit{son}(a)\} \end{aligned}$$

$\mathit{siblings}(e)$ is the set of ambients having an instance with the same parent of e. $\mathit{son}(a)$ returns the entities that are sons of the ambient a. $\mathit{parents}(b_i)$ is the set of parents of indexed ambients b_i. For example in the initial state in Figure 3 we have: $\mathit{siblings}(e_1) = \{b\}$, $\mathit{siblings}(e_3) = \varnothing$, $\mathit{parents}(\mathit{mail}) = \{m\}$, $\mathit{son}(@) = \{e_1, e_2\}$. We explain the rules for the transition relation.

- **In rule**: If a mobile entity e has among its enabled capabilities $\mathsf{in}\, b.Q$ and there exists a sibling ambient b then e moves inside b. Any indexed b_i must be considered.
- **Out rule**: If a mobile entity e executes $\mathsf{out}\, b.Q$ and its father is any indexed ambient b_i, i.e. $e \prec b_i^{\mathsf{ho}}$ then e must move in every ambient containing a copy of b_i.
- **Open rule**: A mobile entity e can execute $\mathsf{open}\, b$, if there exists a $\mathit{son}(\mathsf{A}(e))$ e' modelling a copy of b. Entity e' is dissolved and the component $\mathsf{P}(e)$ acquires the processes contained in $\mathsf{P}(e')$. This is done by the application of OpenUp.
- **Bang rule.** If a process $!Q$ is contained in the set of processes that e must execute, then $!Q$ is expanded using the equivalence $!Q \equiv Q|!Q$.

Note that we do not need structural rules for parallel composition, restriction, ambients since those constructs are implicitly represented in the configuration of a state.

Example 11. The HABA modelling SYS_1 of Example 2 is depicted in Figure 10. as we have seen in that example, for SYS_1 we wanted to check the secrecy property: (UA) *"no untrusted ambients can access D"* and expressed by the *NTL*-formula:

$$\Phi_{\mathsf{UA}} \equiv \exists x : x \rightsquigarrow x_{msg} \wedge \Diamond(x \not\rightsquigarrow x_{msg} \ \wedge \ x\uparrow \neq x_{mail}\uparrow \ \wedge \ \ x\uparrow \neq b).$$

No runs of the HABA satisfies ϕ therefore in SYS_1 only b can access the secret data D. □

Example 12. Part of the HABA for the process $!n[!\mathsf{in}\, n]$ is shown in Figure 11 for $L = 1$ and $M = 1$. With this parameters we can distinguish that there are 0,1,2, and more than 2 ambient nested in n at the same time. Increasing the values of L or M we can accomplish more accurate predictions at the cost of increasing the state space of the HABA.

Given a process P, the subpart of the state containing only fixed entities is $\varepsilon(n(P)) = \biguplus_{a \in n(P)} \varepsilon(a)$. We write λ^* as a short hand for a finite sequence of reallocation $\lambda', \lambda'', \ldots, \lambda^n$ and $q \rightarrow_{\lambda^*} q^n$ as a short hand for the sequence of transitions $q \rightarrow_{\lambda'} q' \rightarrow_{\lambda''} q'' \rightarrow_{\lambda'''} \cdots \rightarrow_{\lambda^n} q^n$ for some state $q', q'', \ldots$.

Theorem 1. *Let $\tilde{P}$ be a well-indexed process. If $\mathit{noi}(P) \rightarrow Q$ then there exists λ^* and a well-indexed process $\tilde{Q}'$ such that $\mathcal{D}(\tilde{P}) \rightarrow_{\lambda^*} (\mathcal{D}(\tilde{Q}') \uplus \varepsilon(n(\tilde{P})))$ and $\mathit{noi}(\tilde{Q}') \equiv Q$.*

Proof. See Appendix A. □

The previous theorem ensures that the abstract semantics for mobile ambients presented in this chapter provides a safe approximation of a process P. Although for many interesting processes the method provides rather precise information, some limitations occur on processes where name restriction is combined with replication. Like other analyses for mobile ambients based on static methods [8, 12, 13, 18], our semantics does not distinguish between processes $!(\nu n)P$ and $(\nu n)!P$ as instead it is done by the standard infinite semantics (cf. observation at page 2). However, our model is able to capture precise information on the number of copies of the same ambients that may be inside another ambient, therefore it is able to distinguish between P and $!P$. The precision can easily be increased.

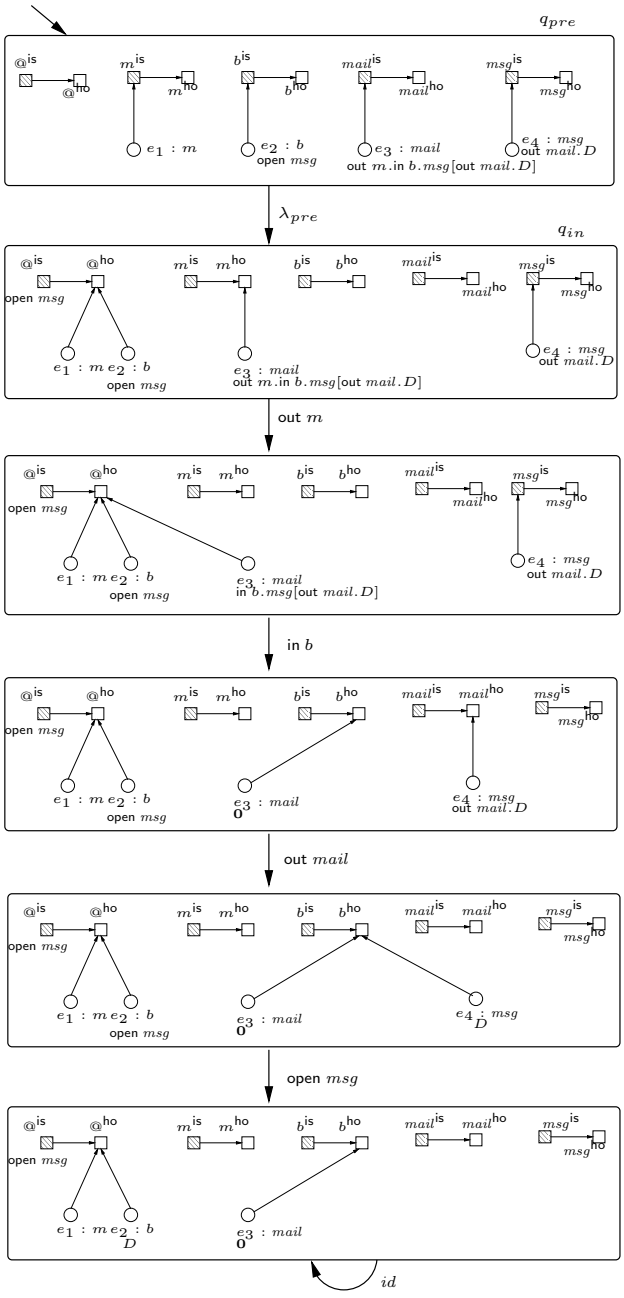

Fig. 10. The HABA representing SYS_1 of Example 2.

