Reachability Problems in BioAmbients

Giorgio Delzanno[1] Gianluigi Zavattaro[2]

[1] *Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova*
[2] *Dip. di Scienze dell'Informazione, Università di Bologna - INRIA FOCUS Team*

Abstract

BioAmbients (BA) is a powerful model for representing various aspects of living cells. The model provides a rich set of operations for the movement and interaction of molecules. The richness of the language motivates the study of fragments of the full model and the comparison with other computational models. In this paper, we investigate the impact of the *merge* capability, used for fusing the contents of two sibling ambients, on the decidability of two reachability problems called Target and Spatial reachability. By enhancing techniques —based on the theory of Petri nets— already used in the context of Mobile Ambients, we prove that both Target and Spatial reachability are decidable for a Turing-complete fragment of BA without *merge*. Then we extend this fragment with a limited form of *merge*, that does not reduce the total number of ambients: in this fragment Target reachability is no longer decidable, but by resorting to the theory of Petri nets with transfer arcs we prove that at least Spatial reachability is still decidable. Finally, we show that if we consider the standard *merge* capability then both reachability problems turn out to be undecidable. Besides characterizing the power of *merge*, the proof techniques that we use also establish an interesting connection between BA and other computational models like standard Petri nets, their extension with transfer arcs, and Two Counter Machines (a Turing complete formalism based on registers).

1 Introduction

BioAmbients [15] (BA) is a well known formalism for the description of biological systems that combines the communication mechanisms of the π-calculus [14] with the notion of ambient as formalized in Mobile Ambients [7]. This combination allows for the representation of biochemical reactions by means of process communication and to model biological compartments by means of ambients. A bioambient $[P]$ is a collection of active processes and nested sub-bioambients represented by the term P. Active processes can perform communication actions with other processes or execute capabilities in

order to modify the ambient nesting. Communication consists of the interaction between an output and an input action performed by processes located in the same ambient, in parent/child ambients, or in two parallel ambients. The capabilities allow processes to modify ambient nesting in three possible ways: one ambient can move inside a parallel ambient, one ambient can move outside from the parent ambient, or two parallel ambients can merge into one single ambient.

In this paper we discuss reachability problems in BA by exploiting and enhancing techniques developed by Busi and Zavattaro for Mobile and Boxed Ambients [4–6]. Classical reachability analysis consists in verifying, given a source process P and a target process Q, whether there exists a computation that starts from P and leads to Q. For ambient-based calculi, extensions of the reachability problem have been considered due to the presence of the ambient nesting structure. In [5,6] the Target reachability problem has been defined: this allows for the specification of a possibly infinite class of targets specified by showing their ambient nesting structure, and indicating lower and upper bounds to the number of specific instances of processes hosted in each ambient. A simpler version in which only lower bounds can be specified, called Spatial reachability, was instead considered in [4].

As an example of the kind of analysis allowed by Target and Spatial reachability problems that we propose, let us consider a generic modeling of the cell represented as an outer membrane that contains the intermediate cytoplasm and the inner nucleus. We could then consider two different external materials, one to which the membrane should be permeable, and another one to which it should not be.

Such a system could be modeled in BA^- in the following way

$$[Material1] \mid [Material2] \mid [Membrane \mid [Cytoplasm \mid [Nucleus]]]$$

where the first two ambients are used to represent and describe the behavior of two different kinds of external material, and another ambient represents the cell with its outer membrane, the intermediate cytoplasm, and the inner nucleus.

We can formalize the expected behavior of the system in terms of reachability problems. For instance, to formalize the fact that the membrane should be permeable to the first kind of material we can state that we expect that the configuration

$$[Material2] \mid [Membrane' \mid [[Material1'] \mid Cytoplasm' \mid [Nucleus]]]$$

is reachable, where $Membrane'$, $Material1'$ and $Cytoplasm'$ respectively de-

scribe the expected state of the membrane, of the transported material, and of the cytoplasm after the membrane has been traversed. This kind of reachability property, i.e. reachability of a given process, is expressable in terms of Target reachability by imposing the lower bounds for the processes to be present equal to their upper bounds.

On the contrary, to formalize that the membrane should not be permeable to the second kind of material, we can state that we expect that the configuration

$$[Material1] \mid [Membrane'' \mid [[Material2''] \mid Cytoplasm'' \mid [Nucleus]]]$$

is not reachable for every process $Membrane''$, $Material2''$ and $Cytoplasm''$. This different kind of reachability property is already expressible in terms of Spatial reachability by indicating only the expected ambient nesting structure, and imposing no constraints on the contained processes.

Reachability is usually undecidable in Turing complete formalisms such as the π-calculus or Mobile Ambients (the ancestors of BA). Nevertheless, at least for Mobile Ambients, very interesting fragments have been studied which are expressive enough to model all computable functions, but for which Target reachability turns out to be decidable. This holds for the fragment without restriction, without the *open* capability (used to dissolve an ambient boundary), and in which replication cannot be applied to ambients. This fragment was first proved to be Turing complete by Maffeis and Phillips [13], and then Spatial and Target reachability were proved to be decidable for such fragment by Busi and Zavattaro in [4–6]. These results follow from a monotonicity property of the considered calculus: because of the absence of the *open* capability, the number of "active" ambients cannot decrease during the computation.

In this paper we apply and extend these results and techniques to BA. We start by considering a fragment of BA similar to the one considered in [13,4]. By resorting to the results in [13] we first show that this fragment is Turing complete, then we show that Target reachability is decidable by adapting to this new context the techniques in [6] based on the theory of Petri nets. Then we consider an extension of this fragment that includes a limited version of the *merge* capability: every time two ambients are merged, then at least a new ambient is created. In this way, the "monotonicity" property about the number of active ambients is preserved. Interestingly, we prove that even if monotonicity is preserved Target reachability is no longer decidable, while the simpler Spatial reachability problem is still decidable. This result is proved by resorting to the theory of Petri nets with transfer arcs. Finally, we show that if we loose the monotonicity property by admitting a *merge* mechanism that can also decrease the total number of active ambients, then also Spatial reachability becomes undecidable.

Structure of the paper

In Section 2 we report the syntax and semantics of BA$^-$, the fragment of BioAmbients that we obtain by removing the choice operator, the communication primitives, the restriction operator, and by imposing that replication is applied only to processes and not to ambients. The elimination of the choice operator and of the communication primitives is done only for simplifying the presentation: the proof of the decidability results can be extended to deal also with these operators as discussed in [17]. On the contrary, the restriction operator and the possibility to apply replication to ambients are eliminated otherwise both Target and Spatial reachability would be already undecidable. [1] In Section 3 we prove that Target reachability is decidable if we remove from BA$^-$ the *merge* primitive. In Section 5 we consider a monotonic version of *merge* that does not decrease the number of active ambients. We first show that for this fragment Target reachability is no longer decidable, but the simpler Spatial reachability problem is still decidable. In Section 6 we prove that if we consider the standard *merge* capability, that allows for the decrement of the number of active ambients, then also Spatial reachability is no longer decidable. Some concluding remarks are reported in Section 7.

This paper is a joint and extended version of [10,17].

2 BioAmbients without Communication and Restriction

In this section we introduce a fragment of BioAmbients, that we call BA$^-$, obtained by removing the choice operator, the communication primitives, the restriction operator, and by limiting the application of the replication operator to processes (and not to ambients).

Definition 2.1 –**BA$^-$**– *Let Label, ranged over by n, m, p, ..., be a denumerable set of labels. The terms of BA$^-$, ranged over by P, Q, R, ..., are*

[1] The undecidability of reachability in the presence of replication applied to ambients follows from the possibility to encode in BA the fragment of Mobile Ambients for which Boneva and Talbot proved the undecidability of reachability in [3]. Such an encoding is described at the beginning of Section 3.

defined by the following grammar:

M, N	$::=$		**Capabilities**
		$enter\, n$	*Synch entry*
		$accept\, n$	*Synch accept*
		$exit\, n$	*Synch exit*
		$expel\, n$	*Synch expel*
		$merge_+\, n$	*Synch merge with*
		$merge_-\, n$	*Synch merge into*
P, Q	$::=$		**Processes**
		$\mathbf{0}$	*Null process*
		$P \vert Q$	*Composition*
		$[P]$	*Ambient (membrane)*
		$M.P$	*Guarded process*
		$!M.P$	*Replication*

In the following we use $\prod_k P$ to denote the parallel composition of k instances of the process P, while $\prod_i P_i$ denotes the parallel composition of the indexed processes P_i. As usual, we frequently omit trailing $\mathbf{0}$.

Processes run inside ambients and perform capabilities to modify the ambient structure. More precisely, capabilities allow a process to move its hosting ambient outside (resp. inside) an outer (resp. a sibling) ambient. Namely, *exit* and *expel* are used for outside movement, while *enter* and *accept* are for inside movement. Moreover, the complementary $merge_+$ and $merge_-$ capabilities allow two sibling ambients to merge their processes into a unique ambient.

Infinite behaviours in BioAmbients are modeled using the replication operator. In BA^- we do not admit the application of replication to ambients, e.g., $![P]$ is not a valid term.

The operational semantics is defined in terms of a structural congruence plus a reduction relation.

Definition 2.2 –**Structural congruence**– *The structural congruence* $\equiv$ *is the smallest congruence relation satisfying the following:*

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

Definition 2.3 –**Reduction relation**– *The reduction relation is the smallest relation* $\rightarrow$ *satisfying the following axioms and rules:*

$$[(enter\, n.P) \mid Q] \mid [(accept\, n.R) \mid S] \rightarrow [[P \mid Q] \mid R \mid S]$$
$$[[(exit\, n.P) \mid Q] \mid (expel\, n.R) \mid S] \rightarrow [P \mid Q] \mid [R \mid S]$$
$$[(merge_+\, n.P) \mid Q] \mid [(merge_-\, n.R) \mid S] \rightarrow [P \mid Q \mid R \mid S]$$
$$P \rightarrow Q \;\Rightarrow\; [P] \rightarrow [Q]$$
$$P \rightarrow Q \;\Rightarrow\; P|R \rightarrow Q|R$$
$$P \equiv P', P \rightarrow Q, Q \equiv Q' \;\Rightarrow\; P' \rightarrow Q'$$

The first three reduction rules handle ambient operations while the remaining rules handle reductions in context and up to structural congruence.

In the following, we use $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$. If $P \rightarrow^* Q$ we say that Q is a *derivative* of P.

We now define the two fragments of BA^- that we consider in the remainder of the paper.

Definition 2.4 –$\mathbf{BA}^-_{nomrg}$ **and** $\mathbf{BA}^-_{mntn}$– *With* BA^-_{nomrg} *we denote the fragment of* BA^- *obtained by removing the prefixes* $merge_+$ *and* $merge_-$*. With* BA^-_{mntn} *we denote the fragment of* BA^- *obtained by assuming that every occurrence of* $merge_+$ *is of the following form:* $merge_+\, n.([Q]|R)$ *for some label* $n \in Label$ *and some processes* Q *and* R*.*

Notice that in both the two above fragments the number of ambients never decreases when a reduction step is executed (the merging of two ambients in BA^-_{mntn} is compensated by the creation of at least one new ambient).

