On the Dynamics of PB Systems with Volatile Membranes

Giorgio Delzanno[1] and Laurent Van Begin[2]*

[1] Università di Genova, Italy. E-mail: giorgio@disi.unige.it
[2] Université Libre de Bruxelles, Belgium. E-mail: lvbegin@ulb.ac.be

Abstract. We investigate decision problems like reachability and boundedness for extensions of PB systems with volatile membranes. Specifically, we prove that reachability and boundedness are decidable for PB systems extended with rules for membrane dissolution. For PB systems extended with membrane creation, reachability is still decidable whereas boundedness becomes undecidable. Furthermore, we show that both problems are undecidable for PB systems extended with both dissolution and creation rules. Finally, we prove that reachability and boundedness become decidable for PB systems with dissolution rules and in which only one instance of each type of membrane can be created during a computation. Our work extends previous results obtained for PB systems by Dal Zilio and Formenti in a paper appeared in the proceedings of WMC 2003 [5].

1 Introduction

The PB systems of Bernardini and Manca [2] are a variant of P-systems [16] in which rules can operate on the boundary of a membrane. A *boundary rule* can be used here to move multisets of objects across a membrane. In biological modeling, PB are very useful for expressing complex interactions among biological membranes [8]. For this reason, it seems important to develop methods for qualitative and quantitative analysis of models specified in this formalism. In this paper we focus our attention on theoretical issues for the qualitative analysis of PB systems. Some preliminary results on decision problems for PB systems have been obtained in [5]. Specifically, in [5] Dal Zilio and Formenti proved that the reachability problem is decidable for PB systems with symbol objects. The *reachability problem* consists in checking if a given system can evolve into a fixed a priori configuration. The decidability proof in [5] is based on an encoding of PB systems into Petri nets [17], an infinite-state model of concurrent systems for which the reachability problem is decidable [14, 11]. A Petri net is a collection of places that contain tokens, and of transitions that define how tokens move from one place to another. The current configuration of a net is called marking. A marking specifies the current number of tokens in each place. A PB system can be encoded as a Petri net in which membranes are modelled as places,

* Research fellow supported by the Belgian National Science Foundation (FNRS).

symbol objects as tokens, configurations as markings, and internal/boundary rules as transitions. The execution of a rule is simulated then by the firing of the corresponding Petri net transition. The Petri net encoding shows that the reachability problem is decidable in PB systems. The same reduction can be used to decide other properties like *boundedness* [6]. In [5] the authors observe that the aforementioned encoding can be extended to more sophisticated Petri net models so as to deal with dynamically changing membrane structures. As an example, Petri net transitions extended with transfer arcs naturally model the dissolution of a membrane. Indeed, a transfer arc can be used to atomically transfer all tokens from one place to another. This operation can be applied to move the content of a dissolved membrane to its father. Unfortunately, as pointed out in [5], this connection cannot be exploited in order to extend the decidability results obtained for PB systems. Indeed, problems like reachability and boundedness become undecidable in presence of transfer or reset arcs [6]. The decidability of reachability and boundedness for PB systems with volatile or moving membrane seems to be still an open problem. In this paper we focus our attention on decision problems for extensions of PB systems with dissolution and creation rules. More specifically, our technical results are as follows.

We first show that reachability is decidable in PB systems with dissolution rules (PBD systems). Dissolution rules are a peculiar feature of P-systems. Thus, PBD systems represent a natural extension of PB system. Our decidability proof is still based on a reduction to a Petri net reachability problem. Our construction extends the Petri net encoding of [5] in order to weakly simulate the original PBD system. More precisely, from a PBD reachability problem we compute a Petri net that may contain executions that do not correspond to real computations of the corresponding PBD system. Spurious computations can however be eliminated by enforcing special conditions (e.g. requiring a special set of places to be empty) on the initial and target markings used to encode a PBD reachability problem. It is important to notice that our reduction does not require the additional power provided by Petri nets with transfer arcs.

As a second result, we show that reachability is decidable in PB systems extended with creation rules (PBC systems). We consider here creation rules inspired to those proposed by Martín-Vide, Pãun, and Rodriguez-Paton in the context of P-systems [13]. Our proof exploits structural properties of PBC systems that allow us to reduce the reachability of a target configuration c, to a reachability problem in a Petri net extracted from both the original PBC system and the configuration c. As for PBD systems, this decidability result is a conservative extension of the result for PB systems obtained in [5].

We consider then a model with both dissolution and creation rules (PBDC systems). For this model, we first give a general negative result for the decidability of reachability, and then study a non-trivial subclass in which reachability becomes decidable. Specifically, we first show that it is sufficient to consider three membranes with the same name to encode a reachability problem for a two counter machine [15] as reachability of a PBDC system. We define then a restricted semantics for PBDC systems in which at most only one copy of each type of membrane can be created during a computation. This semantics is inspired to a view of membrane names/types as bounded resources. Under this semantics, we prove the decidability of PBDC reachability via a reduction to Petri net reachability. In the encoding we use special places to identify the membrane structure of the current configuration. The encoding is exponentially more complex than the encoding used for PBD and PBC systems, since it requires the construction of a Petri net where the number of places is equal to number of tree structures that can be built upon a finite and fixed a priori set of membrane names.

As a last analysis, we study the boundedness problem for the aforementioned extensions of PB systems. Specifically, we first show that boundedness is decidable for PBDC systems with restricted semantics. The proof exploits the theory of well-quasi ordering [7]. As a consequence, we obtain the decidability of boundedness for PBD systems. Finally, we prove that boundedness is undecidable in PBC systems. This result is obtained by encoding counter machines as PBC systems. The encoding exploits the possibility of creating several instances of the same membrane to simulate a counter (i.e. the same encoding cannot be applied in the restricted semantics of PBDC systems).

To our current knowledge, these are the first decidability/undecidability results obtained for reachability and boundedness in extensions of PB systems with dissolution and creation rules. Decision problems for qualitative analysis of subclasses of P-systems have been studied, e.g., in [12, 9]. It is important to notice that the decidability of reachability in PBC systems is not in contradiction with the undecidability of boundedness in the same model. Indeed, several other examples of universal models for which the reachability problem is decidable has recently been discovered in the field of process algebra, see e.g. [3, 4].

Plan of the Paper In Section 2 we recall the main definitions of PB systems, Petri nets, and counter machines. In Section 3, 4, and 5 we study the reachability problem for extensions of PB systems resp. with dissolution, creation, and both dissolution and creation rules. In Section 6 we study the boundedness problem for the aforementioned extensions of PB systems. Finally, in Section 7 we address some conclusion and future work.

2 Preliminaries

In this section we recall the main definitions for PB systems with symbol objects taken from [2, 5], Petri nets [17], and counter machines [15]. We first need some preliminary notions. Let $\mathbb{N}$ be the set of positive integers. Consider a finite alphabet Γ of symbols. A multiset over Γ is a mapping $u : \Gamma \rightsquigarrow \mathbb{N}$. For any $a \in \Gamma$, the value $u(a)$ denotes the multiplicity of a in u (the number of occurrences of symbol a in u). We often use a multiset as a string $a_1 \cdot \ldots \cdot a_n$ of symbols, i.e., $a_i \in \Gamma$. Furthermore, we use ϵ to denote the empty multiset, i.e., such that $\epsilon(a) = 0$ for any $a \in \Gamma$. As an example, for $\Gamma = \{a, b, c, d\}$, $a \cdot b \cdot c \cdot c$ represents the multiset u such that $u(a) = u(b) = 1, u(c) = 2, u(d) = 0$. We use $\Gamma^{\otimes}$ to denote the set of all possible multisets over the alphabet Γ. Given two multisets u, v over Γ, we write $u \doteq v$ if $u(a) = v(a)$ for all $a \in \Gamma$, and $u \preceq v$ if $u(a) \leq v(a)$ for all $a \in \Gamma$. Furthermore, we use $\oplus$ and $\ominus$ to denote multiset union and difference, respectively. Specifically, for any $a \in \Gamma$ we have that $(u \oplus v)(a) = u(a) + v(a)$, and $(u \ominus v)(a) = u(a) - v(a)$. We are ready now to formally define a PB system.