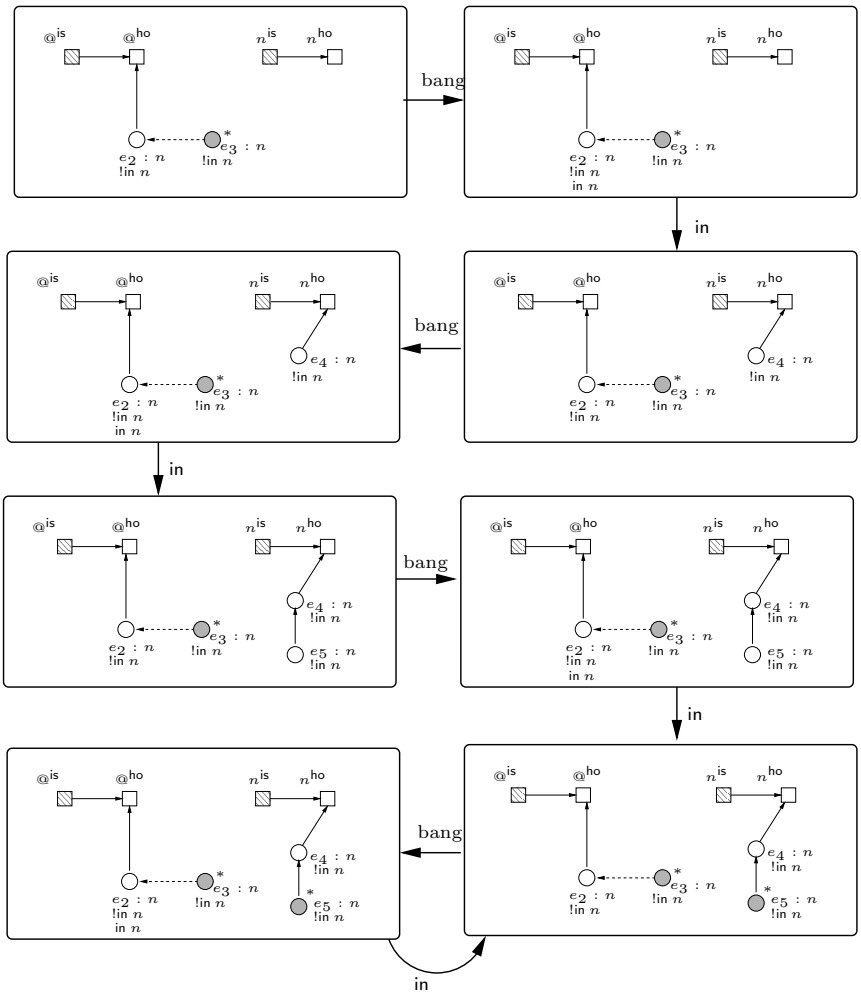

Fig. 11. Part of the HABA representing $!n[!\mathsf{in}\, n]$.

5 Conclusions

The analysis we have developed in this paper represents an alternative approach which goes beyond the analysis for MA based on abstract interpretation and control flow analysis found in the literature. A strong point of our technique seems to rely on its power of counting occurrences of ambients, as well as its flexibility on tuning the precision. Another advantage of our approach w.r.t. static analysis is that the model can be used to investigate properties of the evolution of the computation (via *NTL*). Beside the information "which ambient end up in in which other ambient" our model is able to answer other involved questions. For example, properties like "it is always the case that whenever a and b are inside n, a exit n before b". Or, "a copy of a does not leave b until another copy of a enters b".

References

1. C. Braghin, A. Cortesi, and R. Focardi. Control flow analysis of mobile ambients with security boundaries. In B. Jacobs and A. Rensink, editors, *FMOODS 2002*, pages 197–212. Kluwer, 2002.
2. L. Cardelli and A.D. Gordon. Mobile ambients. In *FOSSACS '98*. pages 140–155. Springer-Verlag, 1998.
3. L. Cardelli and A.D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *POLP 2000*, pages 365–377. ACM Press, 2000.
4. L. Caires. Behavioral and spatial observations in a logic for π-calculus In I. Waluklewicz, editor *FOSSACS 2004*, volume 2987 of *LNCS*, pp. 72–89, 2004.
5. L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Inf. and Comp.*, 186(2):194–235, 2003.
6. W. Charatonik, A.D. Gordon, and J.-M. Talbot. Finite-control mobile ambients. In D. Le Métayer, editor, *ESOP 2002*, volume 2305 of *LNCS*, pages 295–313. Springer, 2002.
7. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximation of fixed points. In *POPL77*, pages 238–252. ACM, 1977.
8. P. Degano, F. Levi, and C. Bodei. Safe ambients: Control flow analysis and security. In *Asian Computing Science Conference*, volume 1961 of *LNCS*, pages 199–214. Springer-Verlag, 2000.
9. D. Distefano. *On model checking the dynamics of object-based software: a foundational approach*. PhD thesis, University of Twente, Nov 2003.
10. D. Distefano, J.-P. Katoen, and A. Rensink. Who is pointing when to whom? on the automated verification of linked list structures. In *FSTTCS 2004*, LNCS, Springer-Verlag 2004. To appear.
11. D. Distefano, A. Rensink, and J.-P. Katoen. Who is pointing when to whom: on model-checking pointer structures. CTIT Tech. Report TR-CTIT-03-12, Dep. Comp. Sc., University of Twente, 2003.
12. R.R. Hansen, J.G. Jensen, F. Nielson, and H.R. Nielson. Abstract interpretation of mobile ambients. In A.Cortesi and G. Filé, editors, *SAS '99*, volume 1694 of *LNCS*, pages 135–148. Springer, 1999.
13. F. Levi and S. Maffeis. An abstract interpretation framework for analysing mobile ambients. In P. Cousot, editor, *SAS 2001*, volume 2126 of *LNCS*, pages 395–411. Springer, 2001.
14. F. Levi and S. Maffeis. On abstract interpretation of mobile ambients. *Inf. Comp.*, 188(2):179–240, 2004.
15. F. Levi and D. Sangiorgi. Controlling interference in ambients. In *POPL 2000*, pages 352–364. ACM Press, 2000.
16. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL'85*, pages 97–107. ACM, 1985.
17. U. Montanari and M. Pistore. An introduction to history-dependent automata. In A. Gordon, A. Pitts, and C. Talcott, editors, *HOOTS II*, volume 10 of *ENTCS*. Elsevier Science Publishers, 1997.
18. F. Nielson, H.R. Nielson, R.R. Hansen, and J.G. Jensen. Validating farewalls in mobile ambients. In J.C.M. Baeten and S. Mauw, editors, *CONCUR '99*, vol. 1664 of *LNCS*, pp. 463–477. Springer, 1999.
19. A. Pnueli. The temporal logic of programs. In *Proceedings FOCS-77*, pages 46–57. IEEE Computer Society Press, 1977.
20. M. Sagiv, T. Reps, and R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. *TOPLAS*, 20(1):1–50, 1998.

A Proofs

In order to prove Theorem 1, we first prove some auxiliary results. The following two lemmas are properties of HABA states modelling ambient processes.