Example 2.5 *As an example, inspired by the cell membrane example in the*

Introduction, consider the following process evolution

$$[enter\, m.enter\, c.Material] \mid [!accept\, m \mid [!accept\, c \mid [Nucleus]]] \quad \rightarrow$$
$$[!accept\, m \mid [enter\, c.Material] \mid [!accept\, c \mid [Nucleus]]] \rightarrow$$
$$[!accept\, m \mid [!accept\, c \mid [Material]| \, [Nucleus]]]$$

The first ambient represents material having the ability to traverse the membrane in two reduction steps. The cell is represented as the nesting of three ambients: an outer ambient representing the membrane, an intermediate ambient representing the cytoplasm, and an inner ambient representing the nucleus. The sequence of capabilities enter m.enter c gives the possibility to the material to traverse the membrane and enter the cytoplasm. In fact, in these last two ambients reside the processes !accept m and !accept c willing to accept all those ambients performing the complementary capabilities. The replication operator is used to represent the fact that the membrane is constantly permeable.

Assume now that the processes M and N, respectively representing the behavior of the material and of the nucleus are defined as follows:

$$Material \;=\; merge_{-}\, n$$
$$Nucleus \;=\; merge_{+}\, n.[DamagedNucleus]$$

In this case, a third reduction step could occur:

$$[!accept\, m \mid [!accept\, c \mid [merge_{-}\, n]| \, [merge_{+}\, n.[DamagedNucleus]]]] \quad \rightarrow$$
$$[!accept\, m \mid [!accept\, c \mid [DamagedNucleus]]]$$

This last step replaces the nucleus with a damaged one, thus representing the effect of dangerous material that attacks the cell by entering its membrane. Notice that in this last example the merge capability is used, but the number of ambients does not decrease. In fact, the process performing the $merge_{+}$ capability satisfies the constraint imposed by the fragment BA^{-}_{mntn}.

2.1 Target and Spatial Reachability

Classical reachability analysis consists in checking if $P \rightarrow^{*} R$ for two given processes P and R. We consider a more general notion of reachability allowing for a partial description of the target process. More precisely, it is possible to

impose constraints on the number of occurrences of guarded processes inside an ambient. Such constraints are both lower bounds (e.g. there must be at least one instance of the guarded process $M.Q$ in a given ambient) and upper bounds (e.g. there can be at most two occurrences of the guarded process $M.Q$ in a given ambient).

We need to introduce some additional notation to denote the partial description of target processes.

We first introduce a notion of normal form for processes that forbids the presence of both the unreplicated and the replicated version of a guarded term in a parallel composition. Any process can be transformed in a structurally congruent process in normal form by using the monoidal axioms for parallel composition and by applying the axiom for replication from right to left (i.e., $M.P \mid !M.P$ is transformed in $!M.P$). In the following we use G, G_i, L, L_i, ... to denote guarded processes, i.e. terms of the form $M.P$.

Definition 2.6 –**Normal form**– *A process P is in normal form if $P = \prod_i G_i \mid \prod_j !G'_j \mid \prod_k [P_k]$ and the following conditions hold:*

- *Q is in normal form for all Q such that $G_i = M.Q$ or $G'_j = M.Q$ for some i, j, M;*
- *P_k is in normal form for all k;*
- *there exist no i, j such that $G_i = G'_j$.*

Proposition 2.7 *Let P be a process. Then there exists a process Q in normal form such that $P \equiv Q$.*

Definition 2.8 –**Target**– *The set of* targets *is defined by the following grammar:*

$$T \ ::= \ \texttt{any} \ \mid \ q \leq G \leq q' \ \mid \ !G \ \mid \ T|T \ \mid \ [T]$$

where $q \in I\!N$ and $q' \in I\!N \cup \{\infty\}$.

We use $I\!N$ to denotes the set of natural numbers and we assume that $q \leq \infty$ for all $q \in I\!N$.

A target `any` requires the presence of zero or more occurrences of any process, while $q \leq G \leq q'$ requires the presence of k occurrences of the guarded process G, with $q \leq k \leq q'$ (if $q' = \infty$ there is no upper bound to the number of occurrences). A target $!G$ requires the presence of one or more occurrences of process $!G$. As the behavior of processes $\prod_k !G$ is the same for any $k \geq 1$, we prefer to require just the presence – or the absence – of a replicated process instead of providing upper and lower bounds to the number of its occurrences.

Targets can be composed in parallel, and can be nested in ambients.

As an example, consider the target

$$[1 \leq \mathit{expel}\, n.P \leq 2 \mid [!G] \mid [\texttt{any} \mid 3 \leq \mathit{exit}\, n.Q \leq \infty]]$$

This target requires that an outer ambient contains one or two occurrences of process $\mathit{expel}\, n.P$, an ambient containing only occurrences of process $!G$ (at least one occurrence is required), and an ambient containing at least three occurrences of the process $\mathit{exit}\, n.Q$ and any other process. Moreover, this target also requires that there is no process at top level.

We consider only a proper subset of *well formed* targets defined as follows.

Basically, a target is well formed if the upper and lower bounds on guarded terms are satisfiable (i.e., target $3 \leq M.P \leq 2$ is not well formed) and if the presence of a replicated version of a guarded process prevents the occurrence of the nonreplicated version of the same process in a parallel composition (i.e., target $G \mid !G$ is not well formed). We also require that at most one occurrence of a replicated process is present in a parallel composition (i.e., target $!G \mid !G$ is not well formed).

Definition 2.9 –Well formed target– *A* target T *is well formed if there exists a target* $S = \prod_i q_i \leq G_i \leq q'_i \mid \prod_j !G'_j \mid \prod_k [T'_k]$ *such that the following conditions hold:*

- *either* $T \equiv S$ *or* $T \equiv S \mid \texttt{any}$*;*
- *processes* G_i, G'_j *are of the form* $M.Q$ *(guarded processes) for all* i, j*;*
- $q_i \leq q'_i$ *for all* i*;*
- *there exist no* i*,* j *such that* $G_i = G'_j$*;*
- *if* $G'_j = G'_{j'}$ *then* $j = j'$*;*
- T'_k *is well formed for all* k*.*

We define the set of processes $\mathit{set}(T)$ that satisfy the constraints imposed by a target T. Basically, we require the presence of the required number of occurrences of a guarded process in each ambient; if the upper bound is ∞, then also the presence of a replicated version of the process satisfies the target (i.e., process $[!G]$ satisfies the target $[3 \leq G \leq \infty]$). If the target `any` is present, then further (different) processes may be present. As already discussed, with a replicated process in the target we just require the presence of at least one occurrence of such a replicated process.

Definition 2.10 –set(T)– *Let* T *be a well formed target. A process* P *is in* $\mathit{set}(T)$ *if* $P \equiv \prod_h L_h \mid \prod_g !L'_g \mid \prod_k [P'_k]$ *and there exists a target* $S = \prod_i q_i \leq G_i \leq q'_i \mid \prod_j !G'_j \mid \prod_k [T'_k]$ *such that the following conditions hold:*

- *either* $T \equiv S$ *or* $T \equiv S \mid \texttt{any}$*;*

- *for all i, either $q_i \leq |\{h \mid L_h = G_i\}| \leq q'_i$ or $q'_i = \infty$ and there exists g such that $L'_g = G_i$;*
- *for all j there exists g such that $L'_g = G'_j$;*
- *if $T \equiv S$ then for any h there exists i such that either $L_h = G_i$ or $L_h = G'_i$ and for any g there exists j such that $L'_g = G'_j$;*
- *for any k, $P'_k \in set(T'_k)$.*

It is worth to note that $set(T)$ is compatible with the structural congruence relation as formalized by the following Proposition.

Proposition 2.11 *Let T be a well formed target and P and Q two processes such that $P \equiv Q$. Then, $P \in set(T)$ if and only if $Q \in set(T)$.*

We are now ready to formalize the notion of *Target reachability*.

Definition 2.12 **–Target reachability–** *Let P be a process and T be a well formed target. We say that T is a target reachable from P (denoted by $TReach(P, T)$) if there exists a process Q such that $P \rightarrow^* Q$ and $Q \in set(T)$.*

In the paper we will consider also a weaker version of reachability analysis in which besides a required ambient nesting structure, only a minimal amount of available processes can be expressed without imposing upper bounds to their occurrences and without limitations on the presence of other (different) processes. Namely, we assume that targets contain only lower bounds, and `any` is present in every ambient. We call Spatial reachability this kind of analysis.

Definition 2.13 **–Spatial reachability–** *Spatial reachability corresponds to the Target reachability problem on targets, that we call* spatial *targets, defined according to the following grammar:*

$$\begin{aligned} S &::= \texttt{any} \mid S' \\ S' &::= q \leq G \leq \infty \mid !G \mid S' \mid S' \mid [S] \end{aligned}$$

Notice that spatial targets are restricted forms of targets in which `any` *occurs in all ambients, and the unique possible upper bound is ∞. Given a process P and a spatial target S, with $SReach(P, S)$ we mean $TReach(P, S)$.*

Example 2.14 *We continue with our running example about material that traverses a cell membrane. In the introduction we have introduced the process*

$$P = [Material1] \mid [Material2] \mid [Membrane \mid [Cytoplasm \mid [Nucleus]]]$$

Suppose that $Material1$, $Membrane$, and $Cytoplasm$ are defined as follows:

$$\begin{aligned} Material1 &= enter\ m.merge_{+}\ c.[enter\ nucleusFood] \\ Material2 &= !accept\ m \\ Cytoplasm &= !merge_{-}\ c \end{aligned}$$

Consider now the target:

$$T = [\mathtt{any}] \mid [!accept\ m \mid [!merge_{-}\ c \mid [1 \leq enter\ nucleusFood \leq 10] \mid [\mathtt{any}]]]$$

It is easy to see that $TReach(P, T)$ holds independently of how $Material2$ and $Nucleus$ are defined as in the corresponding ambients we have introduced any*. The target T is not a spatial target as there are ambients that do not include* any*, and an upper bound is specified for the enter $nucleusFood$ process. We could modify T in order to obtain a spatial target as follows:*

$$S = [\mathtt{any}] \mid [\mathtt{any} \mid !accept\ m \mid [\ \mathtt{any} \mid !merge_{-}\ c \mid [1 \leq enter\ nucleusFood \leq \infty] \mid [\mathtt{any}]]]$$

Clearly, this last target imposes strictly less constraints as in every ambient the presence of any kind of process is admitted. Moreover, as far as the constraint on the process enter $nucleusFood$ is concerned, this is satisfied by any number strictly greater than 1 of occurrences of the process or by its replicated version $!enter\ nucleusFood$. As $TReach(P, T)$ holds, we have that also $SReach(P, S)$ trivially holds.

3 Deciding Target Reachability in BA^{-}_{nomrg}

In this section we prove the decidability of Target reachability for the fragment BA^{-}_{nomrg}.

