Nat Comput (2010) 9:795–818
DOI 10.1007/s11047-010-9214-0

# On the verification of membrane systems with dynamic structure

Giorgio Delzanno · Laurent Van Begin

Received: 16 February 2009 / Accepted: 30 July 2010 / Published online: 9 September 2010

**Abstract** We study computational properties of Gheorge Păun's P-systems extended with rules that model in an abstract way creation, dissolution, fusion and cloning of membranes. We investigate decision problems like reachability of a configuration, boundedness (finiteness of the state space), and coverability (verification of safety properties). Our analysis is aimed at understanding the expressive power of rules that dynamically modify the structure of a membrane.

**Keywords** P-systems · Verification · Decidability results

## 1 Introduction

Gheorge Păun's P-systems (Păun 2000) are a basic model of the living cell defined by a set of hierarchically organized membranes and by rules that dynamically distribute elementary objects in the component membranes. Among several other extensions (see e.g. the P systems webpage), in Bernardini and Manca (2003) have introduced rules that operate on the boundary of a membrane. The resulting model, named PB-systems, can be used to express complex interactions among biological membranes (Franco and Manca 2004).

In this paper we study verification problems for PB-systems extended with rules for dynamically changing the structure of membranes, i.e., *dissolution, creation, fusion* and *cloning* of membranes. The decision problems we consider are inspired by those studied in concurrency models like Petri nets. Specifically, we consider:

– *reachability* of a given configuration;
– *boundedness* of the state space;

G. Delzanno (✉)
Università di Genova, Genova, Italy
e-mail: giorgio@disi.unige.it

L. V. Begin
Université Libre de Bruxelles, Brussels, Belgium
e-mail: lvbegin@ulb.ac.be

– *coverability* of a given configuration, i.e., reachability of an infinite set of configurations sharing a given pattern.

We consider here PB-systems with symbol objects, interleaving semantics and no priorities. These limitations allow us to explore the expressiveness of our model independently from specific execution strategies and additional ordering on elementary objects.

Similar decision problems for qualitative analysis of subclasses of P-systems have been studied, e.g., in Ibarra et al. (2004), in Li et al. (2005) and Dal Zilio and Formenti (2004). For instance, Dal Zilio and Formenti (2004) have shown that reachability, boundedness and coverability are decidable for PB-systems with static membrane structure. In this paper we extend their analysis as explained below.

### 1.1 Reachability

We first show that reachability for PB-systems extended with fusion and cloning rules becomes undecidable. This result follows from a *weak* encoding of multi-counter machines. The encoding is weak in the following sense: some execution paths of the PB-system that simulates the counter machine may take a *wrong* turn into a path that does not correspond to a simulation of any path in the original model. The target configuration of a reachability problem is then used to restrict the simulation to *good* paths only. The encoding exploits the property that the set of reachable configurations of a PB-system with fusion and cloning may contain configurations with unbounded width (due to the presence of cloning) and with multisets of objects of unbounded size (due to the presence of internal and boundary rules). The undecidability proof can be adapted to the reachability problem for PB-systems in which cloning is replaced by creation and/or fusion is replaced by dissolution.

We then show that reachability becomes decidable in PB-systems extended with dissolution and fusion, as well as in PB-systems with creation and cloning. In both extensions, decidability follows from a reduction to Petri net reachability.

### 1.2 Boundedness

We first show that boundedness becomes undecidable for PB-systems extended with dynamic membrane creation. The proof is based on a reduction to boundedness for multi-counter machines, a well-known undecidable problem.

We then apply the theory of well-structured transition systems (Abdulla et al. 1996; Finkel and Schnoebelen 2001) to prove that boundedness is decidable for PB-systems extended with dissolution, fusion, and cloning. By adapting a result for lossy FIFO channel systems in Schnoebelen (2002), we also prove that this decision problem has non-primitive recursive complexity.

### 1.3 Coverability

We consider here a coverability problem defined on top of Kruskal's tree embedding (Kruskal 1960), i.e., we formulate the reachability problem by requiring that the (infinite) set of target configurations is upward-closed with respect to the tree embedding relation. This relation allows us to define tree patterns for reachability queries such as: "can we reach a configuration in which a membrane $m$ contains a membrane $n$ as one of its descendent nodes?".