Lemma 3. *Given a morphism h there exists morphisms h_1, h_2 such that*

$$h^{-1}(\Omega(a, P, k, \mathsf{act}) \uplus \Omega(a, Q, k', \mathsf{act}')) = h_1^{-1}(\Omega(a, P, k, \mathsf{act})) \uplus h_2^{-1}(\Omega(a, Q, k', \mathsf{act}')).$$

Proof. Defining $h_1 = h \restriction \{e \mid e \in \Omega(a, P, k, \mathsf{act})\}$ and $h_2 = h \restriction \{e \mid e \in \Omega(a, Q, k, \mathsf{act})\}$ it can be verified that both h_2 and h_2 are morphisms and they satisfy the equality. □

Lemma 4. *Let $\tilde{P}$ and $\tilde{Q}$ be two well-indexed processes such that $n(\tilde{P}) \neq n(\tilde{Q})$. Given two h_1, h_2 morphisms then*

a) $\mathsf{cf}(h_1^{-1}(\Omega(a, P, k, \mathsf{act})) \uplus h_2^{-1}(\Omega(a, Q, k', \mathsf{act}'))) = \mathsf{cf}(h_1^{-1}(\Omega(a, P, k, \mathsf{act}))) \uplus \mathsf{cf}(h_2^{-1}(\Omega(a, Q, k', \mathsf{act}')))$.
b) $\Omega(a, P, k, \mathsf{act}) \uplus \Omega(a, Q, k', \mathsf{act}') = \Omega(a, Q, k', \mathsf{act}') \uplus \Omega(a, P, k, \mathsf{act})$.
c) $\mathcal{D}(\tilde{P}) = \mathsf{cf}(\Omega(@, P, 1, \mathit{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(\tilde{P})]$.

Proof. (Sketch) The three properties a), b) c) are straightforward using the definition of $\mathcal{D}$, cf and the fact that $\tilde{P}$ and $\tilde{Q}$ are well-indexed. □

One of the most involved part of the proof of Theorem 1 consists on the simulation of the steps performed by a process because of the application of (Red $\equiv$). Here we follow the pattern used in the proof of completeness in [14]. It is easy to define an equivalence relation $\equiv_i$ on well-indexed processes and to prove that $P \equiv Q$ if and only if $\tilde{P} \equiv_i \tilde{Q}$ (see [14] for one way to define it). We define now an equivalence relation $\cong$ on HABA states which is in some sense equivalent to $\equiv_i$, i.e., such that given two well-indexed processes $\tilde{P}$ and $\tilde{Q}$, if $\mathcal{D}(P) \cong \mathcal{D}(\tilde{Q})$ then $\tilde{P} \equiv_i \tilde{Q}$ (and vice-versa). Lemma 5 and Lemma 6 prove the equivalence of the two relations and Lemma 7 proves that equivalent states simulate each other.

Definition A.1. $\cong$ *is defined as the minimal reflexive, symmetric and transitive relation over (the set of states) S such that $\langle \gamma_1, \mathsf{P}_1 \rangle \cong \langle \gamma_2, \mathsf{P}_2 \rangle$ if one of the following hold:*

1. $\langle \gamma_1, \mathsf{P}_1 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}' \rangle \uplus \Omega(a, M.P, 1, \mathsf{act})$ *and* $\langle \gamma_2, \mathsf{P}_2 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\} \rangle \uplus \Omega(a, M.Q, 1, \mathsf{act})$ *and* $P \equiv Q$;
2. $\langle \gamma_1, \mathsf{P}_1 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}' \rangle \uplus \Omega(a, !P, 1, \mathsf{act})$ *and* $\langle \gamma_2, \mathsf{P}_2 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'\{!Q/!P\} \rangle \uplus \Omega(a, !Q, 1, \mathsf{act})$ *and* $P \equiv Q$;
3. $\langle \gamma_1, \mathsf{P}_1 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \Omega(a, b[P], 1, \mathsf{act})$ *and* $\langle \gamma_2, \mathsf{P}_2 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \Omega(a, b[Q], 1, \mathsf{act})$ *and* $P \equiv Q$;
4. $\langle \gamma_1, \mathsf{P}_1 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \Omega(b, a_i[Q], *, \mathsf{act})$ *and* $\langle \gamma_2, \mathsf{P}_2 \rangle = \langle \gamma, \mathsf{P} \rangle \uplus \Omega(b, a_i[Q], *, \mathsf{act}) \uplus \Omega(b, a_j[Q], 1, \mathsf{act})$ *and* $\gamma_2\{a_i/a_j\} = \gamma_1$;
5. $\gamma_2 = \gamma_1$ *and there exists e such that* $!Q \in \mathsf{P}_1(e)$ *and* $\mathsf{P}_2(e') = \begin{cases} \mathsf{P}_1(e') & \text{if } e' \neq e \\ \mathsf{P}_1(e) \cup \rho(Q) & \text{otherwise} \end{cases}$

Lemma 5. *Let P and Q be two processes. If $P \equiv Q$ then $\mathcal{D}(P) \cong \mathcal{D}(Q)$.*

Proof. It is easy to verify by induction on the depth of $P \equiv Q$ that for every case of Table 1 the function $\mathcal{D}$ produces states related by $\cong$. □

Lemma 6. *If $\mathcal{D}(P) \cong q$ then there exists Q such that $\mathcal{D}(Q) = q$ and $P \equiv Q$.*

Proof. By induction on the structure of P.

– case $P = M.P'$. We have

$$\begin{aligned}
\mathcal{D}(M.P') &= \langle \mathsf{cf}(\Omega(@, M.P', 1, \mathsf{tt})), \mathsf{P}_{\Omega(@, M.P', 1, \mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(M.P')] \rangle \\
&= \mathsf{cf}(\Omega(@, M.P', 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto M.P'] \\
&\cong \mathsf{cf}(\Omega(@, M.Q', 1, \mathsf{tt})) \uplus (\varepsilon(@)[@^{\mathsf{is}} \mapsto M.P'])\{M.Q'/M.P'\} \\
&= \mathsf{cf}(\Omega(@, M.Q', 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto M.Q'] \\
&= \langle \mathsf{cf}(\Omega(@, M.Q', 1, \mathsf{tt})), \mathsf{P}_{\Omega(@, M.Q', 1, \mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(M.Q')] \rangle \\
&= \mathcal{D}(M.Q')
\end{aligned}$$

Therefore let $q = \mathsf{cf}(\Omega(@, M.Q', 1, \mathsf{tt})) \uplus (\varepsilon(@)[@^{\mathsf{is}} \mapsto M.Q'])$ for some $Q' \equiv P'$. Taking $Q = M.Q'$ we have $q = \mathcal{D}(M.Q')$ and $P \equiv Q$.

– case $P = !P'$. Similar to the previous case. In fact:

$$\begin{aligned}\mathcal{D}(!P') &= \langle \mathsf{cf}(\Omega(@, !P', 1, \mathsf{tt})), \mathsf{P}_{\Omega(@,!P',1,\mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(!P')]\rangle \\ &= \mathsf{cf}(\Omega(@, !P', 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto !P'] \\ &\cong \mathsf{cf}(\Omega(@, !Q', 1, \mathsf{tt})) \uplus (\varepsilon(@)[@^{\mathsf{is}} \mapsto !P'])\{!Q'/!P'\} \\ &= \mathsf{cf}(\Omega(@, !Q', 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto !Q'] \\ &= \langle \mathsf{cf}(\Omega(@, !Q', 1, \mathsf{tt})), \mathsf{P}_{\Omega(@,!Q',1,\mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(!Q')]\rangle \\ &= \mathcal{D}(!Q')\end{aligned}$$

where $Q' \equiv P'$. We conclude by taking $P = !P'$ and $Q = !Q'$.