We first observe that this fragment, even if it comprises only the *enter/accept* and *exit/expel* capabilities, is already Turing complete. This follows directly from a result for Mobile Ambients due to Maffeis and Phillips [13]. They have shown that computable functions can be encoded into a fragment of Mobile Ambients without restriction, including only the *in* and *out* primitives, and in which replication can be applied to processes only. The difference between this fragment of Mobile Ambients and BA^{-}_{nomrg} is that, in Mobile Ambients, ambients have a name which is used by the *in* and *out* capabilities to indicate the target of the corresponding movement. Namely, in Mobile Ambients we

have the following reduction rules for the *in* and *out* capabilities:

$$n[in\, m.P \mid Q] \mid m[R] \;\rightarrow\; m[n[P \mid Q] \mid R]$$
$$m[n[out\, m.P \mid Q] \mid R] \;\rightarrow\; n[P \mid Q] \mid m[R]$$

This form of movement can be easily encoded in BA^{-}_{nomrg}. In order to model an ambient with name n willing to accept sibling ambients performing $in\, n$ or permitting internal ambients performing $out\, n$ to exit, we can use a BA^{-}_{nomrg} ambient that contains the two processes $!expel\, n$ and $!accept\, n$. Following this approach, the $in\, n$ and $out\, n$ capabilities of Mobile Ambients are directly mapped to the $enter\, n$ and $exit\, n$ capabilities of BA^{-}_{nomrg}. This simple encoding of the Mobile Ambients fragment considered in [13] into BA^{-}_{nomrg}, allows us to conclude that the latter is already Turing complete.

We now move to the proof of decidability of Target reachability for BA^{-}_{nomrg}. As anticipated in the Introduction, the proof is basically an adaptation of the proof of decidability of Target reachability for a fragment of Mobile Ambients presented in [6]. The main difference is due to the different kind of mobility based on the pairs *enter*/*accept* and *exit*/*expel* instead of the *in* and *out* capabilities.

The proof is by reduction to a similar problem for Petri nets.

3.1 P/T Nets

We recall Place/Transition nets with unweighted flow arcs (see, e.g., [16]). Here we provide a characterization of this model which is convenient for our aims.

Definition 3.1 *Given a set S, a* finite multiset *over S is defined as a function $m : S \to \mathbb{N}$ such that the set $dom(m) = \{s \in S \mid m(s) \neq 0\}$ is finite. The* multiplicity *of an element s in m is given by the natural number $m(s)$. The set of all finite multisets over S, denoted by $\mathcal{M}_{fin}(S)$, is ranged over by m. A multiset m such that $dom(m) = \emptyset$ is called* empty*. The set of all finite sets over S is denoted by $\wp_{fin}(S)$.*

Given the multisets m and m', we write $m \subseteq m'$ if $m(s) \leq m'(s)$ for all $s \in S$ while $\oplus$ denotes their multiset union*: $m \oplus m'(s) = m(s) + m'(s)$. The operator $\setminus$ denotes* multiset difference*: $(m \setminus m')(s) =$ if $m(s) \geq m'(s)$ then $m(s) - m'(s)$ else 0. The* scalar product*, $j \cdot m$, of a number j with m is $(j \cdot m)(s) = j \cdot (m(s))$.*

To lighten the notation, we sometimes use the following abbreviation. If m is a multiset containing only one occurrence of an element s (i.e., $dom(m) = \{s\}$

and $m(s) = 1$) we denote m by only s. Multiset union is represented also by comma, i.e., $m, m' = m \oplus m'$. Let m be a multiset over S and m' a multiset over $S' \supseteq S$, such that $(m'(s') = 0)$ for each $s' \in S' \setminus S$; with abuse of notation, we sometimes use m in place of m', and vice versa.

Definition 3.2 –P/T nets– *A P/T net is a pair (S, T) where S is a finite set of* places *and $T \subseteq \mathcal{M}_{fin}(S) \times \mathcal{M}_{fin}(S)$ is a finite set of* transitions*. Finite multisets over the set S of places are called* markings*. Given a marking m and a place s, we say that s contains $m(s)$* tokens*. A P/T net is finite if both S and T are finite. A P/T system is a triple $N = (S, T, m_0)$ where (S, T) is a P/T net and m_0 is the* initial marking*.*

A transition $t = (c, p)$ is usually written in the form $c \to p$. The marking c, usually denoted by ${}^\bullet t$, is called the preset *of t and represents the tokens to be* consumed*; the marking p, usually denoted by $t^\bullet$, is called the* postset *of t and represents the tokens to be* produced*. A transition t is* enabled *at m if ${}^\bullet t \subseteq m$. The execution of a transition t enabled at m produces the marking $m' = (m \setminus {}^\bullet t) \oplus t^\bullet$. This is written as $m \xrightarrow{t} m'$ or simply $m \to m'$ when the transition t is not relevant. We use σ, τ to range over sequences of transitions; the empty sequence is denoted by ε; let $\sigma = t_1, \ldots, t_n$, we write $m \xrightarrow{\sigma} m'$ to mean the* firing sequence $m \xrightarrow{t_1} \cdots \xrightarrow{t_n} m'$*. We say that m' is* reachable from *m if there exists σ such that $m \xrightarrow{\sigma} m'$. We say that m' is* coverable from *m if $m \xrightarrow{t_1} \cdots \xrightarrow{t_n} m''$ with $m' \subseteq m''$.*

Definition 3.3 – target marking – *Let $N = (S, T)$ be a P/T net. A target marking of N is a pair of functions $(inf, sup) \in (S \to I\!N) \times (S \to I\!N \cup \infty)$ such that, for all $s \in S$, $inf(s) \leq sup(s)$.*

Definition 3.4 – target marking satisfiablity – *Let $N = (S, T)$ be a P/T net. A marking m of N satisfies a target marking (inf, sup) of N if, for all $s \in S$, $inf(s) \leq m(s) \leq sup(s)$.*

Definition 3.5 – target marking reachability – *Let $N = (S, T, m_0)$ be a P/T system. A target marking (inf, sup) is reachable if there exists a marking m such that $m_0 \to^* m$ and m satisfies (inf, sup).*

In [6] it is proved that Targer marking reachability can be reduced to reachability of a marking taken from a finite set of markings extracted from (inf, sup). The following results then holds.

Theorem 3.6 ([6]) *Target marking reachability is decidable for P/T systems.*

We note that checking reachability of marking m is equivalent to check reachability of the target marking (m, m).

The proof of decidability of Target reachability is done as in [6] by reduction to the reachability problem on Petri nets. The key point of the proof is the definition of a Petri net semantics for the calculus. We report the formal definition of the contruction of the net correponding to one process and one target, while the other parts of the proof are here described informally (they are trivial adaptations of the corresponding parts described in details in [6]).

Given a process P and a target T, we construct a (finite) Petri net that reproduces the computations of the process P that traverses intermediary states in which the number of active ambients is not greater than the number of ambients in T. To check $TReach(P,T)$ is then equivalent to check reachability of a finite set of markings on the Petri net. The intuition behind this approch relies on the monotonicity of BA^{-}_{nomrg}: because of the absence of the *merge* capability, the number of "active" ambients in a process (i.e., ambients that are not guarded by any capability or communication) cannot decrease during the computation. Moreover, as the applicability of replication is restricted to guarded processes, the number of "active" ambients in a set of structurally equivalent processes is finite (while this is not the case in, e.g., the process $![Q]$). Thanks to the property explained above, in order to check Target reachability it is sufficient to take into account a subset of the derivatives of P: namely, those with a number of active ambients which is not greater than the number of active ambients in the target.

Unfortunately, this subset of derivatives is, in general, not finite, as the processes inside an ambient can grow unlimitedly. Consider, e.g., the process

$$[!expel\, n.Q \mid !accept\, m.R \mid [!exit\, n.enter\, m]]$$

It is easy to see that, for every k,

$$[!expel\, n.Q \mid !accept\, m.R \mid [!exit\, n.enter\, m] \mid \prod_k Q \mid \prod_k R]$$

is a derivative of P.

On the other hand, we note that the set of guarded and replicated terms that can occur as subprocesses of (the derivatives of) a process P (namely, the subterms of kind G or $!G$) is finite.

The idea is to borrow a technique used to map process algebras on Petri nets. A process P is decomposed in the (finite) multiset of its guarded and replicated subprocesses that occur unguarded in P; this multiset is then considered

as the marking of a Place/Transition Petri net. The execution of a computational step in a process will correspond to the firing (execution) of a transition in the corresponding net. However, differently from what happens in process algebras, where processes can be faithfully represented by a multiset of subprocesses, BA^{-}_{nomrg} processes have a tree-like structure that hardly fits in a flat model such as a multiset.

The solution is to consider BA^{-}_{nomrg} processes as composed of two kinds of components; the tree-like structure of ambients and the family of multisets of guarded and replicated subterms contained at top level in each ambient. As an example, consider the process

$$!accept\, n.P \mid [enter\, k.Q \mid G] \mid [accept\, k] \mid [\mathbf{0} \mid [exit\, m]]$$

having the tree-like structure [] | [] | [[]]. Moreover, there is a multiset corresponding to each "node" of the tree: the multiset $\{!accept\, n.P\}$ is associated to the root, $\{enter\, k.Q, G\}$ is associated to the first son of the root, $\{accept\, k\}$ is associated to the second son of the root, the empty multiset $\{\}$ is associated to the third son of the root, and $\{exit\, m\}$ to the son of the third son of the root.

The Petri net we construct is composed of the following two parts: the first part is basically a finite state automaton, where the marked place represents the current tree-like structure of the process. The second part is a set of identical subnets: the marking of each subnet represents the multiset associated to a particular node of the tree. To keep the correspondence between the nodes of the tree and the multiset associated to that node, we make use of labels. A distinct label is associated to each subnet; this label will be used in the tree-like structure to label the node whose contents (i.e., the set of guarded and replicated subprocesses contained in the ambient corresponding to the node) is represented by the subnet.

The set of possible tree-like structures we need to consider is finite, because to verify Target reachability we need to take into account only those processes whose number of active ambients is limited by the number of active ambients in the target. The upper bound on the number of nodes in the tree-like structures also provides an upper bound to the number of identical subnets (at most one for each active ambient). In general, the number of active ambients grows during the computation; hence, we need a mechanism to remember which subnets are currently in use and which ones are not used. When a new ambient is created, a correspondence between the node representing such a new ambient in the tree-like structure and a not yet used subnet is established, and the places of the "fresh" subnet are filled with the marking corresponding to the guarded and replicated subprocesses contained in the newly created ambient.

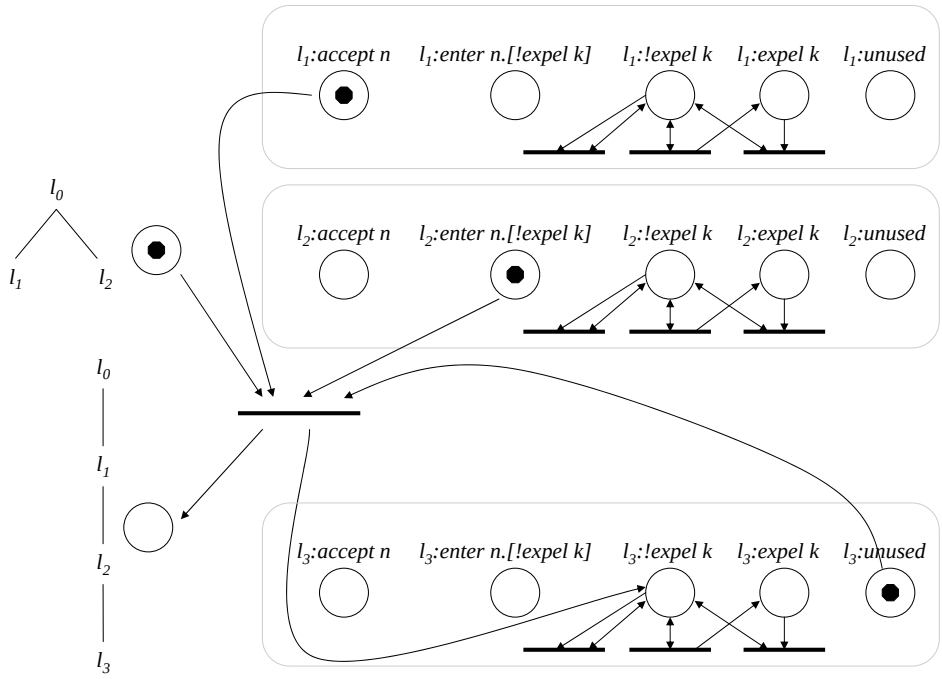

Fig. 1. A portion of the net corresponding to process $[accept\, n] \mid [enter\, n.[!expel\, k]]$.