2.1 P-systems with Boundary Rules

A PB system [2] with symbol object is a tuple $\Pi = (\Gamma, M, R, \mu_0)$, where

- Γ is a finite alphabet of symbols;
- M is a finite tree representing the *membrane structure* with membrane names taken from a set N,
- R is a finite set of rules,
- μ_0 is the *initial configuration*, i.e., a mapping from membranes (nodes in M) to multisets of objects from Γ.

Rules can be of the following two forms:[3]

$$\begin{array}{ll} (1) \ \text{Internal}: & [_i \ u \rightarrow [_i \ v \\ (2) \ \text{Boundary}: & u \ [_i \ v \rightarrow u' \ [_i \ v' \end{array}$$

where $i \in N$, and $u, u', v, v' \in \Gamma^{\otimes}$ and we assume that at least one between u and u' is not empty.

A configuration μ of a PB system Π is a distribution of objects in Γ in the membranes in M, i.e., a mapping from M to $\Gamma^{\otimes}$. A rule of the form (1) is enabled at μ, if i is a membrane in M and $u \preceq \mu(i)$. Its application leads to a new configurations ν' such that $\nu'(i) = (\nu(i) \ominus u) \oplus v$ and $\nu'(j) = \nu(j)$ for any $j \in N$ s.t. $j \neq i$.

Suppose now that membrane j contains as immediate successor in M membrane i. A rule of the form (2) is enabled at μ, if $u \preceq \mu(j)$ and $v \preceq \mu(i)$. Its application leads to a new configurations ν' such that $\nu'(j) = (\nu(j) \ominus u) \oplus u'$ and $\nu'(i) = (\nu(i) \ominus v) \oplus v'$ and $\nu'(k) = \nu(k)$ for any $k \in N$ s.t. $k \neq i, j$. We say that there is a transition $\mu \Rightarrow \mu'$ if μ' can be obtained from μ by applying a rule in R. A computation with initial configuration μ_0 is a sequence of transitions $\mu_0 \Rightarrow \mu_1 \Rightarrow \ldots$. A configuration is reachable from μ_0 if there exists a sequence of transitions $\mu_0 \Rightarrow \ldots \Rightarrow \mu$

Definition 1 (Reachability). *Given a PB system Π with initial configuration μ_0 and a configuration μ, the* reachability problem *consists in checking if μ is reachable from μ_0.*

Definition 2 (Boundedness). *Given a PB system Π with initial configuration μ_0, the* boundedness problem *consists in deciding if the set of configurations reachable from μ_0 is finite.*

Reachability and boundedness are decidable for PB systems with symbol objects. The proof is based on an encoding PB systems into Petri nets defined in [5].

[3] We consider here a slight generalization of the model in [5] in which we allow any kind of transformation between two membranes.

2.2 Petri nets

A Petri net [17, 18] is a pair (P, T, m_0) where P is a finite set of *places*, T is a finite set of *transitions* ($T \subseteq P^{\otimes} \times P^{\otimes}$), and m_0 is the initial marking. A *transition* t is defined by the pre-set ${}^{\bullet}t$ and by the post-set $t^{\bullet}$, two multisets of places in P (we consider here multisets of places instead of adding weights to arcs). A marking is just a multiset over P. Given a marking m and a place p, we say that the place p contains $m(p)$ *tokens*. A transition t is enabled at the marking m if ${}^{\bullet}t \preceq m$. If it is the case, t can fire and produces a marking m' (usually written $m \xrightarrow{t} m'$) defined as $(m \ominus^{\bullet} t) \oplus t^{\bullet}$. A firing sequence is a sequence of markings $m_0 m_1 \ldots$ such that m_i is obtained from m_{i-1} by firing a transition in T at m_i. Finally, we say that m' is reachable from m_0 if there exists a firing sequence from m_0 passing through marking m'. The Petri net reachability problem has been proved to be decidable in [14, 11].

2.3 Counter Machines

A counter machine [15] consists of a finite set of locations/control states $\ell_1, \ldots, \ell_k$, a finite set of counters $c_1, \ldots, c_m$, and a finite set of instructions. The instruction set consists of the increment, decrement, and zero-test, and the nonzero-test operations on each counter. Each operation has three parameters: the current location, the successor location, and the counter on which it operates. When executed in location ℓ, the increment operation on c_i adds one unit to the current value of c_i and then moves to the successor location ℓ'. When executed in location ℓ, the decrement operation on c_i removes one unit from the current value of c_i and then moves to the successor location ℓ'. When executed in location ℓ, the zero-test on c_i moves to the successor location ℓ' only if c_i has value zero. When executed in location ℓ, the nonzero-test on c_i moves to the successor location ℓ' only if c_i has a value strictly greater than zero. For a fixed initial location ℓ_0, the initial configuration has location ℓ_0 and all counters set to zero. A computation is a sequence of configurations obtained by applying instructions associated to locations. The *if-then-else* instruction *ℓ: if $c_i = 0$ goto ℓ_1 else goto ℓ_2* typically found in the definition of counter machines can be simulated here by two rules associated to the same location ℓ: a zero-test $\ell : c_i = 0$ *goto* ℓ_1 and a nonzero-test. $\ell : c_i > 0$ *goto* ℓ_2. It is well-known that the model of *two counter machines* can simulate a Turing machine, i.e., properties like termination, reachability of a location, and boundedness are all undecidable in this model [15].

3 PB Systems with Dissolution Rules

A PB system with dissolution rules (PBD) provides, in addition to internal and boundary rules, a third kind of rules of the following form:

$$\text{(3) Dissolution :} \quad [_i\ u \rightarrow [_i\ v \cdot \delta$$

where δ is a symbol not in Γ. The intuitive meaning of this rule is that after applying the rule $[_i\ u \rightarrow [_i\ v$ the membrane i is dissolved and its content (including its sub-membranes) is moved to the membrane j that contains i as immediate successor in the current membrane structure. To make the semantics formal, we make the membrane structure part of the current configuration, M_0 being the initial tree. Thus, a configuration is now a pair $c = (M, \mu)$, where M is a tree, and μ is a mapping from nodes of M to $\Gamma^{\otimes}$. Rules of type (1) and (2) operate on a configuration $c = (M, \mu)$ without changing the tree structure M and changing μ as specified in the semantics of PB systems. A dissolution rule like (3) operates on a configuration $c = (M, \mu)$ as follows. For simplicity, we assume that membrane i is not the root of M. Suppose now that i is an immediate successor of j in M. The rule is enabled if $u \preceq \mu(i)$. Its application leads to a new configurations $c' = (M', \nu')$ such that

- M' is the tree obtained by removing node i and by letting all successor nodes of i become successors of j;
- ν' is the mapping defined as $\nu'(j) = \nu(j) \oplus (\nu(i) \ominus u) \oplus v$ and $\nu'(k) = \nu(k)$ for any $k \in M$ s.t. $k \neq i, j$.