– case $P = b[P']$. Similar to the previous cases. In fact:

$$\begin{aligned}\mathcal{D}(b[P']) &= \langle \mathsf{cf}(\Omega(@, b[P'], 1, \mathsf{tt})), \mathsf{P}_{\Omega(@,b[P'],1,\mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(b[P'])]\rangle \\ &= \mathsf{cf}(\Omega(@, b[P'], 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \mathbf{0}] \\ &\cong \mathsf{cf}(\Omega(@, b[Q'], 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(b[Q'])] \\ &= \mathcal{D}(b[Q'])\end{aligned}$$

where $Q' \equiv P'$ and $P = b[P']$, $Q = b[Q']$.

– case $P = R_1|R_2$.

$$\begin{aligned}\mathcal{D}(R_1|R_2) &= \langle \mathsf{cf}(\Omega(@, R_1|R_2, 1, \mathsf{tt})), \mathsf{P}_{\Omega(@,R_1|R_2,1,\mathsf{tt})}[@^{\mathsf{is}} \mapsto \rho(R_1|R_2)]\rangle \\ &= \mathsf{cf}(\Omega(@, R_1, 1, \mathsf{tt})) \uplus \mathsf{cf}(\Omega(@, R_2, 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(R_1) \cup \rho(R_2)] \\ &= \mathsf{cf}(\Omega(@, R_1, 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(R_1)] \uplus \mathsf{cf}(\Omega(@, R_2, 1, \mathsf{tt})) \uplus \varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(R_2)] \\ &= \mathcal{D}(R_1) \uplus \mathcal{D}(R_2)\end{aligned}$$

Let q_1, q_2 such that $\mathcal{D}(R_1) \cong q_1$ and $\mathcal{D}(R_2) \cong q_2$ then by induction hypothesis there exists Q_1 and Q_2 such that $\mathcal{D}(Q_1) = q_1$ and $\mathcal{D}(Q_2) = q_2$. But then $\mathcal{D}(R_1|R_2) = \mathcal{D}(R_1) \uplus \mathcal{D}(R_2) \cong \mathcal{D}(Q_1) \uplus \mathcal{D}(Q_2) = \mathcal{D}(Q_1|Q_2)$ and $R_1|R_2 \equiv Q_1|Q_2$.

– case $P = (\nu n)R$.

$$\begin{aligned}\mathcal{D}((\nu n)R) &= \langle \mathsf{cf}(\Omega(@, (\nu n)R, 1, \mathsf{tt})), \uplus\rangle\varepsilon(@)[@^{\mathsf{is}} \mapsto \rho((\nu n)R)] \\ &= \langle \mathsf{cf}(\Omega(@, R, 1, \mathsf{tt})), \uplus\rangle\varepsilon(@)[@^{\mathsf{is}} \mapsto \rho(R)] \\ &= \mathcal{D}(R)\end{aligned}$$

If $q \cong \mathcal{D}(R)$ then there exists Q' such that $\mathcal{D}(Q') = q$ and $Q' \equiv R$. Therefore, $\mathcal{D}((\nu n)R) = \mathcal{D}(R) \cong S = \mathcal{D}(Q') = \mathcal{D}((\nu n)Q')$, and we can conclude by taking $Q = (\nu n)Q'$.

□

We write λ^* as a short hand for a finite sequence of reallocation $\lambda', \lambda'', \ldots, \lambda^n$ and $q \rightarrow_{\lambda^*} q^n$ as a short hand for the sequence of transitions $q \rightarrow_{\lambda'} q' \rightarrow_{\lambda''} q'' \rightarrow_{\lambda'''} \cdots \rightarrow_{\lambda^n} q^n$ for some state $q', q'', \ldots$.

Lemma 7. *If $q_1 \cong q_2$ and $q_1 \rightarrow_{\lambda_1} q_1'$ then there exists λ_2^* such that $q_2 \rightarrow_{\lambda_2^*} q_2'$ and $q_1' \cong q_2'$.*

Proof. We need to show that for any cases of Definition A.1 the transition $q_1 \rightarrow_{\lambda_2} q_1'$ is simulated by a sequence of transitions $q_2 \rightarrow_{\lambda_2^*} q_2'$.

1. Assume that $q_1 \cong q_2$ follows by case 1 of Definition A.1. Then is must be

$$\begin{aligned}q_1 &= \langle \gamma_1, \mathsf{P}_1\rangle = \langle \gamma, \mathsf{P}\rangle \uplus \langle \gamma', \mathsf{P}'\rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) \\ q_2 &= \langle \gamma_2, \mathsf{P}_2\rangle = \langle \gamma, \mathsf{P}\rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\}\rangle \uplus \Omega(a, M.Q, 1, \mathsf{act})\end{aligned}$$

and $P \equiv Q$.
Now, the transition $q_1 \rightarrow_{\lambda_1} q_1'$ is generated by one of the rules of Table 3. Let's analyse the different cases.

Bang In this case, we have $!R \in \mathsf{P}_1(e)$ for some e and R. Note that $e \notin \gamma_{\Omega(a,M.P,1,\mathsf{act})}$ because by definition $\Omega(a, M.P, 1, \mathsf{act}) = \varepsilon(a) \uplus \Omega(a, P, 1, \mathsf{ff})$ therefore entities in this part of the sub-state are inactive, i.e. cannot perform any action. Thus let us assume that $e \in E_\gamma$. Then the only part of the state involved on the transition is $\langle \gamma, \mathsf{P} \rangle$. Therefore:

$$q_1 \rightarrow_\lambda = \langle \gamma, \mathsf{P}[e \mapsto \mathsf{P}(e) \cup \rho(R)] \rangle \uplus \langle \gamma', \mathsf{P}' \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) = q_1'$$

But then also q_2 can perform the same transition since $\langle \gamma, \mathsf{P} \rangle$ is a sub-state of q_2. In particular we have::

$$q_2 \rightarrow_\lambda \langle \gamma, \mathsf{P}[e \mapsto \mathsf{P}(e) \cup \rho(R)] \rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\} \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) = q_2'$$

Hence we can conclude $q_1' \cong q_2'$.
If $e \in E_{\gamma'}$ then the only part of the state involved on the transition is $\langle \gamma', \mathsf{P}' \rangle$. Therefore:

$$q_1 \rightarrow_\lambda \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'[e \mapsto \mathsf{P}'(e) \cup \rho(R)] \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) = q_1'$$

Again also q_2 can perform the same transition and therefore we have:

$$\begin{aligned} q_2 \rightarrow_\lambda & \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\}[e \mapsto \mathsf{P}'\{M.Q/M.P\}(e) \cup \rho(R\{M.Q/M.P\})] \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) \\ = & \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\}[e \mapsto (\mathsf{P}'(e) \cup \rho(R))\{M.Q/M.P\}] \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) \\ = & \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma', \mathsf{P}'[e \mapsto \mathsf{P}'(e) \cup \rho(R)]\{M.Q/M.P\} \rangle \uplus \Omega(a, M.P, 1, \mathsf{act}) \\ = & \ q_2' \end{aligned}$$

Again we can conclude $q_1' \cong q_2'$.