To this aim, each subnet is equipped with a place called *unused*, that contains a token as long as the subnet does not correspond to any node in the tree-like structure.

For example, consider the process $P = [accept\, n] \mid [enter\, n.[!expel\, k]]$. The relevant part of the net is depicted in Figure 1: a subset of the places, representing the tree-like structure, is depicted in the left-hand part of the figure, while the subnets are depicted in the right-hand part. We only report the subnets labelled with l_1, l_2, and l_3, and omit the subnet labelled with l_0 with empty marking. The computation step $[accept\, n] \mid [enter\, n.[!expel\, k]] \rightarrow [[[!expel\, k]]]$ corresponds to the firing of the transition enabled in the depicted net. Notice that the ambient structure of P has three nodes labeled l_0, l_1, l_2. The label l_3 is introduced only in the place used to simulate the derivative of P obtained by executing $enter\, n$. This action creates a new ambient, labelled l_3, inside the ambient with label l_2. Since we assume here that nodes are never deleted, we just need to mark unused labels (using places like $l_3 : unused$) in order to introduce them only once in every derivation (no transition can add tokens to place $l : unused$). Furthermore, notice that the transitions associated to place $!expel\, k$ are needed to model the semantics of replication (generate and absorb tokens of type $expel\, k$) and to maintain only a copy of process $!expel\, k$ (well-formed terms).

Now we are ready to formally define the net to be used to decide reachability of a target T starting from a process P.

In order to define the places of the net, we need the notion of *ambient multisets* and the function α that maps a process to its representation as ambient multiset. The function α behaves as an homomorphism for all process operations but parallel composition, where some care has to be taken to avoid the presence of both the replicated and the unreplicated versions of a guarded process.

Definition 3.7 *An* index set *is a set $I \subseteq \mathbb{N}$ such that $I = \{1, 2, \ldots, k\}$ for some natural number k. The set $\mathcal{A}$ of* ambient multisets *is the least set closed w.r.t. the following equation:*

$$a = \bigoplus_{i \in I} M_i.a_i \oplus \bigoplus_{j \in J} !M'_j.a'_j \oplus \bigoplus_{k \in K} [a''_k]$$

where I, J, K are index sets, $a_i, a'_j, a''_k \in \mathcal{A}$ and $M_i = M'_j$ implies $a_i \neq a'_j$ for all $i \in I$ and $j \in J$ [2]

The function $\alpha : BA^{-}_{nomrg} \to \mathcal{A}$ maps a process to the corresponding ambient multiset and it is defined inductively as follows:

$$\alpha(\mathbf{0}) = \emptyset \qquad \alpha(M.P) = M.\alpha(P)$$
$$\alpha(!M.P) = !M.\alpha(P) \qquad \alpha([P]) = [\alpha(P)]$$

Now assume that $\alpha(P_h) = \bigoplus_{i \in I_h} M_{ih}.a_{ih} \oplus \bigoplus_{j \in J_h} !M'_{jh}.a'_{jh} \oplus \bigoplus_{k \in K_h} [a''_{kh}]$ for $h = 1, 2$, then we define

$$\alpha(P_1 \mid P_2) = \bigoplus_{h=1,2} \Big(\bigoplus_{i \in I_h} \mu_{ih} \oplus \bigoplus_{j \in J_h} !M'_{jh}.a'_{jh} \oplus \bigoplus_{k \in K_h} [a''_{kh}] \Big)$$

where

$$\mu_{i1} = \begin{cases} M_{i1}.a_{i1} & \text{if } \forall j \in J_2 : M_{i1} = M'_{j2} \;\Rightarrow\; a_{i1} \neq a'_{j2} \\ \emptyset & \text{otherwise} \end{cases}$$

and the μ_{i2} are defined simmetrically.

The tree-like structure of the ambients of a process is represented by an ambient tree, that is basically a tree with nodes decorated with (subnet) labels.

[2] We note that a is the empty multiset when I, J, K empty. This gives the base case of the definition.

We also define the set of ambients trees whose number of ambients is bounded by an upper limit.

Definition 3.8 *Let $\mathcal{L}$ be a denumerable set of labels, i.e., $\mathcal{L} = l_0, l_1, l_2, \ldots$. $\mathcal{L}$ is ranged over by $l, l', l'_1, \ldots$; sequences of labels, i.e., elements of $\mathcal{L}^*$, are ranged over by $\lambda, \lambda', \ldots$. The set $\mathcal{T}$ of* ambient trees *is the least set closed w.r.t. the following equation:*

$$t = l \cdot \bigoplus_{i \in I} [t_i]$$

where I is an index set and $t_i \in \mathcal{T}$ for all $i \in I$. The number of ambients in an ambient tree $t = l \cdot \bigoplus_{i \in I}[t_i]$ is defined as

$$\#amb(t) = |I| + \sum_{i \in I} \#amb(t_i)$$

The set of labels in an ambient tree $t = l \cdot \bigoplus_{i \in I}[t_i]$ is defined as

$$labels(t) = \{l\} \cup \bigcup_{i \in I} labels(t_i)$$

The set of ambient trees of size at most h is defined as follows

$$\mathcal{T}_h = \{t \in \mathcal{T} \mid \#amb(t) \leq h \wedge labels(t) \in \{l_0, \ldots, l_h\}\}$$

In the following we will consider ambient trees containing distinct labels.

To formally define the transitions, we need some auxiliary notations.

Ambient tree contexts will be used to model the requirement that the tree-like structure has a specific form, and to update such structure. An ambient tree context is essentially an ambient tree with a hole, that can be fulfilled with a set of trees. The set of ambient tree contexts is generated by the following grammar:

$$\begin{aligned} C[] \;\; &= l \cdot [] \oplus \textstyle\bigoplus_{i \in I}[t_i] \quad \mid \\ &\quad\; l \cdot [l' \cdot C[] \oplus \textstyle\bigoplus_{j \in J}[t'_j]] \oplus \textstyle\bigoplus_{i \in I}[t_i] \end{aligned}$$

where $t_i, t'_j \in \mathcal{T}$ for every i and j.

We introduce some notions relative to the features of ambient multisets.

Definition 3.9 *Assume an ambient multiset a defined as*

$$a = \bigoplus_{i \in I} M_i.a_i \oplus \bigoplus_{j \in J} !M'_j.a'_j \oplus \bigoplus_{k \in K} [a''_k]$$

$$C[[l^m \cdot \mu^m] \oplus [l^n \cdot \mu^n]], \quad l^m : enter\, k.a, \quad l^n : accept\, k.a',$$
$$\bigcup_{l \in \lambda} l : unused, \quad \bigcup_{l' \in \lambda'} l' : unused$$
(enter) $$\downarrow$$
$$C[[l^n \cdot (\mu^n \oplus tree(a', \lambda') \oplus [l^m \cdot (\mu^m \oplus tree(a, \lambda))])]],$$
$$l^m : actproc(a), \quad l^n : actproc(a'), \quad proc(a, \lambda), \quad proc(a', \lambda')$$

$$C[[l^n \cdot (\mu^n \oplus [l^m \cdot \mu^m])]], \quad l^m : exit\, k.a, \quad l^n : expel\, k.a',$$
$$\bigcup_{l \in \lambda} l : unused, \quad \bigcup_{l' \in \lambda'} l' : unused$$
(exit) $$\downarrow$$
$$C[[l^m \cdot (\mu^m \oplus tree(a, \lambda))] \oplus [l^n \cdot (\mu^n \oplus tree(a', \lambda'))]],$$
$$l^m : actproc(a), \quad l^n : actproc(a'), \quad proc(a, \lambda), \quad proc(a', \lambda')$$

$$l : M.a, \quad l :!M.a$$
(fold) $$\downarrow$$
$$l :!M.a$$

$$l :!M.a$$
(unfold) $$\downarrow$$
$$l : M.a, \quad l :!M.a$$

$$l :!M.a \quad l :!M.a$$
$(\texttt{norm})_1$ $$\downarrow$$
$$l :!M.a$$

$$l : \mathbf{0} \quad l : Q$$
$(\texttt{norm})_2$ $$\downarrow$$
$$l : Q$$

Table 1
The transitions schemata. Regarding axioms (enter) and (exit), we assume that λ and λ' are sequences of distinct labels such that $|\lambda| = \#amb(a)$ and $|\lambda'| = \#amb(a')$. In the rule $(\texttt{norm})_2$ the process Q is any process in $sub(P_0)$ where P_0 is the initial process.

The set of sequential and replicated subprocesses of a is defined as follows:

$$\begin{aligned} sub(a) = {} & \{M_i.a_i \mid i \in I\} \cup \\ & \{M'_j.a'_j, !M'_j.a'_j \mid j \in J\} \cup \\ & \textstyle\bigcup_{i \in I} sub(a_i) \cup \bigcup_{j \in j} sub(a'_j) \cup \bigcup_{k \in K} sub(a''_k) \end{aligned}$$

The number of active ambients in a is defined as

$$\#amb(a) = |K| + \sum_{k \in K} \#amb(a''_k)$$

The number of active ambients in a process P is defined as $\#amb(P) = \#amb(\alpha(P))$ and the number of active ambients in a target T, written $\#amb(T)$, is defined similarly.

The following functions are used to construct the new parts of the ambient tree (generated by the active ambients in the continuation) and the marking of the newly activated subnets.