Notice that rules of type $(1-3)$ are enabled at $c = (M, \mu)$ only if the membrane i is in current tree M. The definition of sequences of transitions and of reachability problems can naturally be extended to the new type of rules.

3.1 Decidability of Reachability in PBD Systems

In this section we prove that the reachability problem is decidable in PB systems with dissolution rules. We assume here that names of membranes are all different. However, the construction we present can be extended to the general case. The starting point of our construction is the reduction of reachability for PB systems to reachability in Petri nets given in [5]. Let $\Pi = (\Gamma, M, R, \mu_0)$ be a PB system. For each membrane i in M and each symbol $a \in \Gamma$, the Petri net $\mathcal{N}$ associated to Π makes use of place a^i to keep track of the number of occurrences (multiplicity) of objects of type a in i. Transitions associated to internal rules redistribute tokens in the set of places associated to the corresponding membrane. As an example, a rule like $[_i\ a \cdot b\ \rightarrow\ [_i\ c$ is encoded by a Petri net transition that removes one token from place a^i and one token from place b^i and adds one token to place c^i. Boundary rules are modelled by Petri net transitions that work on places associated to pairs of membranes. As an example, if membrane j contains i, a rule like $a\ [_i\ b\ \rightarrow\ b\ [_i\ a$ is encoded by a Petri net transition that removes one token from place a^j and one token from place b^i and adds one token to place b^j and one token to place a^i. For a membrane structure M, a configuration $\mu : M \rightsquigarrow \Gamma^{\otimes}$ is represented by a marking m_μ such that for every node i in M, a^i has k tokens in m iff a has k occurrences in $\mu(i)$. Reachability of a configuration μ is reduced the to reachability of the marking m_μ starting from m_{μ_0} in $\mathcal{N}$.

How to Cope with Dissolution Rules The Petri net encoding of [5] exploits the property that the membrane structure of a PB system is never changed by the application of a rule. This property does not hold anymore for dissolution rules, since they removes nodes from the current membrane structure. Thus, the size of the membrane structure may decrease in a sequence of transitions. Our decidability proof is still based on a reduction to a Petri net reachability problem. Our reduction exploits the property that the number of applications of dissolution rules is bounded a priori by the size of the initial membrane structure M_0.

Specifically, suppose that in M_0 there is an absolute path $i_0, i_1, \ldots, i_k, i$ where i_0 is the root of M_0. In the construction of the Petri net associated to the membrane i we must take into consideration the possibility that each one of the membranes $i_1, \ldots, i_k$ can dissolve during the execution. This means that boundary rules associated to the membrane i should be redirected to the membrane i_j in the path $i_0, i_1, \ldots, i_k$ such that all membranes $i_{j+1}, \ldots, j_k$ are no more present in the current membrane structure. To achieve this, each membrane i comes with an associated flag $present_i/dissolved_i$. Transitions that encode boundary rules on membrane i are conditioned then by the present/dissolved flags of the membranes $i_0, i_1, \ldots, i_k$ in the path leading to i. In other words we need to implement a sort of *switch* that redirects boundary rules to membrane i_j whenever membranes $i_{j+1}, \ldots, j_k$ are all dissolved.

Another problem to solve is related to the transfer of the contents of a membrane to its immediate ancestor in the membrane structure. To simulate this process, our Petri net operates in two different modes. In the *normal* mode the Petri net simulates boundary and internal rules. Suppose now that j is the membrane that contains i in the current tree structure, and that the dissolution rule $[_i u \to [_i v \cdot \delta$ is enabled. We first execute the internal rule $[_i u \to [_i v$. The Petri net then switches to the $dissolving_i$ mode. In this mode all normal operations are blocked. This is achieved by conditioning all transitions associated to normal operations with the *normal* flag. After this step, we activate a set of transitions that move tokens (one-by-one) from i to j. Since in a Petri net it is not possible to test if a place is empty, instead of testing if all objects in i have been moved to j, we add a rule that non-deterministically stops the transfer process. As a last step, we marked membrane i as dissolved and reactivate the normal operating mode. The last step automatically disable internal/boundary/dissolve rules operating on membrane i (they are conditioned by the $present_i$ flag) and redirects boundary rules of any membrane i' contained in i (in the current tree structure) to membrane j.

The non-deterministic termination of the transfer process may lead to incorrect simulations of the original PBD system. Indeed, we could restart in normal mode with a configuration in which a membrane is marked dissolved and some of its objects have not been transferred to its father. This problem can be solved by the following key observation. We first recall that we are interested in reachability of a configuration $c = (M, \mu)$. Thus, by looking at the structure of M_0 and M we can extract the set of nodes D that must be dissolved in a sequence of transitions leading from c_0 to c. Thus, we can tell good simulations from bad ones by imposing the constraint that, in the marking m_μ modelling the target configuration μ, every place associated to a membrane in D is empty. As a final remark, notice that all information needed in the encoding (nodes and paths in M_0) can statically be extracted from the description of the PBD system and from the initial configuration c_0.

Formal Definition of the Petri Net Encoding

Assume a PBD system $\Pi = (\Gamma, M_0, R, \mu_0)$, where $\Gamma = \{a_1, \ldots, a_m\}$, and M_0 has the membranes with names in $N = \{n_0, n_1, \ldots, n_k\}$, $n_0 \in N$ being the root node. Given a membrane i, let $path(i)$ be the sequence of nodes in the (unique) path from n_0 to i in M_0.

We define the Petri net $\mathcal{N}$ encoding Π in several steps. First of all we assume that $\mathcal{N}$ has at least the places $normal, dissolving_1, \ldots, dissolving_k$ that we use to determine the simulation mode as described in the previous paragraphs. We assume here that $normal$ contains one token iff $dissolving_i$ is empty for all $i : 1, \ldots, k$, and $dissolving_i$ contains one token iff $normal$ as well $dissolving_j$ are empty for any $j \neq i$. Furthermore, for each membrane i, the Petri net $\mathcal{N}$ has a place $present_i$ and a place $dissolved_i$ and, for any $a \in \Gamma$, a place a^i.

Notation: In the rest of the paper given a multiset of objects u and a membrane i we use $\pi_i(u)$ to denote the multiset of places in which, for each $a \in \Gamma$, a^i has the same number of occurrences as those of a in u.

Internal Rules An internal rule $r = [_i u \rightarrow [_i v$ is encoded by a transition t_r that satisfies the following conditions. The pre-set of t_r contains place $normal$ (normal mode), $present_i$ (membrane i is still present), and the multiset of places $\pi_i(u)$. The post-set of t_r contains $normal, present_i$ and the multiset of places $\pi_i(v)$. Thus, the only difference with the encoding of PB system is the condition on the $normal$ and $present_i$ flags (in normal mode internal rules are enabled only when the membrane is not dissolved).