In In this case we have a capability $\mathsf{in}\ b.R \in \mathsf{P}_1(e)$. We have $q_1' = \langle \gamma_1', \mathsf{P}_1' \rangle$ where $\gamma_1' = act_{Q,\mathsf{A}(e)}(move(\gamma_1, e, b^{\mathsf{ho}}))$ and $\mathsf{P}_1' = \mathsf{P}_1[e \mapsto \mathsf{P}_1(e) \backslash \{\mathsf{in}\ b.R\} \cup \rho(R)]$. For what we have observed in case of **Bang** $e \in E_\gamma \cup E_{\gamma'}$.
If $e \in E_{\gamma'}$, the only part of the configuration affected by the transition is $\langle \gamma', \mathsf{P}' \rangle$ for q_1 and — respectively — $\langle \gamma', \mathsf{P}'\{M.Q/M.P\} \rangle$ for q_2, whereas the rest of the configurations remains unchanged. Hence the resulting states are of the form:

$$\begin{aligned} q_1' &= \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma'', \mathsf{P}'' \rangle \uplus \Omega(a, M.P, 1.\mathsf{act}) \\ q_2' &= \langle \gamma, \mathsf{P} \rangle \uplus \langle \gamma'', \mathsf{P}''\{M.Q/M.P\} \rangle \uplus \Omega(a, M.Q, 1.\mathsf{act}) \end{aligned}$$

for the same γ'' which is the result of the pointer update performed by the application of the capability to γ. Therefore also in this case $q_1 \cong q_2$.
Similarly if $e \in E_\gamma$, we easily have that the resulting states are of the form:

$$\begin{aligned} q_1' &= \langle \gamma'', \mathsf{P}'' \rangle \uplus \langle \gamma', \mathsf{P}' \rangle \uplus \Omega(a, M.P, 1.\mathsf{act}) \\ q_2' &= \langle \gamma'', \mathsf{P}'' \rangle \uplus \langle \gamma', \mathsf{P}'\{M.Q/M.P\} \rangle \uplus \Omega(a, M.Q, 1.\mathsf{act}) \end{aligned}$$

because in both state only the sub-state $\langle \gamma, \mathsf{P} \rangle$ evolves into $\langle \gamma'', \mathsf{P}'' \rangle$. Hence, $q_1 \cong q_2$.

Out Similar to **In** case.

Open Similar to **In** case.

2. If $q_1 \cong q_2$ because of case 2 of Definition A.1 similar to case 1.
3. If $q_1 \cong q_2$ because of case 3 of Definition A.1 similar to case 1.
4. Assume $q_1 \cong q_2$ because of case 4 of Definition A.1. If $q_2 \rightarrow_\lambda q_2'$ and the transition involves the sub-state $\langle \gamma, \mathsf{P} \rangle \uplus \Omega(b, a_i[Q], *, \mathsf{act})$ then the same step can be taken be q_1.
   If the transition is performed in the sub-configuration $\Omega(b, a_j[Q], 1, \mathsf{act})$ then in q_1 this step can be simulated by by a copy of the ambient a_i in its sub-configuration $\Omega(b, a_i[Q], *, \mathsf{act})$. The existence of this copy of a_i is ensured by the canonical form. Moreover, $\gamma_1'\{a_i/a_j\} = \gamma_2'$.
5. Assume $q_1 \cong q_2$ because of case 5 of Definition A.1. We distinguish two cases. Any transition performed by q_1 can be clearly performed by q_2 as well. This is because q_2 has the same topology as q_1 and every entity has a superset of the capabilities that must be performed by q_1.

On the contrary, assume q_2 performs a transitions $q_2 \rightarrow_{\lambda_2} q_2'$ because of some capability M of an entity e. If this transition does not involve the process Q then q_1 can perform the same transition since $M \in \mathsf{P}_1(e)$. However, if M occur in Q then q_1 can perform in e a **(Bang)** transition $q_1 \rightarrow_{\lambda_1} q_1'$ so that $\mathsf{P}_1'(e) = \mathsf{P}_2(e)$. At that point q_1' can perform the same transition as q_2, i.e., $q_1 \rightarrow_{\lambda_1} q_1' \rightarrow_{\lambda_2} q_1''$ where $q_1'' \cong q_2'$.

□

The next proof uses the following facts implied by the definition of Ω:

$$\forall a \in n(P) : \forall Q : \varepsilon(a) \uplus \Omega(a, Q, k, \mathsf{tt}) = \Omega(a, Q, k, \mathsf{tt}) \tag{9}$$

$$\forall Q : \varepsilon(n(Q)) \uplus \Omega(@, Q, k, \mathsf{tt}) = \Omega(@, Q, k, \mathsf{tt}). \tag{10}$$

Moreover the definition of union of states implies that:

$$\forall a : \varepsilon(a) \uplus \varepsilon(a) = \varepsilon(a)$$

Theorem 1. Let $\tilde{P}$ be a well-indexed process. If $noi(\tilde{P}) \rightarrow Q$ then there exists λ^* and a well-indexed process $\tilde{Q}'$ such that $\mathcal{D}(\tilde{P}) \rightarrow_{\lambda^*} (\mathcal{D}(\tilde{Q}') \uplus \varepsilon(n(\tilde{P})))$ and $noi(\tilde{Q}') \equiv Q$.

Proof. By induction on the depth of the derivation of $P \rightarrow Q$. The last rule used in the transition $P \rightarrow Q$ can be any of those listed in Table 2. We distinguish the possible cases[10]

– case (Red In). We have that $\tilde{P}$ is of the form $\tilde{P} = n_i[\mathsf{in}\ m.\tilde{P}'|\tilde{Q}'']|m_j[\tilde{R}']$ and $P = n[\mathsf{in}\ m.P'|Q'']|m[R']$. Therefore by the (Red In) we have $Q = m[n[P'|Q'']|R']$. Let $\tilde{Q}' = m_j[n_i[\tilde{P}'|\tilde{Q}'']|\tilde{R}']$, we have obviously $noi(\tilde{Q}') \equiv Q$.
By definition of $\Omega(@, \tilde{P}, 1, \mathsf{tt})$ we have:

$$\begin{aligned}\Omega(@, \tilde{P}, 1, \mathsf{tt}) &= \Omega(@, n_i[\mathsf{in}\ m.\tilde{P}'|\tilde{Q}''], 1, \mathsf{tt}) \uplus \Omega(@, m_j[\tilde{R}'], 1, \mathsf{tt}) \\ &= \varepsilon(@) \uplus \varepsilon(n_i) \uplus \Omega(n_i, \mathsf{in}\ m.\tilde{P}'|\tilde{Q}'', 1, \mathsf{ff}) \uplus \Omega(m_i, \tilde{R}', 1, \mathsf{tt}) \uplus \\ &\quad (e^{\mathsf{c}}_{n_i}, 1, \rho(\mathsf{in}\ m.\tilde{P}'|\tilde{Q}'')) \prec (@^{\mathsf{ho}}, 1, \mathbf{0}) \ \uplus \ (e^{\mathsf{c}}_{m_j}, 1, \rho(\tilde{R}')) \prec (@^{\mathsf{ho}}, 1, \mathbf{0})\end{aligned}$$