Definition 3.10 *Let*

$$a = \bigoplus_{i \in I} M_i.a_i \oplus \bigoplus_{j \in J} !M'_j.a'_j \oplus \bigoplus_{k \in K} [a''_k]$$

be an ambient multiset.[3] *Take a sequence of labels $\lambda = l'_1 \dots l'_{|K|} \lambda_1 \dots \lambda_{|K|}$ such that $|\lambda_i| = \#amb(a''_i)$ for all $i \in K$. The function $tree(a, \lambda)$ constructs a portion of ambient tree representing the active ambients in a, where nodes are labelled with the elements of λ taken in breadth first, left-to-right order:*

$$tree(a, \lambda) = \bigoplus_{k \in K} [l'_k \cdot tree(a''_k, \lambda_k)]$$

The function $actproc(a)$ gives the portion of a corresponding to the active sequential and (unguarded) replicated subprocesses:

$$actproc(a) = \bigoplus_{i \in I} M_i.a_i \oplus \bigoplus_{j \in J} !M'_j.a'_j$$

For each active ambient in a, the function $proc(a, \lambda)$ constructs the marking for the places of the corresponding subnet:

$$proc(a, \lambda) = \bigoplus_{k \in K} l'_k : actproc(a''_k) \oplus \bigoplus_{k \in K} proc(a''_k, \lambda_k)$$

The set $Trans$ contains all the instances of the transition schemata reported in Table 1: rules `(enter)` and `(exit)` deal with the execution of a capability;

[3] To be precise, at this point we have to fix an order on the elements of the multiset a, i.e., instead of a we must consider the sequence $\overline{a} = M_1.a_1 \dots M_{|I|}.a_{|I|} !M'_1.a'_1 \dots !M_{|J|}.a'_{|J|} \overline{[a''_1]} \dots \overline{[a''_{|K|}]}$. We need to fix the ordering of the elements to obtain the right correspondence between the labels in the ambient tree and the labels of the active nets.

rules (fold) and (unfold) permit to apply the structural congruence law for replication to unguarded processes and rules $(\texttt{norm})_1$ and $(\texttt{norm})_2$ are used to obtain well-formed processes. Notice that places of the form $l : \mathit{unused}$ only occur as precondition. Indeed, we use them to associate fresh labels to ambients that are created after firing a transition. As mentioned before, since nodes are never deleted there are no transitions with places like $l : \mathit{unused}$ as postcondition (i.e. used labels cannot become fresh again). The P/T net is constructed as follows (where the number n is used to represent the maximal number of active ambients to be considered in the P/T net).

Definition 3.11 *Let P be a BA^-_{nomrg} process and let n be a natural number such that $\#amb(P) \leq n$. We define the net $Net(P, n) = (Pl, Tr)$, where*

$$\begin{aligned} Pl &= \textstyle\bigcup_{i=0}^{n}(\{l_i : Q \mid Q \in sub(\alpha(P))\} \cup \{l_i : unused\}) \ \cup \ \mathcal{T}_n \\ Tr &= \{(c, p) \in Trans \mid c, p \subseteq Pl\} \end{aligned}$$

Following the same proof technique presented in [6] it is possible to show that the net semantics corresponds to the semantics of the calculus: the reachable configurations are the same even if the computation steps are different due to the presence in the net of the (fold), (unfold) and $(\texttt{norm})_i$ $i : 1, 2$ rules.

Definition 3.12 *Let P be BA^-_{nomrg} process and let Q be a process such that $P \rightarrow^* Q$. Consider now a natural number n such that $\#amb(Q) \leq n$.*

Let λ be a sequence of distinct labels in $\{l_1, \ldots, l_n\}$ such that $|\lambda| = \#amb(Q)$, and let the set of labels not in λ defined as $C_\lambda = \{l_i \mid i = 1, \ldots, n \wedge l_i \notin \lambda\}$.

The decomposition *of Q w.r.t. λ is defined as*

$$\begin{aligned} dec(Q, l_0\lambda) = \ & l_0 \cdot tree(\alpha(Q), \lambda), \\ & l_0 : actproc(\alpha(Q)), \\ & proc(\alpha(Q), \lambda), \\ & \textstyle\bigcup_{l \in C_\lambda} l : unused \end{aligned}$$

The decomposition of Q turns out to be a marking of $Net(P, n)$, because the following property holds.

Proposition 3.13 *Let P, Q be a BA^-_{nomrg} process. We have that if $P \rightarrow^* Q$ then $sub(\alpha(Q)) \subseteq sub(\alpha(P))$*

The proof is by induction. The following propositions relates the P/T net semantics with the structural congruence $\equiv$, and the reduction relation $\rightarrow$ for

processes.

Proposition 3.14 *Let P be BA^-_{nomrg} process and let Q be a process such that $P \rightarrow^* Q$. Consider now a natural number n such that $\#amb(Q) \leq n$. Given a BA^-_{nomrg} process Q', we have that $Q \equiv Q'$ if and only if there exists a sequence λ of distinct labels in $\{l_0, \ldots, l_n\}$ such that $|\lambda| = \#amb(Q) = \#amb(Q')$ and $dec(Q, \lambda) \stackrel{\sigma}{\rightarrow} dec(Q', \lambda)$ in $Net(P, n)$ where σ is a sequence that includes an arbitrary number (possibly 0) of transitions that are instances of* `(fold)` *or* `(unfold)` *or* $(\texttt{norm})_i$ $i : 1, 2$ *rules.*

Proposition 3.15 *Let P be BA^-_{nomrg} process and let Q be a process such that $P \rightarrow^* Q$. Consider now a natural number n such that $\#amb(Q) \leq n$. Given a BA^-_{nomrg} process Q' such that $\#amb(Q') \leq n$, we have that $Q \rightarrow Q'$ if and only if there exist two sequences λ and λ' of distinct labels in $\{l_0, \ldots, l_n\}$ such that $|\lambda| = \#amb(Q)$, $\lambda' = \#amb(Q')$, and $dec(Q, \lambda) \stackrel{\sigma}{\rightarrow} dec(Q', \lambda')$ in $Net(P, n)$ where σ is a sequence of transitions that includes an arbitrary number (possibly 0) of transitions that are instances of* `(fold)` *or* `(unfold)` *or* $(\texttt{unfold})_i$ *rules $i : 1, 2$ and exactly one transition which is an instance of* `(enter)` *or* `(exit)`*.*

We now defined the set of target markings $TargMark(P, T, \lambda)$ associated to a target process T. Given two target markings (inf_1, sup_1) and (inf_2, sup_2) defined on disjoint sets of places, we use $(inf_1, sup_1) \oplus (inf_2, sup_2)$ to denote the target marking (inf, sup) where inf is the disjoint union of the functions inf_1 and inf_2, and sup is the disjoint union of the functions sup_1 and sup_2.

Definition 3.16 – TargMark(P,T,λ) – *Let P be a BA^-_{nomrg} process and let T be a well formed target reachable from P, i.e., such that $TReach(P, T)$. Let*

$$S = \prod_{i \in I} q_i \leq M_i.P_i \leq q'_i \mid \prod_{j \in J} !M'_j.P'_j \mid \prod_{k \in K} [T''_k]$$

be a target such that either $T \equiv_T S$ or $T \equiv_T S|\texttt{any}$.

Take a sequence of labels $\lambda = l'_0 \lambda_1 \ldots \lambda_{|K|}$ such that $|\lambda_k| = \#amb(T''_k)$ for all $k \in K$. We define $TargMarkSub(P, T, \lambda)$ as the following set of target markings

$$\begin{aligned} TargMarkSub(P, T, \lambda) \; = \{ & (inf_T, sup_T) \oplus \textstyle\bigoplus_{k \in K} (inf_{T''_k}, sup_{T''_k}) \mid \\ & (inf_{T''_k}, sup_{T''_k}) \in TargMarkSub(P, T''_k, \lambda_k) \wedge \\ & (inf_T, sup_T) \text{ satisfes the property below } \} \end{aligned}$$

where (inf_T, sup_T) is a target marking defined on the set of places $\{l'_0 : Q \mid Q \in sub(\alpha(P))\}$ such that:

- *for all $i \in I$, one of the following holds:*
  - *$inf(l'_0 : M_i.P_i) = q_i$ and $sup(l'_0 : M_i.P_i) = q'_i$*

  - $q'_i = \infty$, $inf(l'_0 : M_i.P_i) = 0$, $sup(l'_0 : M_i.P_i) = \infty$, $inf(l'_0 :!M_i.P_i) = 1$, *and* $sup(l'_0 :!M_j.P_j) = \infty$;
- *for all* $j \in J$, $inf(l'_0 :!M'_j.P'_j) = 1$ *and* $sup(l'_0 :!M'_j.P'_j) = \infty$;
- *for all other places* $l'_0 : Q$ *not considered in the previous items,* $inf(l'_0 : Q) = 0$ *and either* $sup(l'_0 : Q) = 0$, *if* $T \equiv_T S$, *or* $sup(l'_0 : Q) = \infty$, *if* $T \equiv_T S|\texttt{any}$.

We define the set of target markings associated to the source process P, the target T and the sequence of labels $l_0\lambda$ as follows:

$$\begin{aligned} TargMark(P,T,l_0\lambda) \ = \{&(infTree, supTree) \oplus (inf, sup) \mid \\ &(inf, sup) \in TargMarkSub(P,T,l_0\lambda) \wedge \\ &(infTree, supTree) \text{ satisfes the property below } \} \end{aligned}$$

where $(infTree, supTree)$ is a target marking defined on the set of places $\mathcal{T}_{\#amb(T)} \cup \bigcup_{l \in l_0\lambda} l : unused$ such that:

- $infTree(l_0 \cdot tree(T,\lambda)) = 1$ and $supTree(l_0 \cdot tree(T,\lambda)) = 1$;
- if $p \neq l_0 \cdot tree(T,\lambda)$ then $infTree(p) = 0$ and $supTree(p) = 0$.

Note that given a process P, a target T, and a sequence of labels λ, the set of target markings $TargMark(P,T,\lambda)$ is finite.

We are now ready to formalize the correspondence between satisfiability of a target for processes and satisfiability of a target marking for P/T nets.

Proposition 3.17 *Let* P, R *be* BA^-_{nomrg} *processes such that* $P \to^* R$. *Let* T *be a well formed target reachable from* P, *i.e.* $TReach(P,T)$. *We have that* $R \in set(T)$ *if and only if for every sequence* λ *of distinct labels in* $\{l_1, \ldots, l_{\#amb(T)}\}$ *such that* $|\lambda| = \#amb(R)$ *we have that* $dec(R, l_0\lambda)$ *satisfies at least one target marking in* $TargMark(P,T,l_0\lambda)$.

We finally conclude formalizing the reduction of the target reachability of processes to the target marking reachability problem on P/T nets.

Theorem 3.18 *Let* P *be a* BA^-_{nomrg} *process and let* T *be a well formed target such that* $\#amb(P) \leq \#amb(T)$.

We have that T *is reachable from* P, *i.e.* $TReach(P,T)$, *if and only if there exists a sequence* λ *of distinct labels in* $\{l_0, \ldots, l_{\#amb(T)}\}$ *such that* $|lambda| = \#amb(T) + 1$ *and at least one of the target markings in* $TargMark(P,T,\lambda)$ *is reachable in* $Net(P, \#amb(T))$.

Proof By definition, the target T is reachable by the process P if and only if there exists a process R such that (i) $R \in set(T)$ and (ii) R is reachable from

the process P. By Proposition 3.17 we have that (i) holds if and only if for every sequence of labels $\lambda = l_0\lambda'$ we have that $dec(R, \lambda)$ satisfies at least one of the target markings in $TargMark(P, T, \lambda)$. By Theorem 5.4 we have that (ii) holds if and only if there exists a sequence of labels λ such that $dec(R, \lambda)$ is reachable in $Net(P, \#amb(T))$. □

As a trivial corollary we have that the target reachability problem is decidable in BA^-_{nomrg}.

Corollary 3.19 *Let P be a BA^-_{nomrg} process and T be a well formed target. The Target reachability problem $TReach(P, T)$ is decidable.*

Proof By Theorem 3.18 we have that in order to solve the target reachability problem for the BA^-_{nomrg} process P and the target T, one can check the reachability of one of the target markings in the set $\oplus_\lambda TargMark(P, T, l_0\lambda)$, where λ ranges over the set of sequences of distinct labels in $\{l_1, \ldots, l_{\#amb(T)}\}$, in the P/T system $Net(P, \#amb(T))$.