Boundary Rules Let $path(i) = (n_0, n_1, \ldots, n_q, i)$ with $q \geq 0$. A boundary rule $r = u[_i v \rightarrow u'[_i v'$ is encoded by a set $B_r = \{b_r^{n_0}, \ldots, b_r^{n_q}\}$ of transitions. The pre-set of transition $b_r^{n_j}$ contains places $normal$ and $present_i$ together with the set of places $D^{n_j} = \{present_{n_j}, dissolved_{n_{j+1}}, \ldots, dissolved_{n_q}\}$, and the multisets $\pi_{n_j}(u)$ and $\pi_i(v)$. The post-set contains places $normal$ and $present_i$ together with the set of places D^{n_j} defined for the pre-set and the multisets $\pi_{n_j}(u')$ and $\pi_i(v')$. The pre-condition D^{n_j} allows us to select the membrane that is the immediate ancestor of i in the current configuration, i.e., a membrane $n_j \in path(i)$ that is not dissolved and such that all the intermediate membranes between n_j and i in $path(i)$ are dissolved. Notice that, by the assumptions we made on the $normal/dissolving$ and $present/dissolved$ flags, in normal mode one and only one rule in B_r can be enabled at a given configuration (represented by a marking).

Dissolution Rules Let $path(i) = (n_0, n_1, \ldots, n_q, i)$ with $q \geq 0$. Consider a dissolution rule $r = [_i u \rightarrow [_i v \cdot \delta$. We first model the internal rule by the transition

s_r. The pre-set of s_r contains the places *normal* and $present_i$ and the multiset $\pi_i(u)$. The post-set contains the place $dissolving_i$ and the multiset $\pi_i(v)$.

We then model the transfer of the contents of membrane i to its current immediate ancestor via a set of transitions $S_r^a = \{s_a^{n_0}, \ldots, s_a^{n_q}\}$ for each $a \in \Gamma$. The pre-set of transition $s_a^{n_j}$ contains places $dissolving_i$ (i is dissolving) and a^i (the source of a token to be transferred) together with the set of places D^{n_j} defined in the case of boundary rules. The post-set contains places $dissolving_i$ and a^j (the destination of a transferred token), and the set D^{n_j}.

Finally, we add a transition d_r^i to stop the transfer of tokens and to switch the operating mode back to *normal*. The pre-set of d_r^i contains the place $dissolving_i$ and its post-set contains the places *normal* and $dissolved_i$. Notice that the simulation phase of a dissolution rule for membrane i can be activated only if $present_i$ is not empty. This implies that once the $dissolving_i$ flag is reset (i.e. the mode goes back to *normal*) it cannot be set in successive executions (a membrane can dissolve at most once).

Places, Transitions and Configurations The Petri net $\mathcal{N}$ is built by taking the union of the places and transitions used in the encoding described before. Let M be a membrane structure with a subset of the nodes in M_0 (initial structure of Π). A configuration $c = (M, \mu)$ is encoded by a marking m_c in which there is one token in *normal*, one token in $present_i$ for each membrane i in M, and one token in $dissolved_j$ for each j not in M. Furthermore, for each membrane i in c and $a \in \Gamma$, place a^i has as many tokens as the number occurrences of a in $\mu(i)$. All the remaining places in $\mathcal{N}$ ($dissolving_i$ for $i : 0, \ldots, k$ and $present_j$ for all membranes j not in M) are empty.

Reachability Problem By construction of $\mathcal{N}$, it is immediate to see that if there is a sequence of transitions from $c_0 = (M_0, \mu_0)$ to $c = (M, \mu)$ passing through the configurations $c_1, \ldots, c_v$ then there is a firing sequence from m_{c_0} to m_c passing through the markings $m_{c_1}, \ldots, m_{c_v}$. Such a firing sequence is obtained by completing all transfers of objects required by the simulation of each dissolution rule (i.e. after the simulation of the dissolution of membrane i, the normal mode is reactivate only when the places associated to the objects contained in i are all empty). Vice versa, suppose there exists a firing sequence from m_{c_0} to m_c. We first notice that only the markings in which the place *normal* is not empty correspond to configurations of the original PBD system. Furthermore, suppose that during the simulation of the dissolution of membrane i, the transfer of objects is stopped when some of the places associated to objects in i are not empty. Let m be the resulting marking. Now we notice that the first step of the simulation of dissolution is to set the $present_i$ flag to false. This implies that in the marking m place $present_i$ is empty, while there exists $a \in \Gamma$ such that a^i is not empty (some token has not been transferred). It is easy to check that if m has these two properties, for any marking m' derived from m by applying transitions of $\mathcal{N}$, the content of the place a^i in m' is the same as in m. Indeed, transitions that simulate internal, boundary and dissolution rules operating directly on i are no more enabled (the condition $present_i$ fails). Furthermore, a dissolution rule on a membrane j nested into i in M_0 cannot transfer tokens to i since $dissolved_i$ is checked when searching for the father of j in the current tree structure. In other words, if the simulation of a dissolution rule is not correctly executed, then there exists at least one non-empty place a^i for a dissolved membrane i. By definition, however, m_c is the marking in which all places associate to dissolved membranes are empty. Thus, if m_c is reachable from m_{c_0} then the corresponding firing sequence corresponds to a real computation in $\mathcal{N}$. Thus, we have that the reachability of a configuration $c = (M, \mu)$ in Π can be encoded as the reachability of the marking m_c in $\mathcal{N}$ from m_{c_0}. From the decidability of reachability in Petri nets, we obtain the following theorem.

Theorem 1. *Reachability is decidable in PBD systems.*

This result extends (and is consistent with) the Petri net encoding and the decidability result for reachability in PB systems of [5].

4 PB System with Creation

In this section we consider an extension of PB systems inspired to the membrane creation operation studied in [13]. Let N be a possibly infinite list of membrane names. A PB system with creation rules (PBC) provides, in addition to internal and boundary rules, a third kind of rules of the following form:

$$(4)\ \text{Creation :} \quad a \ \rightarrow [_i \ v \ ]_i$$

where $a \in \Gamma$, $v \in \Gamma^{\otimes}$, and $i \in \mathsf{N}$. The intuitive meaning of this rule is that after applying the rule $a \rightarrow [_i v]_i$ inside a membrane j, object a is replaced by the new membrane i containing the multiset of objects v. To make the semantics formal, we assume that membrane structures are trees whose nodes are labelled with names in N. Furthermore, we make both the set of used names and the membrane structure part of the current configuration, $N_0 \subseteq \mathsf{N}$ being the initial set of used names, and M_0 being the initial tree defined over N_0. Thus, a configuration is now a triple $c = (N, M, \mu)$ where N is a set of names, M is a tree with nodes labelled in N, and μ is a mapping $M \rightsquigarrow \Gamma^{\otimes}$. Rules of type (1) and (2) operate on a configuration $c = (N, M, \mu)$ without changing N and M. A creation rule like (4) operates on a configuration $c = (N, M, \mu)$ as follows. Suppose that n is a node in M. The rule is enabled if $a \in \mu(n)$. Its application leads to a new configurations $c' = (N', M', \nu')$ such that

- $N' = N \cup \{i\}$;
- M' is the tree obtained by adding a new node m labelled by i as a successor of node n;
- ν' is the mapping defined as $\nu'(n) = \nu(n) \ominus a$, $\nu'(m) = v$, and and $\nu'(p) = \nu(p)$ for $p \neq m, n$, $p \in N$.

Notice that rules of type (4) can be applied in any membrane. Indeed, the only precondition for the application of rule 4 is the existence of object a in a membrane. Furthermore, such an application may create different nodes with the same membrane name.

The reachability problem can naturally be reformulated for the extended semantics of rules. Specifically, it consists in checking whether a given target configuration c is reachable from the initial configuration c_0.