Symmetrically,

$$\begin{aligned}\Omega(@, \tilde{Q}', 1) = \varepsilon(@) \uplus \varepsilon(m_j) \uplus \varepsilon(n_i) \uplus \Omega(n_i, \tilde{P}'|\tilde{Q}'', 1, \mathsf{tt}) \uplus \Omega(m_j, \tilde{R}', 1, \mathsf{tt}) \uplus \\ (e^{\mathsf{c}}_{m_j}, 1, \rho(\tilde{R}')) \prec (@^{\mathsf{ho}}, 1, \mathbf{0}) \ \uplus \ (e^{\mathsf{c}}_{n_i}, 1, \rho(\tilde{P}'|\tilde{Q}'')) \prec (m^{\mathsf{ho}}_j, 1, \mathbf{0})\end{aligned}$$

Let $\mathcal{D}(\tilde{P}) = (\mathsf{cf}(\gamma_{\Omega(@,\tilde{P},1,\mathsf{tt})}), \mathsf{P}_{\Omega(@,\tilde{P},1,\mathsf{tt})}[@^{\mathsf{is}} \mapsto \mathbf{0}])$. The configuration $\gamma_{\mathcal{D}(\tilde{P})}$ — even after the transformation of representation produced by the application of cf — can be considered as the union of several configurations. For the concrete part of $\gamma_{\Omega(@,\tilde{P},1,\mathsf{tt})}$, for example the following must be sub-configurations of $\gamma_{\mathcal{D}(\tilde{P})}$:

$$\gamma' = (e^{\mathsf{c}}_{n_i}, 1, \mathsf{P}(e^{\mathsf{c}}_{n_i})) \prec (@^{\mathsf{ho}}, 1, \mathbf{0})$$
$$\gamma'' = (e^{\mathsf{c}}_{m_j}, 1, \mathsf{P}(e^{\mathsf{c}}_{m_j})) \prec (@^{\mathsf{ho}}, 1, \mathbf{0})$$

In fact, we can safely assume that even after the application of h_{cf}, there is a concrete entity of type m_j and a concrete one of type n_i that have a reference to $@^{\mathsf{ho}}$ (by L-canonicity). Since the actual entity does not matter, we can assume for simplicity that these are precisely $e^c_{n_i}, e^c_{m_j}$. Hence, up to isomorphism, $\gamma_{\mathcal{D}(\tilde{P})}$ is of the form

$$\gamma_{\mathcal{D}(\tilde{P})} = \gamma' \uplus \gamma'' \uplus \gamma'''$$

for some γ'''.

[10] In this proof, we write indexed processes with a tilde. The same process without tilde is the result of the application of the function noi. That is, for any indexed process $\tilde{R}$, $noi(\tilde{R}) = R$.

Rule **In** of Table 3 starts by the application of

$$\mathrm{IOUp}(\mathcal{D}(\tilde{P}), \mathsf{in}\, m.\tilde{P}'|\tilde{Q}'', e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j)$$

that in turn applies first $move(\gamma_{\mathcal{D}(\tilde{P})}, e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j)$. The latter substitute γ' by the appropriate pointer update dictated by its definition. To the resulting configuration, activation applies:

$$\begin{aligned}
&act_{\tilde{P}'|\tilde{Q}'',n_i}(\gamma_{\varepsilon(@)} \uplus (\{e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j\}, \{(e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j)\}\mathbf{1}_{\{e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j\}}) \uplus \\
&\quad (\{e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}}\}, \{(e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}})\}, \mathbf{1}_{\{(e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}})\}}) \uplus \\
&\quad \gamma_{\Omega(n_i, \mathsf{in}\, m.\tilde{P}'|\tilde{Q}'', 1, \mathsf{ff})} \uplus \gamma_{\Omega(m_i, \tilde{R}', 1, \mathsf{tt})}) \\
= \gamma_{\varepsilon(@)} &\uplus (\{e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j\}, \{(e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j)\}, \mathbf{1}_{\{e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j\}}) \uplus \\
&(\{e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}}\}, \{(e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}})\}, \mathbf{1}_{\{(e^{\mathsf{c}}_{m_j}, @^{\mathsf{ho}})\}}) \uplus \\
&\gamma_{\Omega(n_i, \tilde{P}'|\tilde{Q}'', 1, \mathsf{tt})} \uplus \gamma_{\Omega(m_i, \tilde{R}', 1, \mathsf{tt})})
\end{aligned}$$

Therefore, we have (also using (9)) that the configuration of

$$\mathrm{IOUp}(\mathcal{D}(\tilde{P}), \mathsf{in}\, m.\tilde{P}'|\tilde{Q}, e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j)$$

is $\gamma_{\Omega(@,\tilde{Q}',1)}$. For the P component we have:

$$\mathsf{P}_{\mathcal{D}(\tilde{P})}[e^{\mathsf{c}}_{n_i} \mapsto \mathsf{P}_{\mathcal{D}(\tilde{P})}(e)\backslash \mathsf{in}\, m.\tilde{P}'|\tilde{Q}'' \cup \rho(\tilde{P}'|\tilde{Q}'')] = \mathsf{P}_{\mathcal{D}(\tilde{P})}[e^{\mathsf{c}}_{n_i} \mapsto \rho(\tilde{P}'|\tilde{Q}'')]$$

which this is precisely: $\mathsf{P}_{\Omega(@,\tilde{Q}',1,\mathsf{tt})}$. Thus we have:

$$\begin{aligned}
\mathrm{IOUp}(\mathcal{D}(\tilde{P}), \mathsf{in}\, m.\tilde{P}'|\tilde{Q}, e^{\mathsf{c}}_{n_i}, m^{\mathsf{ho}}_j) &= \langle \gamma_{\Omega(@,\tilde{Q},1,\mathsf{tt})}, \mathsf{P}_{\Omega(@,\tilde{Q},1,\mathsf{tt})} \rangle \\
&= \Omega(@, \tilde{Q}', 1, \mathsf{tt})
\end{aligned}$$

Note that since $n(\tilde{Q}') = n(\tilde{P})$, and by (10) we have

$$\Omega(@, \tilde{Q}', 1, \mathsf{tt}) \uplus \varepsilon(n(\tilde{P})) = \Omega(@, \tilde{Q}', 1, \mathsf{tt})$$

Now, according to the **In** rule of Table 3, the target state is the canonical form of a state in the safe expansion, together with the updated set of capabilities we have just computed. Thus:

$$\mathcal{D}(\tilde{P}) \rightarrow_\lambda (h_{\mathsf{cf}} \circ h^{-1}(\gamma'), \mathsf{P}_{\Omega(@,\tilde{Q}',1,\mathsf{tt})}).$$

for all $(\gamma', h) \in \mathsf{SExp}(\gamma_{\Omega(@,\tilde{Q}',1,\mathsf{tt})})$. Among the canonical configurations (for the properties of the canonical form) there must be a γ'' where $\mathsf{cf}(\gamma'')$ correspond to $\mathcal{D}(\tilde{Q}')$.