As the above set of sequences of labels is finite, and for each sequence λ we have that $TargMark(P, T, l_0\lambda)$ is finite (see the observation after Definition 3.16), we can conclude that also the set of target markings $\oplus_\lambda TargMark(P, T, l_0\lambda)$ to be considered is finite. Hence, the decidability result directly follows from the decidability of target marking reachability for P/T systems proved in the Theorem 3.6. □

4 Undecidability of Target Reachability in $\mathbf{BA}^-_{mntn}$

We have observed in the previous section that the monotonicity of the number of active ambients in a computation is a fundamental property in the proof of decidability of Target reachability. In this section we show that monotonicity is not a sufficient condition. In fact, we prove that Target reachability is no longer decidable in BA^-_{mntn} even if in this calculus the considered form of *merge* cannot decrease the total number of active ambients. After, we prove that even if Target reachability is not decidable for BA^-_{mntn}, the weaker Spatial reachability problem is still computable. The proof is by reduction to the coverability problem for Petri with *transfer arcs*, an extension of Petri nets in which transitions have the possibility to move all the tokens in given places to another one.

The proof of undecidability is by reduction from an undecidable problem for Two Counter Machines (2CM), a well-known register based Turing powerful formalism. We now report a definition of Two Counter Machines convenient

for our aims.

Definition 4.1 –Two Counter Machine (2CM)– *A Two Counter Machine is defined by a set of instructions and by two counters c_1 and c_2 that hold natural numbers. We assume that every instruction is stored at a location denoted with L, M, ... The instructions are of four possible kinds: $INC_i(L, M)$, $DEC_i(L, M)$, $TSTZ_i(L, M)$ and $TSTNZ_i(L, M)$ where L is the location of the instruction, M is the location of the next instruction to execute, and i is the index of the counter on which the instruction acts: INC increments the contents of the counter, DEC decrements the counter if it is not empty, TSTZ tests if the counter is empty, TSTNZ tests if it is not empty. In order to model conditional tests, we allow the two distinct test instructions to be stored in the same location. The initial configuration of a 2CM consists of an initial location L and of the two counters initially empty. The* loop *problem for 2CM consists of checking the existence of a non-empty computation that starts and then returns back to the initial configuration. It is trivial to reduce the halting problem for 2CMs to the loop problem, hence also the latter is undecidable.*

We now prove, by reduction from the 2CM loop problem, that Target reachability is undecidable for BA^{-}_{mntn}.

Theorem 4.2 *Target reachability is undecidable in BA^{-}_{mntn}.*

Proof We exhibit a *weak* simulation into BA^{-}_{mntn} of 2CMs. It is weak because the simulation of the test-for-zero instruction is not completely faithful as it can be executed also when the register is not empty. Nevertheless, if this is the case, the simulation leaves some garbage. If no garbage is left by a simulated computation, then it faithfully reproduces a computation of the corresponding 2CM.

Consider a 2CM with instructions $I_1, \ldots, I_n$. The current configuration is encoded using 5 ambients that we will label as *prog*, *loc*, c_1, c_2, and g: *prog* contains the encoding of the instructions, *loc* keeps track of the current control location, c_1 and c_2 keep track of the current values of the counters, g has a subambient h needed to collect (and keep separated from the other ambients) all local agents representing "units" when the zero-test is (weakly) simulated.

Specifically, the encoding of a 2CM with instruction $I_1, \ldots, I_n$ and initial location L is defined as follows:

$$P_0 = \underbrace{[!\llbracket I_1 \rrbracket \mid \ldots \mid !\llbracket I_n \rrbracket]}_{prog} \mid \underbrace{\llbracket L \rrbracket}_{loc} \mid \underbrace{\llbracket c_1 = 0 \rrbracket}_{c_1} \mid \underbrace{\llbracket c_2 = 0 \rrbracket}_{c_2} \mid \underbrace{[!accept\ g \mid \overbrace{[!merge_{-}\ h]}^{h}]}_{g}$$

The ambient *loc* keeps track of the current location L:

$$[\![L]\!] = [merge_{-}\ L]$$

To represent $c_i = k$ we fill the ambient c_i with k occurrences of the process $merge_{-}\ c_i$ and a single occurrence of the process $merge_{-}\ z_i$:

$$[\![c_i = k]\!] = [merge_{-}\ z_i \mid \underbrace{merge_{-}\ c_i \mid \ldots \mid merge_{-}\ c_i}_{k}]$$

for $i : 1, 2$.
The encoding of the instructions is defined as follows.
If $I = DEC_i(L, M)$, then $[\![I]\!]$ is defined as $merge_{+}L.(\mathbf{A_1} \mid expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_{+}\ c_i.(\mathbf{A_2} \mid expel\ a)]$$
$$\mathbf{A_2} = [exit\ a.merge_{-}\ M]$$

The intuition of the previous definition is as follows. The *loc* ambient is first merged with the *prog* ambient using the synchronization label L (the current location). This action creates the ambient $\mathbf{A_1}$ that is expelled by the merged ambients immediately after. $\mathbf{A_1}$ is merged with the ambient c_i (thus consuming a "unit", i.e., a local agent $merge_{-}\ c_i$). The ambient $\mathbf{A_2}$ is created inside the resulting merged ambients and expelled to become $[\![M]\!]$.

If $I = INC_i(L, M)$, then $[\![I]\!]$ is defined as $merge_{+}\ L.(\mathbf{A_1} \mid expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_{+}\ z_i.(\mathbf{A_2} \mid expel\ a \mid merge_{-}\ z_i \mid merge_{-}\ c_i)]$$
$$\mathbf{A_2} = [exit\ a.merge_{-}\ M]$$

Again the *loc* ambient is first merged with the *prog* ambient via L. This action creates the ambient $\mathbf{A_1}$ that is expelled by the merged ambients immediately after. $\mathbf{A_1}$ is merged with the ambient c_i by performing $merge_{-}\ z_i$. The ambient $\mathbf{A_2}$ is created inside the resulting ambient, say $\mathbf{A_1}+c_i$, and expelled to become $[\![M]\!]$. In the meantime two new "units" are released: one to compensate the $merge_{-}\ z_i$ consumed to execute the merge, and one unit $merge_{-}\ c_i$ for the increment.

The encoding of the zero test is more tricky and exploits the ambient we called g (garbage) at the begininning of the proof. If $I = TSTZ_i(L, M)$, then $[\![I]\!]$ is

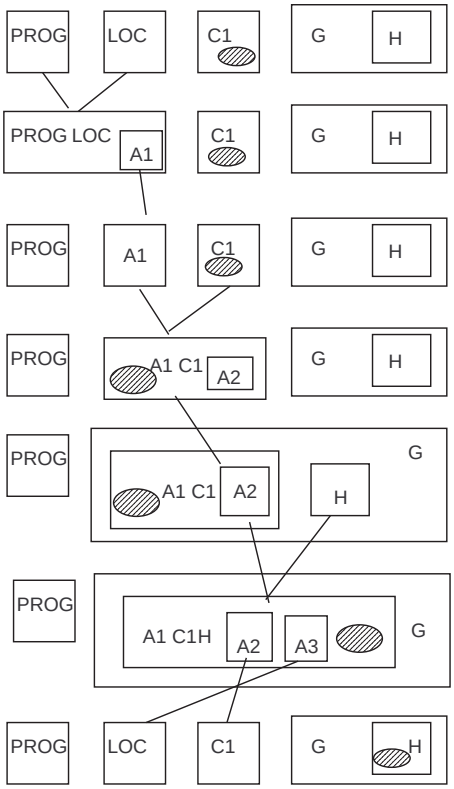

Fig. 2. Encoding of zero test.

defined as $merge_+\ L.(\mathbf{A_1} \mid expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_+\ z_i.(\mathbf{A_2} \mid enter\ g.P \mid expel\ b.expel\ d)]$$
$$P \quad = merge_+\ h.(\mathbf{A_3} \mid expel\ a.expel\ c)$$
$$\mathbf{A_2} = [exit\ a.exit\ b.merge_-\ z_i]$$
$$\mathbf{A_3} = [exit\ c.exit\ d.merge_-\ M]$$

The intuition is as follows. The *loc* ambient is first merged with the *prog* ambient via L. This action creates the ambient $\mathbf{A_1}$ that is expelled by the merged ambients immediately after. $\mathbf{A_1}$ is merged with the ambient c_i via the label z_i. $\mathbf{A_2}$ (that will become $[\![c_i = 0]\!]$) is released inside the resulting ambient, we will refer to as $\mathbf{A_1} + c_i$. At this stage, $\mathbf{A_1} + c_i$ enters g, merges with h, and creates another internal ambient $\mathbf{A_3}$ (that will become $[\![M]\!]$). As last steps, $\mathbf{A_2}$ and $\mathbf{A_3}$ are moved at top level in sequence. If the counter c_i was not zero, then the local agents inside c_i remain blocked inside the subambient h of g. This way they cannot interact with the other ambients at the top level. This idea is graphically illustrated in Fig. 2, where the grey oval represents the local agents (units) moved from the c_i ambient to the h ambient.

Finally, if $I = TSTNZ_i(L, M)$, then $[\![I]\!]$ is defined as $merge_+\ L.(\mathbf{A_1} \mid accept\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_+\ c_i.(\mathbf{A_2} \mid merge_-\ c_i \mid expel\ a)]$$

$$\mathbf{A_2} = [exit\ a.merge_-\ M]$$

The construction we have defined reduces the 2CM loop problem, assuming that the initial location is L, to the problem of checking the existence of a non-empty computation that starts from P_0 and leads to P_0. On the one hand, a 2CM computation that starts and returns back to the initial configuration is simulated by P_0 with a non-empty computation leading to P_0. On the other hand, every computation leading to P_0 reproduces faithfully a 2CM computation as in P_0 there is no garbage inside the ambient h. Finally, we observe that in order to check the existence of a computation $P_0 \rightarrow^+ P_0$ it is sufficient to consider every process Q such that $P_0 \rightarrow Q$ (notice that there are only finitely many of such processes) and tests whether $TReach(Q, T_0)$, where T_0 is a target obtained from P_0 by replacing the three guarded processes $merge_-\ L$, $merge_-\ z_1$, and $merge_-\ z_2$ with the targets $1 \leq merge_-\ L \leq 1$, $1 \leq merge_-\ z_1 \leq 1$, and $1 \leq merge_-\ z_2 \leq 1$, respectively. □ As a corollary of the previous theorem, we also have that reachability of a given process is an undecidable problem. Indeed, the type of target expression we considered is of the form $1 \geq P \geq 1$ which corresponds to fix a single instance of each guarded process.

5 Deciding Spatial Reachability in BA^-_{mntn}

Despite Target reachabiliy is undecidable for BA^-_{mntn}, the weaker Spatial reachability problem (that considers targets with `any` in every ambient and with all upper bounds equal to ∞) is still decidable. The proof is still by reduction to Petri nets, but we need to move to a different class of Petri nets including also transfer arcs. Transfer arcs are associated to transitions, and are used to move all the tokens inside some given source places to a given target place.