Similarly to the case of boundedness, we show that coverability becomes undecidable for PB-systems extended with dynamic membrane creation. The proof is based on a reduction of the control state reachability problem for two counter machines, a well-known undecidable problem. Despite of this negative result, we can apply the theory of well-structured transition to prove that coverability (w.r.t. Kruskal's tree embedding) is decidable for PB-systems extended with dissolution, fusion, and cloning and that the problem has non-primitive recursive complexity.

From our results, it follows that PB-systems extended with fusion, cloning, and dissolution define a model whose expressive power is in between that of Petri nets and of Turing Machines. Furthermore, creation rules add expressive power to PB-systems in that they make coverability and boundedness undecidable. Interestingly, the addition of creation rules is not sufficient to make reachability undecidable, but its combination with rules that may delete membranes like dissolution or fusion makes the resulting model Turing equivalent.

The introduction of maximal parallelism and/or priorities would lead to a Turing complete model as in the case of PB-systems.

## 1.4 Related work

As shown in the P-systems web page (2000), several variants of Păun's original model have been proposed in the literature. Some examples are P-systems with active membranes and division (Păun 2001; Păun et al. 2004), with string objects (Păun 2000), with mobile membranes (Petre and Petre 1999), and with gemmation of vesicles (Besozzi et al. 2001).

Paun and Popa (2000) proposed a model with proteins on membranes and exchange rules similar to the boundary rules of PB systems. Concerning dynamic creation of membranes, in P-systems with active membranes (Păun 2001) every membrane is able to divide itself replicating its content in a new membrane. An interpretation of membranes as mobile ambients is proposed in Petre and Petre (1999), as a way to model secure transportation of substances between non-adjacent membranes. Creation of new membranes is also studied in Gemmating P-systems (Besozzi et al. 2001). Gemmating P-systems provide pre-dynamical rules, which are rules defining the gemmation of mobile membranes and the fusion of a mobile membrane with a skin membrane (i.e. since membranes are permeable only to small substances, mobile membranes are used to transport proteins). Gemmating P-systems are as powerful as Turing machines. The expressive power of fragments of Gemmating P-systems is studied in Besozzi et al. (2003). The universality problem of different forms of division in P-systems with active membranes has been studied in Alhazov et al. (2000) and Păun et al. (2004).

In the present paper we first introduce a set of abstract operations that can dynamically change the membrane structure. The operations are inspired to those introduced in models like Besozzi et al. (2001, 2003), Păun (2001) and Petre and Petre (1999). We then focus our attention on decision problems related to verification of functional properties of dynamical systems. In our analysis we consider maximal (resp. minimal) combinations of rules for structural modifications of membranes for obtaining decidability (resp. undecidability) results for problems like coverability, i.e., reachability of a configuration with certain structural properties. The coverability problem is of particular interest for verification of safety properties (does the system behave as we expect?). The study of property like coverability is a distinguishing point of our analysis with respect to previous work on the expressive power of P systems with dynamic membrane structure. Indeed, it is important to notice that properties like coverability may turn to be decidable even for models that are Turing equivalent under other types of observable properties (e.g. reachability of a fixed configuration).

As mentioned in the introduction, decidability results for basic PB systems have been obtained in Dal Zilio and Formenti (2004). Specifically, Dal Zilio and Formenti proved that reachability, boundedness and coverability are decidable for PB systems with symbol objects by using a reduction to Petri nets. In Delzanno and Van Begin (2007) we proved that reachability is still decidable for an extension of PB systems with creation of new membranes with fixed content (e.g., an empty membrane) or with membrane dissolution, but not both. In the present paper we combine the analysis in Delzanno and Van Begin (2007) and the preliminary results on fusion and cloning in Delzanno and Van Begin (2008) in order to study the expressive power of the model obtained by extending PB-systems with all four types of dynamic modifications of the membrane structure. Specifically, we show that the undecidability proofs for PB systems with creation and dissolution in Delzanno and Van Begin (2007) and with fusion and cloning (Delzanno and Van Begin 2008) can be combined to obtain negative results for specific combinations of the four extended rules. Furthermore, we extend the positive result for reachability in PB with creation in Delzanno and Van Begin (2007) by showing that reachability remains decidable when adding cloning [an operation considered in Delzanno and Van Begin (2008)]. Furthermore, we prove that reachability is decidable for PB-systems with both dissolution [introduced in Delzanno and Van Begin (2007)] and fusion [introduced in Delzanno and Van Begin (2008)].

To our knowledge, these are the first (un)decidability results for such a rich extension of PB-systems with interleaving semantics and symbol objects. Similar verification problems have been investigated for other variants of P-systems (e.g., signaling and catalytic P-systems) in Li et al. (2005).