– case (Red Out). Similar to the (In Rule) case.
– case (Red Open). We have $\tilde{P} = \mathsf{open}\, n.\tilde{P}'|n_i[\tilde{R}]$ and $noi(\tilde{P}) = \mathsf{open}\, n.P'|n[R]$ and $Q = P'|R$. Let $\tilde{Q}' = \tilde{P}'|\tilde{R}$ then $noi(\tilde{Q}') \equiv Q$.
By definition of $\Omega(@, \tilde{P}, 1, \mathsf{tt})$ we have:

$$\begin{aligned}
\Omega(@, \tilde{P}, 1, \mathsf{tt}) &= \Omega(@, \mathsf{open}\, n.\tilde{P}', 1, \mathsf{tt}) \uplus \Omega(@, n[\tilde{R}], 1, \mathsf{tt}) \\
&= \varepsilon(@) \uplus \Omega(@, \tilde{P}', 1, \mathsf{ff}) \uplus \Omega(n_i, \tilde{R}, 1, \mathsf{tt}) \uplus \varepsilon(n_i) \uplus (e^{\mathsf{c}}_{n_i}, 1, \rho(\tilde{R})) \prec (@^{\mathsf{ho}}, 1, \mathbf{0})
\end{aligned}$$

For the process $\tilde{Q}'$ we have:

$$\Omega(@, \tilde{P}'|\tilde{R}, 1, \mathsf{tt}) = \Omega(@, \tilde{P}', 1, \mathsf{tt}) \uplus \Omega(@, \tilde{R}, 1, \mathsf{tt})$$

Let $\mathcal{D}(\tilde{P}) = (\mathsf{cf}(\gamma_{\Omega(@,\tilde{P},1,\mathsf{tt})}), \mathsf{P}_{\Omega(@,\tilde{P},1,\mathsf{tt})})$. As observed for the case (Red In), because of the canonical form, we can separate the part of the configuration $\gamma_{\mathcal{D}(\tilde{P})}$ related to the execution

of the rule from the rest of the configuration. Therefore, the first step of the application of OpenUp result in:

$$\begin{aligned}
&\mathrm{OpenUp}(\mathcal{D}(\tilde{P}), @^{\mathsf{is}}, e^{\mathsf{c}}_{n_i}, \tilde{P}') \\
&= \langle act_{@,\tilde{P}'}(diss(\gamma_{\mathcal{D}(\tilde{P})}, @^{\mathsf{ho}}, e^{\mathsf{c}}_{n_i})), \\
&\quad \mathsf{P}_{\mathcal{D}(\tilde{P})}[@^{\mathsf{is}} \mapsto \mathsf{P}_{\mathcal{D}(\tilde{P})}(@^{\mathsf{is}}) \backslash \{\mathsf{open}\, n.\tilde{P}'\} \cup \rho(\tilde{P}') \cup \mathsf{P}_{\mathcal{D}(\tilde{P})}(e^{\mathsf{c}}_{n_i})]\rangle \\
&= \langle act_{@,\tilde{P}'}(diss(\gamma_{\mathcal{D}(\tilde{P})}, @^{\mathsf{ho}}, e^{\mathsf{c}}_{n_i})), \mathsf{P}_{\mathcal{D}(\tilde{P})}[@^{\mathsf{is}} \mapsto \rho(\tilde{P}'|\tilde{R})]\rangle
\end{aligned}$$

Let's now concentrate only on the configuration part:

$$act_{@,\tilde{P}'}(diss(\gamma_{\mathcal{D}(\tilde{P})}, @^{\mathsf{ho}}, e^{\mathsf{c}}_{n_i}))$$

where

$$\begin{aligned}
\gamma_{\mathcal{D}(\tilde{P})} = \gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{P}',1,\mathsf{ff})} \uplus \gamma_{\Omega(n_i,\tilde{R},1,\mathsf{tt})} \uplus \gamma_{\varepsilon(n_i)} \\
\uplus \langle \{e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}}\}, \{(e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}})\}, \mathbf{1}_{\{e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}}\}} \rangle.
\end{aligned}$$

Moreover, according to the definition of *diss*, after the deletion of $e^{\mathsf{c}}_{n_i}$ that cancel the subset

$$\langle \{e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}}\}, \{(e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}})\}, \mathbf{1}_{\{e^{\mathsf{c}}_{n_i}, @^{\mathsf{ho}}\}} \rangle$$

the pointer structure must be modified according to the following schema:

$$\mu' = \mu_{\gamma_q}[\mu_{\gamma_q}(e^{\mathsf{c}}_{n_i}) \mapsto @^{\mathsf{ho}},\ \mu^{-1}_{\gamma_q}(n^{\mathsf{ho}}_i) \mapsto @^{\mathsf{ho}}]$$

The first kind of update that links other possible copies of n_i to $@^{\mathsf{ho}}$ is implicit in the representation of the configuration we are using. Namely: if there would be other entities linked to $e^{\mathsf{c}}_{n_i}$, since we consider the configuration as the union of the part related to $e^{\mathsf{c}}_{n_i}$ and the rest of the configuration, for the very nature of the union operator, the other copies of n_i are already linked to $@^{\mathsf{ho}}$ (and represented in other part of the union). For the second update, note that the entities in $\mu^{-1}_{\gamma_q}(n^{\mathsf{ho}}_i)$ that are those inside n_i, in $\gamma_{\mathcal{D}(\tilde{P})}$ are contained in the part of the configuration build by $\Omega(n_i, \tilde{R}, 1, \mathsf{tt})$. In fact this is the only part of the configuration that embeds ambients inside n_i. Note furthermore, that these ambients must be precisely the set $enab(\tilde{R})$. Therefore, moving these ambients from n_i into $@$, corresponds to replace the part of the configuration $\Omega(n_i, \tilde{R}, 1, \mathsf{tt})$ by $\Omega(@, \tilde{R}, 1, \mathsf{tt})$. We conclude that after the pointer update dictated by *diss* to the configuration $\gamma_{\mathcal{D}(\tilde{P})}$ we have:

$$act_{@,\tilde{P}'}(\gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{P}',1,\mathsf{ff})} \uplus \gamma_{\Omega(@,\tilde{R},1)} \uplus \gamma_{\varepsilon(n_i)})$$

and by the application of *act* that replace $\Omega(@, \tilde{P}', 1, \mathsf{ff})$ by $\Omega(@, \tilde{P}', 1, \mathsf{tt})$ we obtain:

$$\begin{aligned}
&act_{@,\tilde{P}'}(\gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{P}',1,\mathsf{ff})} \uplus \gamma_{\Omega(@,\tilde{R},1,\mathsf{tt})} \uplus \varepsilon(n_i)) \\
&= \gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{P}',1,\mathsf{tt})} \uplus \gamma_{\Omega(@,\tilde{R},1,\mathsf{tt})} \uplus \gamma_{\varepsilon(n_i)} \\
&= \gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{P}'|\tilde{R},1,\mathsf{tt})} \uplus \gamma_{\varepsilon(n_i)} \\
&= \gamma_{\varepsilon(@)} \uplus \gamma_{\Omega(@,\tilde{Q}',1,\mathsf{tt})} \uplus \gamma_{\varepsilon(n_i)} \\
&= \gamma_{\Omega(@,\tilde{Q}',1,\mathsf{tt})} \uplus \gamma_{\varepsilon(n_i)}.
\end{aligned}$$