Definition 5.1 –P/T net with Transfer arcs– *A P/T net with transfer arcs is a P/T net (S, T) where to each transition $t = (c, p)$ in T, it is associated a (possibly empty) set γ of* transfer arcs *(we write these transitions in the form $c \rightarrow p[\gamma]$). A transfer arc is defined by a pair composed of a set of places S (the source places) and by a target place p. Transfer arcs must work on disjoint source and target places. If a transition $t = (c, p)[\gamma]$ has the transfer arcs $\gamma = \{\langle S_1, p_1\rangle, \ldots, \langle S_n, p_n\rangle\}$ (where we assume $(S_i \cup \{p_i\}) \cap (S_j \cup \{p_j\}) = \emptyset$, for every pair of disjoint indexes i and j), then its execution $m \xrightarrow{t} m'$ is computed as follows. We first compute the marking $m_1 = (m \setminus c)$, then we execute all*

$$
\begin{array}{ll}
 & C[[l^m \cdot \mu^m] \oplus [l^n \cdot \mu^n]], \quad l^m : merge_+\, k.a, \quad l^n : merge_-\, k.a', \\
 & \bigcup_{l \in \lambda} l : unused, \quad \bigcup_{l' \in \lambda'} l' : unused \\
 & \downarrow \\
(\texttt{merge}) & C[[l^m \cdot (\mu^m \oplus \mu^n \oplus tree(a, \lambda) \oplus tree(a', \lambda'))]], \quad l^n : unused, \\
 & l^m : actproc(a), \quad l^m : actproc(a'), \quad proc(a, \lambda), \quad proc(a', \lambda') \\
 & [\{\ \langle \{l^n : Q\}, l^m : Q \rangle \ \mid Q \in \mathcal{Q}\}]
\end{array}
$$

Table 2
The schema for the transitions $Trans(\mathcal{Q})$ for *merge*, defined parametrically on the set of sequential and replicated subprocesses $\mathcal{Q}$. We assume that λ and λ' are sequences of distinct labels such that $|\lambda| = \#amb(a)$ and $|\lambda'| = \#amb(a')$.

transfers, i.e., for every $i : 1, \ldots, n$ move all tokens from the places in S_i in m'' to the place p_i. Let m_2 be the resulting marking. Then, $m' = m_2 \oplus p$.

The coverability problem is known to be decidable for P/T nets with transfer arcs [12].

In order to prove that Spatial reachability is decidable for BA^-_{mntn} we proceed similarly to the proof reported in Section 3: given a process P and a spatial target S, we construct a P/T net with transfer arcs and we check whether at least one among a finite set a markings is coverable in such a net.

The P/T net (with transfer arcs) is defined as in Section 3, the unique difference is the addition of the transitions defined in Table 2. The new transitions deal with the *merge* capability: when a *merge* occurs, all the active processes inside the net region hosting the process performing $merge_-$ are transferred to the net region hosting the process performing the complementary $merge_+$. The former net region is freed (i.e. it is marked unused), and the ambient tree is updated accordingly.

Definition 5.2 *Let P be a BA^-_{mntn} process and let n be a natural number such that $\#amb(P) \leq n$. We define the net $Net(P, n) = (Pl, Tr)$, where*

$$
\begin{aligned}
Pl &= \textstyle\bigcup_{i=0}^{n}(\{l_i : Q \mid Q \in sub(\alpha(P))\} \cup \{l_i : unused\}) \ \cup \ \mathcal{T}_n \\
Tr &= \{c \to p[\emptyset] \mid (c, p) \in Trans, c, p \subseteq Pl\} \cup \\
&\qquad \{c \to p[\gamma] \in Trans(sub(\alpha(P))) \mid c, p \subseteq Pl\}
\end{aligned}
$$

Notice that $Trans(sub(\alpha(P)))$ is the set of transitions defined by the schema in Table 2.

We now discuss how to use the P/T net defined above to verify Spatial reachability. We first formalize the monotonicity property that we have already informally discussed so far.

Proposition 5.3 *Let P, Q be BA^-_{mntn} processes such that $P \rightarrow^* Q$. We have that $\#amb(P) \leq \#amb(Q)$.*

Proof By induction on the length of $P \rightarrow^* Q$. If the length is 0 the proposition trivially holds. If $P \rightarrow P' \rightarrow^* Q$ we have, by the inductive hypothesis, that $\#amb(P') \leq \#amb(Q)$. Proceeding by induction on the length of the proof of $P \rightarrow P'$ it is easy to prove that $\#amb(P) \leq \#amb(P')$. The unique non trivial case is when the *merge* rule is applied. In general, the application of such rule could decrease by one the number of active ambients, but this is not possible in BA^-_{mntn} as in the continuation of the process performing $merge_+$ there is always at least one new active ambient. □

Following the same proof technique in [6], it is possible to prove the following result.

Theorem 5.4 *Let P, Q be BA^-_{mntn} processes such that $\#amb(P) \leq \#amb(Q)$. We have that $P \rightarrow^* Q$ iff there exists a sequence λ of distinct labels in $\{l_0, \ldots, l_{\#amb(Q)}\}$ such that $|\lambda| = \#amb(Q) + 1$ and $dec(Q, \lambda)$ is a marking of $Net(P, \#amb(Q))$ that is reachable from the marking $dec(P, l_0 \ldots l_{\#amb(P)})$.*

This last theorem formalizes the correctness of the P/T net semantics: given a maximal number n, all the derivatives of a process P with less than n active ambients corresponds to the processes whose marking is reachable in $Net(P, n)$ starting from the marking $dec(P, l_0 \ldots l_{\#amb(P)})$.

Now, in order to reduce Spatial reachability to net coverability, we need to characterize the set of processes that satisfies a spatial target S, i.e. $set(S)$, as an unpward closed set of net markings. More precisely, given a process P and a spatial target S with n active ambients, we show the existence of a finite set $min(S, P)$ of markings of $Net(P, n)$ such that a process Q, reachable from P, satisfies the spatial target S if and only if its decomposition covers one of the markings in $min(S)$.

Definition 5.5 –min(S)– *Let P be a BA^-_{mntn} process and let S be the well formed spatial target*

$$\texttt{any} \mid \prod_{i \in I} q_i \leq G_i \leq \infty \mid \prod_{j \in J} !G'_j \mid \prod_{k \in K} [S'_k]$$

We define $min(S, P)$ *as the following set of markings of* $Net(P, \#amb(S))$*:*

$$\{\ dec(Q, \lambda) \mid Q \in minProc(S, P), \\ \lambda \text{ is a sequence of distinct labels in } \{l_0, \ldots, l_{\#amb(S)}\}\ \}$$

where $minProc(S, P)$ *is the following set of processes:*

$$\{\textstyle\prod_{i\in L}\prod_{q_i} G_i \mid \prod_{i\in I\setminus L}!G_i \mid \prod_{j\in J}!G'_j \mid \prod_{k\in K}[Q'_k] \quad \mid \\ L \subseteq I \ \wedge\ Q'_k \in minProc(S'_k, P) \ \wedge\ \forall i, j : G_i, !G_j \in sub(\alpha(P))\}$$

Notice that in the definition of $minProc(S, P)$ we consider the possibility to satisfy the target $q_i \leq G \leq \infty$ in two possible ways: either with the presence of at least q_i instances of G_i, or with the presence of the replicated process $!G_i$. Moreover, it is easy to see that both the sets $minProc(S, P)$ and $min(S, P)$ are finite. We now prove that the upward closed set of markings defined by the basis $min(S, P)$ correctly characterizes the derivatives of P that satisfies the spatial target S.

Theorem 5.6 *Let* P *be a* BA^-_{mntn} *process, let* S *be a well formed spatial target, and let* Q *be a derivative of* P*. We have that* $Q \in set(S)$ *if and only if there exists a marking* $m \in min(S, P)$ *and a sequence* λ *containing the distinct labels* $\{l_0, \ldots, l_{\#amb(S)}\}$ *such that* $m \subseteq dec(Q, \lambda)$*.*

Proof Let m be a marking in $min(S, P)$ such that $dec(Q, \lambda)$.
We start with the *only-if* part. To prove that $Q \in set(S)$ we proceed by induction on the number of active ambients in Q. It is sufficient to observe –both in the base and in the inductive case– that the process Q (at top level) must include at least the same sequential and replicated subprocesses (at top level) in the marking m, thus satifying the constraints imposed (at top level) by the definition of $set(S)$ (see Definition 2.10). The presence (at top level) in S of the target `any`, as well as the usage of the upper limit ∞, play here a fundamental role: in fact, Q could contain also other processes besides the minimal ones present in m.
The *if* part is proved in a similar manner. □

We can finally conclude with the main result of this section.

Corollary 5.7 *The Spatial reachability problem is decidable in* BA^-_{mntn}*.*

Proof Consider a BA^-_{mntn} process P and a well formed spatial target S with n active ambients. We have that also the processes in $set(S)$ have n active ambients so, by Proposition 5.3, if $\#amb(P) > n$ than $SReach(P, S)$ does not hold. So it is not restrictive to assume that $\#amb(P) \leq n$. By Theorems 5.4

and 5.6 we have that checking $SReach(P, S)$ corresponds to verifying the coverability, in the P/T net with transfer arcs $Net(P, n)$, of at least one of the markings in $min(S, P)$ starting from the initial marking $dec(P, l_0 \dots l_n)$. As the set of markings to be considered is finite, the thesis follows from the decidability of coverability for P/T nets with transfer arcs [12]. □

6 Undecidability of Spatial Reachability in BA$^-$

We now show that the monotinicity of the number of active ambients is a necessary condition for the decidability of Spatial reachability. In fact, if we consider the general *merge* operator of BA$^-$, that allows one to reduce the number of active ambients, then Spatial reachability is no longer decidable.

Theorem 6.1 *Spatial reachability is undecidable in BA$^-$.*

Proof We exhibit an encoding of Two Counter Machines (2CMs). For simplicity, we consider a nondeterministic modeling of the increment and decrement operations. We represent these instructions with two distinct encodings that reside in the same location. For instance, we present an encoding for decrements that works fine when the counter contains 1, and an encoding for the other cases. When a decrement instruction should be simulated, one of the two encodings is selected nondeterministically. In case the wrong one is chosen, the simulation blocks. The same is done also for increment, separating the case in which the counter is empty from the case in which it is not.

Consider a 2CM with instructions $I_1, \dots, I_n$. The current configuration is encoded using four ambients that we will label as *prog*, *loc*, c_1, and c_2: *prog* contains the encoding of the instructions, *loc* keeps track of the current control location, c_1 and c_2 keep track of the current values of the counters. Assuming the initial location L, the initial configuration is defined as follows

$$P_0 \;=\; \underbrace{[\; !\llbracket I_1 \rrbracket \mid \dots \mid !\llbracket I_n \rrbracket \;]}_{prog} \mid \underbrace{\llbracket L \rrbracket}_{loc} \mid \underbrace{\llbracket c_1 = 0 \rrbracket}_{c_1} \mid \underbrace{\llbracket c_2 = 0 \rrbracket}_{c_2}$$

The encoding is defined using the set of labels $\mathcal{L} = Loc \cup \{a, b, z_1, z_2, c_1, c_2\}$.