In presence of creation rules the membrane structure can grow in an arbitrary manner both in width and depth. Notice that in our model we distinguish nodes from membrane names. Thus, different nodes may have the same name. As a simple example, consider the rule $a \to [_1 a]_1$ in a system with a single membrane $[_0 a]_0$. The evolution of this PBC system may lead to membrane structures of arbitrary nesting level, e.g.,

$$[_0[_1 a]_1]_0 \quad [_0[_1[_1 a]_1]_1]_0 \quad [_0[_1[_1[_1 a]_1]_1]_1]_0 \quad \dots$$

Now consider the rules $[_0 a \to [_0 a \cdot a$ and $a \to [_1 b]_1$. Then the membrane $[_0 a]_0$ can generate membrane structures of arbitrary width, e.g.,

$$[_0[_1 b]_1]_0 \quad [_0[_1 b]_1[_1 b]_1]_0 \quad [_0[_1 b]_1[_1 b]_1[_1 b]_1]_0 \quad \dots$$

Despite of these powerful features of PBC systems, the reachability problems can still be decided by resorting to an encoding into Petri net reachability as explained in the next section.

4.1 Decidability of Reachability in PBC Systems

Differently from the encoding defined for PBD systems in the previous section, the encoding needed here is function of both the initial and the target configuration. Indeed, since PBC rules can only add new nodes, to decide if a configuration $c = (N, M, \mu)$ is reachable from c_0 we can restrict our attention to membrane structures of size comprised between the size of M_0 and the size of M and with (possibly repeated) labels in N.

Actually, we can make some simplification that allows us to build a Petri net by considering only the target configuration M. Indeed, as shown in [13], with creation rules we can safely consider initial configurations with only the root membrane (we can always add a finite number of creation rules to generate any initial configuration in a preliminary phase of the computation).

So let Π be a PBC system with initial configuration $c_0 = (N_0, M_0, \mu_0)$ where M_0 is a single node labelled n_0. Consider now a target configuration (N, M, μ) where $N = \{n_0, \dots, n_k\}$ and M has m nodes with $k \leq m$ and the root node of M is labelled n_0.

Starting from Π and c we build the Petri net $\mathcal{N}$ described next. For each node n in M, the Petri net $\mathcal{N}$ has places $used_n$ and $notused_n$ (used as one flip-flop), and a^n for each $a \in \Gamma$ (to model the content of membrane n). We assume that $used_n$ is not empty iff the membrane has been created and it is in use, and $notused_n$ is not empty iff the membrane has still to be created. PBC rules are modelled as follows.

Internal Rules For each node n in M with label i, an internal rule $r = [_i u \rightarrow [_i v$ is encoded by a transition t_r^n that satisfies the following conditions: The pre-set contains place $used_n$ together with multiset $\pi_n(u)$; The post-set contains $used_n$ together with multiset $\pi_n(u)$.
The differences with the encoding of PB/PBD systems is the condition on the $used_n$ flag and the fact that we work on nodes of membrane structures and not directly on membrane names (as said before two different nodes may have the same name). The pre-condition on $used_n$ is needed in order to enable rules operating on node n only after the corresponding creation rule has been fired.

Boundary Rules For each node m in M that has an immediate successor n with label i, a boundary rule $r = u[_i v \rightarrow u'[_i v'$ is encoded by a transition $b_r^{m,n}$ that satisfies the following conditions. The pre-set contains places $used_n$ and $used_m$ together with the multisets $\pi_m(u)$ and $\pi_n(v)$. The post-set contains places $used_n$ and $used_m$ together with the multisets $\pi_m(u')$ and $\pi_n(v')$. Notice that, differently from the encoding used in PBD, in PBC we do not have to consider paths in the membrane structure M.

Creation Rules For each node m in M that has an immediate successor n with label i, a creation rule $r = a \rightarrow [_i v]_i$ is encoded by a transition $c_r^{m,n}$ that satisfies the following conditions. The pre-set contains places $used_m$, a^m and $notused_n$. The post-set contains places $used_m$ and $used_n$, together with the multiset $\pi_n(v)$

Places, Transitions and Configurations The Petri net $\mathcal{N}$ is built by taking the union of the places associated to each membrane and the union of the set of transitions used to encode internal, boundary and creation rules described before. A configuration $c' = (N', M', \mu')$ is encoded by a marking $m_{c'}$ in which for each node n in M' there is one token in $used_n$, and for each $a \in \Gamma$, a^n has as many tokens as the number occurrences of a in $\mu(n)$. Furthermore, for each node n' in M that do not occur in M', we put a token in $notused_{n'}$. All the remaining places are empty.

Reachability Problem By construction of $\mathcal{N}$, it is immediate to see that there is a sequence of transitions from $c_0 = (N_0, M_0, \mu_0)$ to $c = (N, M, \mu)$ passing through the configurations $c_1, \ldots, c_v$ if and only if there is a firing sequence from m_{c_0} to m_c passing through the markings $m_{c_1}, \ldots, m_{c_v}$. The creation of a new node n corresponds to the activation of the part of the Petri net $\mathcal{N}$ that models node n. Since nodes are created in "cascade", a node m is created only after all ancestors have been created. This property is ensured by the condition on the *used* flag inserted in the transitions modelling creation rules.

Following from the decidability of reachability in Petri nets, we obtain the following theorem.

Theorem 2. *Reachability is decidable in PB systems with creation rules.*

This result extends (and is consistent with) the Petri net encoding and the decidability result for reachability in PB systems of [5].

5 PB Systems with Dissolution and Creation

In this section we consider an extension of PB systems with both dissolution and creation rules (PBDC systems). The semantics is obtained in a natural way by adapting the semantics of dissolution rules to membrane structures with labelled nodes. More precisely, a dissolution rule applied to a configuration (N, M, μ) modifies M and μ as specified in Section 3 while it does not modify N, i.e., the set N of used names can only grow monotonically. Notice that the reachability problem for PBDC systems allows to determine if, for a PBDC system Π, it is possible from the initial configuration to build a membrane structure M with a mapping μ that associates multisets of objects to nodes, whatever the set of names N is. Indeed, the set of possible names for membranes that appear in executions of Π is determined by the creation rules of the PBDC system (and its initial configuration). Hence, the number of possibilities for N is finite and the problem can be reduced to a finite number of reachability problems.

In presence of both creation and dissolution the membrane structure can change in an arbitrary manner. As a simple example consider the rules $a \to [_1 a]_1$, and $[_1 a \to [_1 a \cdot \delta$ in a system with a single membrane $[_0 a]_0$. The evolution of this PB system may lead to membrane structures that grow and shrink in an arbitrary way as in the sequence:

$$[_0[_1 a]_1]_0 \quad [_0[_1[_1 a]_1]_1]_0 \quad [_0[_1 a]_1]_0 \quad \ldots$$

This feature gives additional power to PB systems. Indeed, we can reduce the reachability problem for two counter machines (known to be undecidable) to reachability in a PBDC system. For this reduction it is enough to consider dissolution and creation rules working on three membranes with the same name. Specifically, consider a system with initial configuration

$$c_0 = [_0 \; s_0 \; [_0 \; c_1 \;]_0 \; [_0 \; c_2 \;]_0 \;]_0$$

here s_0 represents the initial control state of a two counter machine. Membrane $[_0 c_i]_0$ is used to represent counter c_i with value zero for $i : 1, 2$. Counter c_i with value k is represented by the membrane $[_0 c_i \cdot u]_0$ where u is the multiset with k occurrences of a special object a.
The increment of counter c_i in control state s and update to control state s' is encoded by the boundary rule

$$s \; [_0 \; c_i \to s' \; [_0 \; c_i \cdot a$$

for $i : 1, 2$.
The decrement of counter c_i in control state s and update to control state s' is encoded by the boundary rule

$$s \; [_0 \; c_i \cdot a \to s' \; [_0 \; c_i$$

for $i : 1, 2$.
The zero test on counter c_i in control state s and update to control state s' is

simulated by three rules. We first move to an auxiliary state aux_s and dissolve the membrane with label i.