Now, according to the **Open** rule of Table 3, the target states are the canonical form of the state in the safe expansion, among which there is h one such that $(h_{\mathsf{cf}} \circ h^{-1}(\gamma_{\Omega(@,\tilde{Q}',1)} \uplus \varepsilon(n_i)), \mathsf{P}_{\Omega(@,\tilde{Q}',1)}) = \mathcal{D}(\tilde{Q}') \uplus \varepsilon(n_i)$ using the same observation we applied for **In**. Moreover, since $n(\tilde{P}) = \{n_i\}$, also in this case we conclude:

$$\mathcal{D}(\tilde{P}) \rightarrow_{\lambda} \mathcal{D}(\tilde{Q}') \uplus \varepsilon(n(\tilde{P})).$$

– case (Red Par). Let $\tilde{P} = \tilde{P}_1|\tilde{P}_2$ and $noi(P) = P_1|P_2$. If $P_1 \to P_1'$ by (Red Par) then $Q = P_1'|P_2$.

$$\begin{aligned}
\mathcal{D}(\tilde{P}) &= \mathsf{cf}(\Omega(@, \tilde{P}_1|P_2, 1, \mathsf{tt})) \\
&= h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1, 1, \mathsf{tt}) \uplus \Omega(@, \tilde{P}_2, 1, \mathsf{tt})) \\
&= [\text{by Lemma 3}] \\
&\quad h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1, 1, \mathsf{tt})) \uplus h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_2, 1, \mathsf{tt})) \\
&= [\text{since } \tilde{P} \text{ well-indexed and by Lemma 3}] \\
&\quad h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1, 1, \mathsf{tt})) \uplus h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_2, 1, \mathsf{tt})) \\
&= [\text{by definition}] \\
&\quad \mathcal{D}(\tilde{P}_1) \uplus \mathcal{D}(\tilde{P}_2)
\end{aligned}$$

By induction hypothesis: $\mathcal{D}(\tilde{P}_1) \to_\lambda \mathcal{D}(\tilde{\P}_1' \uplus \varepsilon(n(\tilde{P}_1)))$. Therefore

$$\mathcal{D}(\tilde{P}_1|\tilde{P}_2) \to_\lambda \mathcal{D}(\tilde{P}_1' \uplus \varepsilon(n(\tilde{P}_1))) \uplus \mathcal{D}(\tilde{P}_2)$$

On the other hand we have:

$$\begin{aligned}
\mathcal{D}(\tilde{P}_1' \uplus \varepsilon(n(\tilde{P}_1))) \uplus \mathcal{D}(\tilde{P}_2) &= \mathcal{D}(\tilde{P}_1') \uplus \varepsilon(n(\tilde{P}_1)) \uplus \mathcal{D}(\tilde{P}_2) \uplus \varepsilon(n(\tilde{P}_2)) \\
&= [\text{by definition}] \\
&\quad h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1', 1, \mathsf{tt})) \uplus h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_2, 1, \mathsf{tt})) \uplus \varepsilon(n(\tilde{P}_1|\tilde{P}_2)) \\
&= [\text{by Lemma 3}] \\
&\quad h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1', 1, \mathsf{tt}) \uplus \Omega(@, \tilde{P}_2, 1, \mathsf{tt})) \uplus \varepsilon(n(\tilde{P})) \\
&= h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1'|P_2, 1, \mathsf{tt})) \uplus \varepsilon(n(\tilde{P})) \\
&= \mathcal{D}(Q \uplus \varepsilon(n(\tilde{P})))
\end{aligned}$$

– case (Red Amb). Let $\tilde{P} = n_i[\tilde{P}_1]$. If $P_1 \to P_2$ then by (Red Amb) $noi(\tilde{P}) \to n[P_2] = Q$. We have that:

$$\mathcal{D}(\tilde{P}) = \varepsilon(@) \uplus \Omega(n_i, \tilde{P}_1, 1, \mathsf{tt}) \uplus A$$

where $A = (e_{n_i}^{\mathsf{c}}, 1, \rho(P_1)) \prec (@^{\mathsf{ho}}, 1, \mathbf{0})$ Therefore we have

$$\begin{aligned}
\mathcal{D}(\tilde{P}) &= [\text{by definition}] \\
&\quad h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}, 1, \mathsf{tt})) \\
&\quad h_{\mathsf{cf}}^{-1}(\varepsilon(@) \uplus \Omega(@, \tilde{P}_1, 1, \mathsf{tt}) \uplus A) \\
&= [\text{by Lemma 3}] \\
&\quad h_{\mathsf{cf}}^{-1}(\varepsilon(@)) \uplus h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_1', 1, \mathsf{tt}) \uplus h_{\mathsf{cf}}^{-1}(A)) \\
&= h_{\mathsf{cf}}^{-1}(\varepsilon(@)) \uplus \mathcal{D}(\tilde{P}_1) \uplus h_{\mathsf{cf}}^{-1}(A)
\end{aligned}$$

By induction hypothesis we have $\mathcal{D}(\tilde{P}_1) \to_\lambda \mathcal{D}(P_2 \uplus \varepsilon(n(P_1)))$ therefore

$$\begin{aligned}
\mathcal{D}(\tilde{P}) \to_\lambda\ & h_{\mathsf{cf}}^{-1}(\varepsilon(@)) \uplus \mathcal{D}(\tilde{P}_2 \uplus \varepsilon(n)(P_1)) \uplus h_{\mathsf{cf}}^{-1}(A) \\
=\ & h_{\mathsf{cf}}^{-1}(\varepsilon(@)) \uplus h_{\mathsf{cf}}^{-1}(\Omega(@, \tilde{P}_2, 1, \mathsf{tt}) \uplus \varepsilon(n(P_1))) \uplus \varepsilon(n(P_1))) \uplus h_{\mathsf{cf}}^{-1}(A) \\
=\ & \mathcal{D}(Q) \uplus \varepsilon(n(\tilde{P}))
\end{aligned}$$

– case (Red Res). Similar to case (Red Amb).
– case (Red $\equiv$). If $noi(P) \to Q$ by (Red $\equiv$) then there exists P'', Q'' such that $P \equiv P''$, $Q \equiv Q''$ and $P'' \to Q''$. Therefore, by Lemma 5 we have $\mathcal{D}(P) \equiv \mathcal{D}(P'')$ and $\mathcal{D}(Q) \equiv \mathcal{D}(Q'')$. By induction hypothesis we have

$$\mathcal{D}(P'') \to_{\lambda_D} (Q''') \tag{11}$$

such that $noi(Q''') \equiv Q''$. Since $\mathcal{D}(P) \cong \mathcal{D}(P'')$, then by (11) and by Lemma 7 $\mathcal{D}(P) \rightarrow_{\lambda_1^*} q$ such that $\mathcal{D}(Q''') \cong q$. By Lemma 6 there exists R such that $\mathcal{D}(R) = q$ and $noi(Q''') = noi(R)$. This in turn implies that $noi(R) \equiv noi(Q'')$ and $noi(R) \equiv (Q)$. We conclude by taking $R = \tilde{Q}'$. □