The ambient *loc* keeps track of the current location L via the following process

$$\llbracket L \rrbracket = [merge_{-}\ L]$$

To represent $c_i = 0$, we use the following ambient

$$\llbracket c_i = 0 \rrbracket \;=\; [!exit\ z_i \mid !merge_{-}\ z_i]$$

for $i : 1, 2$. To represent $c_i = k$ with $k > 0$, we use the ambient

$$[\![c_i = k]\!] \ ::= \ [merge_- \ c_i \mid [\![c_i = k - 1]\!]]$$

for $i : 1, 2$.

The encoding of the instructions is defined as follows.

If $I = DEC_i(L, M)$ and $c_i = 1$, then $[\![I]\!] = merge_+ \ L.(\mathbf{A_1} \mid expel \ a)$, where

$$\mathbf{A_1} = [exit \ a.merge_+ \ c_i.expel \ z_i.merge_- \ M]$$

The intuition of the previous definition is as follows. The *loc* ambient is first merged with the *prog* ambient using the synchronization label L (the current location). This action creates the ambient $\mathbf{A_1}$ that is expelled by the merged ambient immediately after. $\mathbf{A_1}$ is merged with the ambient c_i. The resulting ambient expels the z_i ambient ($c_i = 1$) and then becomes the new ambient for the new location $[\![M]\!]$.

If $I = DEC_i(L, M)$ and $c_i > 1$, then $[\![I]\!] = merge_+ \ L.(\mathbf{A_1} \mid expel \ a)$, where

$$\mathbf{A_1} = [exit \ a.merge_+ \ c_i.(\mathbf{A_2} \mid expel \ a.merge_- \ M)]$$
$$\mathbf{A_2} = [merge_+ \ c_i.exit \ a.merge_- \ c_i]$$

As in the previous case the *loc* ambient is first merged with the *prog* ambient using the synchronization label L (the current location). This action creates the ambient $\mathbf{A_1}$ that is expelled immediately after. $\mathbf{A_1}$ is merged with the ambient c_i. A new ambient $\mathbf{A_2}$ is created inside the resulting merged ambient say $\mathbf{A_1} + c_i$. $\mathbf{A_2}$ is merged with the c_i ambient at the same level and the resulting ambient is moved at top level (it represents $c_i - 1$) while $\mathbf{A_1} + c_i$ is transformed in the ambient $[\![M]\!]$.

If $I = INC_i(L, M)$ and $c_i = 0$, then $[\![I]\!] = merge_+ \ L.(\mathbf{A_1} \mid \mathbf{B_1} \mid expel \ a.expel \ a)$, where

$$\mathbf{A_1} = [exit \ a.merge_+ \ z_i.enter \ a.\mathbf{A_2}] \mid expel \ b$$
$$\mathbf{A_2} = [exit \ b.exit \ a.merge_- \ M]$$
$$\mathbf{B_1} = [exit \ a.accept \ a.expel \ a.merge_- \ c_i]$$

As for DEC the *loc* ambient is first merged with the *prog* via L (the current location). This action creates the ambients $\mathbf{A_1}$ and $\mathbf{B_1}$ that are expelled immediately after. $\mathbf{A_1}$ is merged with the ambient z_i and then enters inside $\mathbf{B_1}$ where it releases an ambient $\mathbf{A_2}$. $\mathbf{A_2}$ is expelled by the two nested ambients and, thus, moved at top level as the new location $[\![M]\!]$. In the meantime $\mathbf{B_1}$ creates a local agent $merge_- \ c_i$ to become $[\![c_i = 1]\!]$.

A similar idea is used for $c_i > 1$. Formally, if $I = INC_i(L, M)$ and $c_i > 1$, then $[\![I]\!] = merge_+\ L.(\mathbf{A_1} \mid \mathbf{B_1} \mid expel\ a.expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_+\ c_i.enter\ a.(\mathbf{A_2} \mid merge_-\ c_i)] \mid expel\ b$$
$$\mathbf{A_2} = [exit\ b.exit\ a.merge_-\ M]$$
$$\mathbf{B_1} = [exit\ a.accept\ a.expel\ a.merge_-\ c_i]$$

The tests $c_i = 0$ and $c_i > 0$ are simulated by using merge steps either with label z_i or with label c_i.

Formally, if $I = TSTZ_i(L, M)$, then $[\![I]\!] = merge_+\ L.(\mathbf{A_1} \mid expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_+\ z_i.(\mathbf{A_2} \mid expel\ a)]$$
$$\mathbf{A_2} = [exit\ a.merge_-\ M]$$

If $I = TSTNZ_i(L, M)$, then $[\![I]\!] = merge_+\ L.(\mathbf{A_1} \mid expel\ a)$, where

$$\mathbf{A_1} = [exit\ a.merge_+\ c_i.(merge_-\ c_i \mid \mathbf{A_2} \mid expel\ a)]$$
$$\mathbf{A_2} = [exit\ a.merge_-\ M]$$

This construction reduces the 2CM halting problem, assuming that the initial location is L, to the problem of checking the existence of a computation that starts from P_0 and leads to a process having at top level an ambient containing the active process $merge_-\ M$ for some halt location M (i.e. a location that does not contain any instruction). But this problem corresponds to check whether $SReach(P_0, [merge_-\ M|\texttt{any}]|\texttt{any})$. Hence the thesis directly follows from the undecidability of the halting problem for 2CM. □

7 Conclusion

In this paper we applied the techniques introduced in [4,5] to study the decidability of different types of reachability problems in fragments of BioAmbients with a restricted use of the *merge* capability. The fragments studied in this paper do not allow communication and restrictions. Furthermore, replication is applied only to guarded processes of the form $M.P$. This restrictions correspond to those introduced for the Mobile Ambients in [4]. We considered here two types of reachability problems: target reachability (reachability of configurations with constraints on occurrences of guarded processes); spatial reachability (reachability of configurations in which the ambient structure of the target configuration is fixed a priori and in which we pose lower bounded constraints on the number of occurrences of guarded processes). Under the

previous assumptions, target reachability turns out to be decidable without *merge* capability even if the resulting fragment (that comprises movement operations) is still Turing complete. Furthermore, we proved that spatial reachability is decidable when adding a restricted use of the *merge* capability that ensures that its application does not decrease the number of ambients of the current configuration. Interestingly, target reachability is undecidable in the resulting fragment.

As for the case of Mobile Ambients, reachability is used here as a technical tool for a fine comparison of different features of BioAmbients that go beyond standard analysis based on Turing completeness. The present paper is based on preliminary work separately done by the authors in [10,17]. In [10] Delzanno and Montagna study decidability issues for reachability and spatial reachability with restricted use of the *merge* capability and a restricted semantics for replication inspired to [3]. In [17] Zavattaro studies target reachability in fragments with guarded replication and without the *merge* capability. In the present paper we draw a complete picture of the aspects studied in [10,17] by taking the fragment with guarded replication as common idiom to reformulate decidability and undecidability results for target and spatial reachability with no or restricted use of the *merge* capability. Furthermore, from a technical point of view, we present a simplified proof of the decidability result for Spatial Reachability in [10]. Specifically, we prove the result with a direct encoding into Petri nets with transfer arcs without need of resorting to the term rewriting calculus used in [17].

The use of reachability problems for the analysis of the expressive power of biological models is considered in the context of extensions of Gheorge Paun's P-systems in [11] with operations like creation, deletion, cloning and fusion of membranes, and for Danos-Laneve's κ-model in [9]. P-systems and BioAmbients are both computational models defined over hierarchical structures. Fusion and merge have similar effect on the corresponding data structures (membranes and bioambients). However the extensions of P-systems studied in [11] do not provide for movement capability (a feature that, taken alone, may lead to Turing completeness) as the fragments of BioAmbients studied in the current paper. Furthermore, target reachability is not considered in [11]. A uniform analysis of BioAmbients and P-systems with different notions of reachability based in the style of [1,2] represents an interesting direction for further investigations.

The κ-model [8] is a formalism based on graph rewriting specifically designed for modeling intermolecular interaction, thus operating on a different level of granularity with respect to BioAmbients. Even when focusing on the structures underlying the two models, a comparison of the results obtained in the present paper and those studied in [9] would require a better understanding of the relation between the corresponding models which differ both in the type

of configurations (trees vs graphs) and in the different features of the specification language (movement and merge capabilities vs graph matching and replacement).

References

[1] B. Aman and G. Ciobanu. On the Reachability Problem in P Systems with Mobile Membranes. In Proc. of WMC'07, pages 113–123, volume 4860 of *Lecture Notes in Computer Science*. Springer, 2007.

[2] Bogdan Aman and Gabriel Ciobanu. Turing Completeness Using Three Mobile Membranes. In Proc. of UC'09, pages 42–55, volume 5715 of *Lecture Notes in Computer Science*. Springer, 2009.

[3] I. Boneva and J.-M. Talbot. When Ambients Cannot be Opened. *Theoretical Computer Science*, 333: 127-169, Elsevier, 2005.

[4] N. Busi and G. Zavattaro. Deciding Reachability in Mobile Ambients. In Proc. of ESOP'05, pages 248–262, volume 3444 of *Lecture Notes in Computer Science*. Springer, 2005.

[5] N. Busi and G. Zavattaro. Reachability Analysis in Boxed Ambients. In Proc. of ICTCS'05, pages 143–159, volume 3701 of *Lecture Notes in Computer Science*. Springer, 2005.

[6] N. Busi and G. Zavattaro. Deciding reachability problems in Turing-complete fragments of Mobile Ambients. *Mathematical Structures in Computer Science*, 19(6):1223–1263. Cambridge University Press, 2009.

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

[8] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1): 69-110. Elsevier, 2004

[9] G. Delzanno, C. Di Giusto, M. Gabbrielli, C. Laneve, G. Zavattaro. The κ-Lattice: Decidability Boundaries for Qualitative Analysis in Biological Languages. In Proc. of CMSB'09, pages 158–172, volume 5688 of *Lecture Notes in Computer Science*. Springer, 2009.

[10] G. Delzanno and R. Montagna. On Reachability and Spatial Reachability in Fragments of BioAmbients. In Proc. of MeCBIC'06, volume 171(2) of *Electronic Notes in Theoretical Computer Science*, pages 69-79. Elsevier, 2007.

[11] G. Delzanno and L. Van Begin. On the verification of membrane systems with dynamic structure. *Natural Computing*, 9(4): 795-818. Springer, 2010.

[12] C. Dufourd, A. Finkel, and P. Schnoebelen. Reset Nets Between Decidability and Undecidability. In Proc. of ICALP'98, pages 103–115, volume 1443 of *Lecture Notes in Computer Science*. Springer, 1998.

[13] S. Maffeis and I. Phillips. On the Computational Strength of Pure Mobile Ambients. *Theoretical Computer Science*, 330: 501-551, Elsevier, 2005.

[14] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Journal of Information and Computation*, 100:1–77. Academic Press, 1992.

[15] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E.Y. Shapiro. BioAmbients: an abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, Elsevier, 2004.

[16] W. Reisig. *Petri nets: An Introduction.* EATCS Monographs in Computer Science, Springer, 1985.

[17] G. Zavattaro. Reachability Analysis in BioAmbients. In *Proc. MeCBIC'08*, volume 227 of *Electronic Notes in Theoretical Computer Science*, pages 179–193. Elsevier, 2009.