$$s\ [_0\ c_i \to aux_s\ [_0\ \epsilon \cdot \delta$$

where ϵ is the empty multiset. We then create a new empty instance of the same membrane containing the objects c_i and out'_s via the rule

$$aux_s \to [_0\ c_i \cdot out_{s'}\ ]_0$$

Finally, we use a boundary rule to move to the next state s'

$$[_0\ c_i \cdot out_{s'} \to s'\ [_0\ c_i$$

We assume here that no other rule uses aux_s and $out_{s'}$. The effect of the execution of these three rules is that of moving the contents of the counter on which the zero-test is executed to the top level membrane 0. Indeed, the membrane containing c_i is first dissolved and then re-created. If the zero-test is executed when a counter is not zero, some object will remain inside membrane 0 in all successive configurations. This feature can be used to distinguish good simulations from bad ones. Specifically, let us consider the reachability problem for a two counter machine in which the initial and the target configurations both coincide with the configuration with a given control state s and both counters set to zero. This problem can be expressed as the reachability of the configuration c_0 from c_0. Thus, the following property holds.

Theorem 3. *Reachability is undecidable in PBDC systems in which configurations have at most three different membranes with the same name.*

5.1 PBDC Systems with Restricted Semantics

As a final analysis, we consider a restricted semantics for PBDC systems in which newly created membranes must be assigned fresh and unused names. In other words we assume that creation rules can be applied at most once for each *type* of membrane. Another possible view is that membrane names are themselves resources that can be used at most once.

Formally, assume a configuration $c = (N, M, \mu)$. Suppose that n is a node in M. In the restricted semantics, the creation rule (4) is enabled if $a \in \mu(n)$ and $i \notin N$, i.e., the name i is *fresh*. Its application leads to a new configurations $c' = (N', M', \nu')$ such that $N' = N \cup \{i\}$, M' is the tree obtained by adding a new node m labelled i as a successor of node n, and ν' is the mapping defined as $\nu'(n) = \nu(n) \ominus a$, $\nu'(m) = v$, and $\nu'(p) = \nu(p)$ for $p \neq m, n$, $p \in N$).

Since with creation rules in the style of [13] the set of rules operating on membranes is fixed and known a priori, we can assume that the number of distinct names is finite (it corresponds to the set of names occurring in internal, boundary, dissolution and creation rules and in the initial configuration). This restriction yields the following key observation.

Observation 1. If the set of possible membrane names N is finite and every name in N can be used only once, then starting from a configuration with a single membrane, the number of distinct membrane structures that we can generate is finite. Every such membrane structure has at most $|\mathsf{N}|$ nodes.

This property does not imply that the number of configurations is finite. Indeed, there are no restrictions on creation and deletion of *objects* inside membranes. As an example, the PBDC system with the internal rule $[_0 a \to [_0 a \cdot a$ and the initial membrane $[_0 a]_0$ generates an infinite set of configurations (membrane 0 with any number of repetitions of object a).

The aforementioned property can be used to show that reachability is decidable in PBDC Systems with restricted semantics. Let Π be a PBDC system defined over a finite set of names Λ. Suppose that Λ has cardinality K. Furthermore, assume that the initial configuration $c_0 = (N_0, M_0, \mu_0)$ is such that M_0 is a single node. We first build the set Θ of all possible membrane structures with *at most* K nodes labelled with distinct labels taken from Λ. As an example, if $\Lambda = \{0, 1, 2\}$, then we will consider all trees with at most three nodes and such that each node has a distinct label taken from Λ, i.e., $[_0\ ]_0$, $[_1\ ]_1$, $[_1\ [_0\ ]_0\ ]_1$, $[_2\ [_0\ ]_0\ [_1\ ]_1]_2$, and so on. Notice that for a fixed membrane structure T we can always determine the immediate ancestor j of a node i (if it exists) at static time.

Starting from Π and Θ, we now define a Petri net $\mathcal{N}$ that satisfies the following conditions. First of all, we associate a place T to each membrane structure $T \in \Theta$. We assume that only one of such places can be non empty during the simulation of the restricted semantics. A non-empty place $T \in \Theta$ corresponds to the current membrane structure. Furthermore, for each $i \in \Lambda$ we add to $\mathcal{N}$ places $used_i$ and $notused_i$ (to model freshness of name i), and, for each $a \in \Gamma$, place a^i (to model the content of membrane i in the current membrane structure). Notice that since names are used at most once, we can safely confuse nodes of membrane structure with their labels (each node has a different label in Λ). PBDC rules are modelled as the finite set of transitions in $\mathcal{N}$ defined as follows.

Internal Rules For each membrane structure $T \in \Theta$ with a membrane i, an internal rule $r = [_i u \to [_i v$ is encoded by a transition t_r^T that satisfies the following conditions. The pre-set contains the places T (the current membrane structure) and $used_i$ (i is in use) together with multiset $\pi_i(u)$. The post-set contains places T and $used_i$ together with multiset $\pi_i(v)$. Thus, only the internal rules defined on the current membrane structure are enabled.

Boundary Rules For each membrane structure $T \in \Theta$ with a membrane j with immediate successor i, a boundary rule $r = u[_i v \to u'[_i v'$ is encoded by a transition $b_r^{T,i,j}$ that satisfies the following conditions. The pre-set contains places T, $used_i$, $used_j$, and the multisets $\pi_i(u)$ and $\pi_i(v)$. The post-set contains places T, $used_i$, $used_j$, and the multisets $\pi_i(u')$ and $\pi_i(v')$. Thus, only boundary rules defined on the current membrane structure are enabled.

Creation Rules For each $T \in \Theta$ such that i does not occur in the set of names in T (the side condition that ensures the freshness of generated membrane names), and for each name j occurring in T, a creation rule $r = a \rightarrow [_i v]_i$ is encoded by a transition $c_r^{T,i,j}$ that satisfies the following conditions. The pre-set contains places T, $used_j$, $notused_i$, and a^j. The post-set contains places $used_i$ and $used_j$, the multiset $\pi_i(v)$, and the place $T_{j+i} \in \Theta$ associated to the membrane structure obtained from T by adding a new node labelled i as immediate successor of j.