Decidability results for reachability problems in fragments of models of biological systems given in process algebra (Bioambients) with restricted form of replication are given in Delzanno and Montagna (2007) and Zavattaro (2009). A detailed comparison of PB-systems with cloning and fragments of Bioambients (Regev et al. 2004) like those in Delzanno and Montagna (2007) and Zavattaro (2009) would deserve further investigations.

## 2 Preliminaries

In this section we recall the basic concepts needed throughout the paper. In particular, we briefly recall the main definitions of well-structured transition systems (Abdulla et al. 1996; Finkel and Schnoebelen 2001), Petri nets (Petri 1962; Reisig 1985), counter machines (Minsky 1967) and PB-systems (Franco and Manca 2004).

We first need some preliminary notions. Let $\mathbb{N}$ be the set of positive integers including 0. Consider a finite alphabet $\Gamma$ of symbols. A multiset over $\Gamma$ is a mapping $\mathbf{u} : \Gamma \mapsto \mathbb{N}$. For any $a \in \Gamma$, the value $\mathbf{u}(a)$ denotes the multiplicity of $a$ in $\mathbf{u}$ (the number of occurrences of symbol $a$ in $\mathbf{u}$). We often represent a multiset as a string $a_1 \cdots a_n$ of symbols, i.e., $a_i \in \Gamma$. Furthermore, we use $\epsilon$ to denote the empty multiset, i.e., such that $\epsilon(a) = 0$ for any $a \in \Gamma$. As an example, for $\Gamma = \{a, b, c, d\}$, the string $a\ b\ c\ c$ represents the multiset $\mathbf{u}$ such that $\mathbf{u}(a) = \mathbf{u}(b) = 1$, $\mathbf{u}(c) = 2$, and $\mathbf{u}(d) = 0$. We use $\Gamma^{\otimes}$ to denote the set of all possible finite multisets over the alphabet $\Gamma$. Given two multisets $\mathbf{u}, \mathbf{v}$ over $\Gamma$, we write $\mathbf{u} \preceq \mathbf{v}$ if $\mathbf{u}(a) \leq \mathbf{v}(a)$ for all $a \in \Gamma$. We use $\mathbf{u} \prec \mathbf{v}$ to denote that $\mathbf{u} \preceq \mathbf{v}$ and $\mathbf{v} \not\preceq \mathbf{u}$. Furthermore, we use $\oplus$ and $\ominus$ to denote multiset union and difference, respectively. Specifically, for any $a \in \Gamma$ we have that $(\mathbf{u} \oplus \mathbf{v})(a) = \mathbf{u}(a) + \mathbf{v}(a)$, and $(\mathbf{u} \ominus \mathbf{v})(a) = \max(0, \mathbf{u}(a) - \mathbf{v}(a))$ where max($i$, $j$) with $i,j \in \mathbb{N}$ returns the largest number between $i$ and $j$.

## 2.1 Well-structured transition systems

A *transition system* is a tuple $G = (S, T)$ where $S$ is a (possibly infinite) set of configurations, $T \subseteq S \times S$ is a transition relation between configurations. We use $\gamma \rightarrow \gamma'$ to denote that $(\gamma, \gamma') \in T$. We say that $\gamma'$ is reachable from $\gamma$ if $(\gamma, \gamma')$ is in the reflexive-transitive closure of $T$.

A quasi-ordering $(S, \sqsubseteq)$ is a *well-quasi ordering* (wqo for short) if for any infinite sequence $\gamma_1\gamma_2\ldots\gamma_i\ldots$ such that for all $i \geq 1, \gamma_i \in S$, there exist indexes $i < j$ such that $\gamma_i \sqsubseteq \gamma_j$.

A transition system $G = (S, T)$ is a *well-structured transition system* (*wsts* for short) with respect to a quasi-order $\sqsubseteq \subseteq S \times S$ iff:

(i) $(S, \sqsubseteq)$ is a well-quasi ordering;
(ii) for any configurations $\gamma_1, \gamma_1', \gamma_2$ such that $\gamma_1 \sqsubseteq \gamma_1'$ and $\gamma_1 \rightarrow \gamma_2$, there exists $\gamma_2'$ such that $\gamma_1' \rightarrow \gamma_2'$ and $\gamma_2 \sqsubseteq \gamma_2'$, i.e., $G$ is *monotonic*.

A wsts is said *strictly monotonic* when $\gamma_1 \sqsubset \gamma_1'$, i.e., $\gamma_1 \sqsubseteq \gamma_1'$ and $\gamma_1' \not\sqsubseteq \gamma_1$, implies that $\gamma_2 \sqsubset \gamma_2'$. In this paper we focus on the following problems on (well-structured) transition systems.

*Problem 1 (Reachability)* Given a transition system $G$ with an initial configuration $\gamma_0$ and a configuration $\gamma$, the *reachability problem* is defined as the problem of checking whether $\gamma$ is reachable from $\gamma_0$.

*Problem 2 (Boundedness)* Given a transition system $G$ with an initial configuration $\gamma_0$, the *boundedness problem* is defined as the problem of checking whether the set of configurations reachable from $\gamma_0$ is finite.

*Problem 3 (Coverability)* Given a transition system $G$ and a quasi-order $\sqsubseteq$ over configurations of $G$, an initial configuration $\gamma_0$ and a configuration $\gamma$, the $\sqsubseteq$*-coverability problem* is defined as the problem of checking whether there is a configuration $\gamma'$ which is reachable from $\gamma_0$ and such that $\gamma \sqsubseteq \gamma'$.

## 2.2 Petri nets and Transfer nets

A Petri net is a model for concurrency defined by a set of places and tokens that flow from one place to another according to a given set of transitions. A Petri net (Petri 1962; Reisig 1985) is a tuple $(P, T, \mathbf{m}_0)$ where $P$ is a finite set of *places*, $T$ is a finite set of *transitions* $(T \in P^{\otimes} \times P^{\otimes})$, and $m_0$ is the initial configuration. A Petri net configuration, called marking, is a multiset over $P$ that counts how many tokens are in a each place. Token are indistinguishable objects (we can just count them).

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). The semantics of Petri nets is given in terms of transition systems. Given a marking $\mathbf{m}$ and a place $p$, we say that the place $p$ contains $\mathbf{m}$*(p) tokens*.

The transition relation is defined as follows. A transition $t$ is enabled at the marking $\mathbf{m}$ if $^\bullet t \preceq \mathbf{m}$. If this is the case, $t$ can fire and produces a marking $\mathbf{m}'$ (usually written $\mathbf{m} \xrightarrow{t} \mathbf{m}'$)

defined as $(\mathbf{m} \ominus {}^\bullet t) \oplus t^\bullet$. A pair of markings $(\mathbf{m}, \mathbf{m}')$ is in the transition relation if there is a transition $t \in T$ such that $\mathbf{m} \xrightarrow{t} \mathbf{m}'$.

As an example, given $P = \{p, q, r\}$ the transition $t$ with ${}^\bullet t = ppq$ and $t^\bullet = qqrr$ is enabled when the current marking has at least two tokens in place $p$ and one in place $q$. The firing of the transition removes those tokens and adds two tokens to place $q$ and two tokens to place $r$. Thus, we have that

$$\mathbf{m}_0 = ppppq \xrightarrow{t} \mathbf{m}_1 = ppqqrr \xrightarrow{t} \mathbf{m}_2 = qqqrrrr.$$

Hence, $\mathbf{m}_2$ is reachable from $\mathbf{m}_0$.

A Transfer net is a Petri net where transitions may be extended with one transfer arc. A transfer moves all the tokens from a source place to a target place. Formally, a Transfer net is a tuple $(P, T, \mathbf{m}_0, \mathsf{T})$ where $(P, T, \mathbf{m}_0)$ is a Petri net and $\mathsf{T} : T \mapsto P \times P$ is a partial function that associates to some transitions a pair of (distinct) places. A transition $t$ is enabled in a marking $\mathbf{m}$ as in the Petri net case; and firing $t$ leads to a marking $\mathbf{m}'$, noted $\mathbf{m} \xrightarrow{t} \mathbf{m}'$, computed in three steps as follows. The first step consists in computing the intermediate marking $\mathbf{m}_1 = \mathbf{m} \ominus^\bullet t$. Second, we compute the intermediate marking $\mathbf{m}_2$ from $\mathbf{m}_1$ such that either $\mathsf{T}$ is undefined and $\mathbf{m}_2 = \mathbf{m}_1$; otherwise $\mathsf{T} = (p_1, p_2)$ and $\mathbf{m}_2(p_1) = 0, \mathbf{m}_2(p_2) = \mathbf{m}_1(p_1) + \mathbf{m}_2(p_2)$ and $\mathbf{m}_2(p) = \mathbf{m}_1(p)$ for all the places $p \notin \{p_1, p_2\}$. Finally, the target marking is $\mathbf{m}' = \mathbf{m}_2 \oplus t^\bullet$.