Dissolution Rules For each membrane structure $T \in \Theta$ with a membrane j with immediate successor i, a dissolution rule $r = [_i u \rightarrow [_i v \cdot \delta$ is encoded by the following set of transitions. We first define a transition $c_r^{T,i,j}$ that starts the dissolution phase of node i. The pre-set of $c_r^{T,i,j}$ contains places T, $used_i$, $used_j$, and the multiset $\pi_i(u)$. The post-set contains the place $dissolve^{T,i,j}$, and, the multiset $\pi_i(v)$. Notice that, by removing a token from the place T, we automatically disable all transitions not involved in the dissolution phase (i.e. T plays the role of flag *normal* used for simulating dissolution rules in PBD systems). Now, for each $a \in \Gamma$, we model the transfer of the content of node i to node j via a transition $m_r^{T,i,j,a}$ that satisfies the following conditions: The pre-set contains the places $dissolve^{T,i,j}$ and a^i (the source of a token to be transferred). The post-set contains the places $dissolve^{T,i,j}$ and a^j (the destination of a transferred token). Finally, let T_{j-i} be the membrane structure obtained by T by removing membrane i and moving all of its sub-membranes into membrane j. Then, we add transition $d_r^{T,i,j}$ to non-deterministically stop the transfer of tokens and to update the membrane structure to T_{j-i}, i.e., the pre-set of this transition contains the place $dissolve^{T,i,j}$ and its post-set contains the places T_{j-i}, $used_i$ and $used_j$. Notice that name i remains marked as *used* after dissolving the corresponding membrane (i.e. it cannot be used in successive creation rules).

Places, Transitions and Configurations The Petri net $\mathcal{N}$ is built by taking the union of the places and transitions used to encode internal, boundary, creation and dissolution rules described before.
A generic configuration $c = (N, M, \mu)$ is encoded by a marking m_c in which: there is one token in the place associated to the membrane structure M, one token in $used_i$ for each $i \in N$, one token in $notused_i$ for each $i \in \Lambda \setminus N$, and, for each i that occurs in M and for each $a \in \Gamma$, as many tokens in a^i as the number of occurrences of object a in $\mu(i)$. All other places are empty.

Reachability Problem Notice that after a membrane with name i is introduced by a creation rule (i.e. the place $unused_i$ is emptied while one token is put in $used_i$), no other membranes with the same name can be created (there is no rule that puts a token back to $unused_i$). The membrane i however can dissolve in a successive transition, i.e. in a target configuration $used_i$ can be non-empty (i.e. $i \in N$), even if i does not occur in the current membrane structure. Also notice that in m_c we enforce all places associated to membranes not occurring in M to be empty. The combination of these two properties allows us to distinguish good simulations (i.e. in which after the application of dissolution rules all tokens are transferred to the father membrane) from bad ones (some tokens are left in a place a^i, $used_i$ is non empty, but i is no more in the current membrane structure). Following from this observation and from the construction of $\mathcal{N}$, we have that $c = (N, M, \mu)$ is reachable from c_0 if and only if the marking m_c is reachable from m_{c_0}.

Following from the decidability of reachability in Petri nets, we obtain the following theorem.

Theorem 4. *Reachability is decidable in PBDC systems with restricted semantics.*

6 Boundedness Problem for Extended PB Systems

In [5] Dal Zilio and Formenti exploit the Petri net encoding used for deciding reachability to prove that boundedness is decidable too for PB systems with symbol objects. In this section we investigate the boundedness problem for the different extensions of PB systems proposed in the present paper. In particular, we show that the boundedness problem is decidable for PBDC systems with restricted semantics where a membrane name can be used at most once. This result implies that boundedness is decidable for PB systems with dissolution (and standard semantics). Finally, we show undecidability of the boundedness problem for PB systems with creation and standard semantics.

6.1 Boundedness for PBDC Systems with Restricted Semantics

To prove decidability of boundedness in PBDC systems with restricted semantics, let us first define the following partial order $\sqsubseteq$ over configurations. Assume two configurations $c_1 = (N_1, M_1, \mu_1)$ and $c_2 = (N_2, M_2, \mu_2)$. We define $c_1 \sqsubseteq c_2$ if and only if $N_1 = N_2$ and $M_1 = M_2$ (i.e. c_1 and c_2 have the same tree structure), and $\mu_1(n) \leq \mu_2(n)$ (the multiset associated to n in c_1 is contained in that associated to n in c_2) for all node n in M_1. If we fix an upper bound on the number of possible nodes occurring in a membrane structure along a computation, then $\sqsubseteq$ has the following property.

Proposition 1. *Fixed a $k \in \mathbb{N}$, for any infinite sequence of configurations $c_1 c_2 \ldots$ with membrane structure of size at most k, there exist positions $i < j$ such that $c_i \sqsubseteq c_j$ (i.e. $\sqsubseteq$ is a* well-quasi ordering*).*

The proof is a straightforward application of composition properties of well-quasi ordering, see e.g. [1]. Now assume an infinite computation $c_0 = (N_0, M_0, \mu_0) c_1 = (N_1, M_1, \mu_1) \ldots$ of a PBDC system with restricted semantics. From Observation 1 it follows that for all $i \geq 0$ the number of nodes in M_i is bounded by the number of possible names. Hence, by Prop. 1 we know that there exist positions $i < j$ such that $c_i \sqsubseteq c_j$. Furthermore, if $c_i \sqsubseteq c_j$ and $c_i \neq c_j$, then $N_i = N_j$, $M_i = M_j$, and $\mu_i \prec \mu_j$. Thus, the transition sequence σ from c_i to c_j does not modify the membrane structure but strictly increases the number of objects contained at each of its node. This implies that the application of σ can be iterated starting from c_j, leading to a infinite strictly increasing, w.r.t $\sqsubseteq$, sequence of configurations.

As a consequence of these properties, the boundedness problem for PBDC systems can be decided by building a computation tree T such that: the root node n_0 of T is labelled by the initial configuration c_0, if $n_0, \ldots, n_k$ is a path in T such that n_i is labelled with c_i $i : 0, \ldots, k$, and for all $i : 0, \ldots k-1$ $c_i \neq c_k$ and $c_i \not\sqsubseteq c_k$ then we add a node n' labelled c' as successor of n_k if and only if $c_k \Rightarrow c'$. Furthermore, the PBDC system is not bounded if and only if there exists a leaf n labelled with c and a predecessor n' of n labelled with c' in T such that $c' \sqsubseteq c$. Since, $\sqsubseteq$ is a decidable relation and, the tree T is finite (by Observation 1 and Prop. 1), the following property then holds.

Theorem 5. *The boundedness problem is decidable for PBDC systems with restricted semantics.*

From Theorem 5, we know that boundedness is decidable for PB systems (consistently with the result in [5]) and for PB systems with dissolution (they form a subclass of PBDC systems where the restricted and standard semantics coincides).

6.2 Boundedness Problem for PBC Systems

The boundedness problem turns out to be undecidable for PBC systems with standard semantics in which there is no limit on the number of instances of a given type of membranes that can be created during a computation. The proof is based on a reduction of counter machines with increment, decrement and zero-test to PBC systems. The idea is to use nested membranes to model the current value of a counter. For instance,

$$[_1 used\ [_1\ used\ [_1\ unused\ [_1\ end\ ]_1]_1]_1]_1$$

can be used to encode counter c_1 with value 2 (the number of occurrences of symbol *used*). Hence a configuration of a two counter machine with both counters set to zero is encoded as a configuration of the form

$$[_0\ \ell\ [_1\ unused\ [_1\ \ldots [_1\ end\ ]_1\ \ldots]_1]_1 [_2\ unused\ [_2 \ldots [_2\ end\ ]_2\ \ldots]_2\ ]_2\ ]_0$$

where ℓ is a symbol corresponding to the current location of the two counter machine, and membrane with name i encodes counter i for $i : 1, 2$. Increment of counter i in location ℓ with ℓ_1 as successor location is simulated by replacing the first *unused* symbol encountered when descending the tree from the membrane 0 with the symbol *used*. This is implemented by the following set of rules that descend the structure of a membrane of type i in search for the first occurrence of symbol *unused*:

$$\begin{array}{l}
\ell\ [_i\ unused\ \rightarrow\ \ell_1\ [_i\ used\\
\ell\ [_i\ used\ \rightarrow\ \ell'_i\ [_i\ down\\
down\ [_i\ used\ \rightarrow\ down\ [_i\ down\\
down\ [_i\ unused\ \rightarrow\ up\ [_i\ used\\
down\ [_i\ up\ \rightarrow\ up\ [_i\ used\\
\ell'_i\ [_i\ up\ \rightarrow\ \ell_1\ [_i\ used
\end{array}$$

where $\ell'_i, down, up$ are auxiliary symbols (ℓ'_i is blocking for the other counter(s), *down* propagate the search down in the tree, *up* propagate the success notification up in the tree). Furthermore, if the current tree has no membrane containing the *unused* symbol (i.e. all membranes contain *used*) then a new membrane is created by applying the following additional rules:

$$\begin{array}{l}
down\ [_i\ end\ \rightarrow\ down\ [_i\ create\\
create\ \rightarrow\ [_i\ exit \cdot end\ ]_i\\
\epsilon\ [_i\ exit\ \rightarrow\ up\ [_i\ \epsilon
\end{array}$$

where ϵ denotes the empty multiset. Decrement is simulated by changing the last occurrence of *used* encountered by descending the membrane tree from the root into symbol *unused*. We can safely assume here that the counter is non-zero, i.e., that there is at least one membrane with *used* object. The rules that implement decrement are defined as follows.

$$\begin{array}{l}
\ell\ [_i\ used\ \rightarrow\ \ell'_i\ [_i\ used_1\\
used_1\ [_i\ used\ \rightarrow\ used_1\ [_i\ used_1\\
used_1\ [_i\ unused\ \rightarrow\ used_2\ [_i\ unused\\
used_1\ [_i\ end\ \rightarrow\ used_2\ [_i\ end\\
used_1\ [_i\ used_2\ \rightarrow\ used_3\ [_i\ unused\\
used_1\ [_i\ used_3\ \rightarrow\ used_3\ [_i\ used\\
\ell'_i\ [_i\ used_2\ \rightarrow\ \ell_1\ [_i\ unused\\
\ell'_i\ [_i\ used_3\ \rightarrow\ \ell_1\ [_i\ used
\end{array}$$

where $\ell'_i, used_1, used_2, used_3$ are auxiliary symbols, $used_1$ is used to mark nodes during the downward search (for *unused*), $used_2$ is used to mark the *used* node to be replaced by *unused*, and $used_3$ is used to replace nodes marked with $used_1$ with *used* during the return from the search. The zero-test on counter c_i in location ℓ with successor ℓ_1 can be implemented by testing if all objects in membranes i are *unused* (or *end*). Note that increment, resp. decrement, of c_i is encoded by a top-down traversal of membranes i until reaching a membrane containing an object *unused*, resp. *used*, which is then replaced by *used*, resp. *unused*. Furthermore, each membrane contains one object. Hence, no membrane i contain a *used* symbols if and only if the top level membrane i has object *unused* or *end*. This can be checked with the following rules:

$$\begin{array}{l}
\ell\ [_i\ unused\ \rightarrow\ \ell_1\ [_i\ unused\\
\ell\ [_i\ end\ \rightarrow\ \ell_1\ [_i\ end
\end{array}$$

Similarly, testing that c_i is not equal to zero in location ℓ with successor ℓ_1 amounts to check that the top level membrane i has object *used*, i.e.,

$$\ell\ [_i\ used\ \rightarrow\ \ell_1\ [_i\ used$$

By construction, we directly have that a two counter machine is bounded (Are its counters bounded by a constant $k \in \mathbb{N}$?) if and only if its encoding into PBC systems is bounded. Since boundedness is undecidable for two-counter machines with zero-test, we obtain the following negative result.

Theorem 6. *The boundedness problem is undecidable for PBC systems.*

Remark The proof of Theorem 6 shows that PBC systems can simulate two-counter machines by means of membrane structures with unbounded depth. Let us now go back to the reachability problem of a target configuration μ. Since PBC rule never remove membranes, the target configuration μ gives us an upper bound on the size of membrane structures that may occur in a sequence of transition leading to it. Thus, the set of possible membrane structures than we have to consider to solve a reachability problem is always finite (Notice that this not imply that we have to consider a finite number of configurations). In other words, for reachability problems, we do not have to deal with the full computational power of PBC systems. For this reason, the undecidability of boundedness proved in Theorem 6 is not in contradiction with the decidability of reachability proved in Theorem 2. Similar results obtained for fragments of process calculi [3, 4] seem to indicate that, in general, the decidability of reachability cannot be use to give an estimation of the expressive power of a computational model.

7 Conclusions

In this paper we have investigated the decidability of reachability and boundedness in extensions of PB systems with rules that dynamically modify the tree structure of membranes. We conjecture that some of the positive results presented here can be extended to PB systems with some form of movement and with dynamic generation of membrane names. We plan to investigate these problems in future work.

References

1. P. Aziz. Abdulla and A. Nylén. Better is Better than Well. On Efficient Verification of Infinite-State Systems. LICS 2000: 132-140.
2. F. Bernardini, V. Manca P Systems with Boundary Rules WMC 2002: 107-118.
3. N. Busi and G. Zavattaro. Deciding Reachability in Mobile Ambients. ESOP 2005: 248-262.
4. N. Busi and G. Zavattaro. Deciding Reachability in Boxed Ambients. ICTCS 2005: 143-159.
5. S. Dal Zilio and E. Formenti. On the Dynamics of PB Systems: A Petri Net View. WMC 2003: 153167.

6. C. Dufourd, A. Finkel, Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. ICALP 1998: 103-115.
7. A. Finkel, Ph. Schnoebelen. Well-structured transition systems everywhere! TCS 256(1-2): 63-92 (2001).
8. G. Franco, V. Manca. A Membrane System for the Leukocyte Selective Recruitment. WMC 2003: 181-190.
9. O. H. Ibarra, Z. Dang, Ö. Egecioglu. Catalytic P systems, semilinear sets, and vector addition systems. TCS 312(2-3): 379-399 (2004)
10. R. M. Karp, R. E. Miller. Parallel Program Schemata, J. of Computer and System Sciences 3: 147-195 (1969).
11. S. R. Kosaraju. Decidability of Reachability in Vector Addition Systems. STOC 1982: 267-281.
12. C. Li, Z. Dang, O. H. Ibarra, H.-C. Yen. Signaling P Systems and Verification Problems. ICALP 2005: 1462-1473
13. C. Martin-Vide, Gh. Pãun, A. Rodriguez Paton. On P Systems with Membrane Creation Computer Science J. of Moldova 9(2): 134–145 (2001).
14. E. W. Mayr. An Algorithm for the General Petri Net Reachability Problem. SIAM J. Comput. 13(3): 441-460 (1984).
15. M. Minsky. Computation: Finite and Infinite Machines, Prentice-Hall, 1967.
16. G. Paun. Computing with membranes. J. of Computer and System Science 61(1): 108-143, 2000.
17. C. A. Petri. Kommunikation mit Automaten. Ph.D. Thesis. University of Bonn, 1962.
18. W. Reisig. Petri Nets - An Introduction. Springer, 1985.