The following theorem recalls a fundamental result for Petri nets.

**Theorem 1** (Kosaraju 1982; Mayr 1984) *The reachability problem for Petri nets is decidable.*

The boundedness and the $\preceq$-coverability problems for Transfer nets are decidable (Abdulla et al. 1996; Finkel and Schnoebelen 2001). Moreover, the construction of Schnoebelen (2002) can be adapted to Transfer nets to show that those problems have a non-primitive recursive complexity.

**Theorem 2** (Abdulla et al. 1996; Finkel and Schnoebelen 2001; Schnoebelen 2002) *The boundedness and $\preceq$-coverability problems for Transfer nets are decidable with algorithms of non-primitive recursive time complexity.*

### 2.3 Counter machines

A counter machine (Minsky 1967) 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 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 $\ell$, the increment operation on $c_i$ adds one unit to the current value of $c_i$ and then moves to the successor location $\ell'$. A decrement of the counter $c_i$ in location $\ell$ can be executed if the value of $c_i$ is strictly greater than zero. When executed, the decrement operation on $c_i$ removes one unit from the current value of $c_i$ and then moves to the successor location $\ell'$. When executed in location $\ell$, the zero-test on $c_i$ moves to the successor location $\ell'$ only if $c_i$ has value zero. For a fixed initial location $\ell_0$, the initial configuration has location $\ell_0$ and all counters set to zero. The successors of a configuration $c$ are those obtained by applying an instruction corresponding to the location of $c$. The *if-then-else* instruction $\ell$ : *if* $c_i = 0$ *goto* $\ell_1$ *else goto* $\ell_2$ typically found in the definition of counter machines can be simulated here by the following rules: the positive case is simulated with a zero-test $\ell : c_i = 0$ *goto* $\ell_1$ and the negative case is simulated with a decrement $\ell :$ *dec* $c_i$ *goto* $\ell'$ followed by an increment $\ell' :$ *inc* $c_i$ *goto* $\ell_2$ where $\ell'$ is a fresh control state.

In the following, we consider the quasi-order $\sqsubseteq$ over configurations such that $c \sqsubseteq c'$ if and only if $c$ and $c'$ have the same location. In other words, the $\sqsubseteq$-coverability problem asks if a particular location is reachable. It is well-known that the model of *2-counter machines*, i.e. counter machines with two counters, can simulate a Turing machine. As a consequence, the next theorem (adapted from Minsky 1967) holds

**Theorem 3** *The boundedness, reachability and $\sqsubseteq$-coverability problems for 2-counter machines are undecidable.*

## 2.4 P-systems with boundary rules

A PB system (Bernardini and Manca 2003) with symbol objects is a tuple $\Pi = (\Gamma, N, M_0, R, \mu_0)$, where

- $\Gamma$ is a finite alphabet of symbols.
- $N$ is a finite set of membrane names/types.
- $M_0$ is a finite tree representing the *membrane structure*. Each node $n$ of $M_0$ corresponds to a membrane and is labeled with a membrane name/type $\mathsf{type}(n) \in N$. We use $\mathsf{nodes}(M_0)$ to denote the set of nodes of $M_0$.
- $R$ is a finite set of rules.
- $\mu_0 : \mathsf{nodes}(M_0) \mapsto \Gamma^{\otimes}$ is the *initial configuration*, i.e., a mapping from membranes to multisets of objects from $\Gamma$.

Rules can be of the following two forms[1]:

$$\text{Internal}: \quad [_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v} \tag{1}$$

$$\text{Boundary}: \quad \mathbf{u}\, [_i\, \mathbf{v} \rightarrow \mathbf{u}'\, [_i\, \mathbf{v}' \tag{2}$$

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

The semantics of PB-systems is given in term of transition systems. The set of configurations of a PB system $\Pi$ is the set of pairs $(M_0, \mu)$ where $\mu$ is a distribution of objects in $\Gamma$ in the membranes in $M$, i.e., $\mu : \mathsf{nodes}(M_0) \mapsto \Gamma^{\oplus}$ is a mapping from $\mathsf{nodes}(M_0)$ to $\Gamma^{\otimes}$. Notice that the tree structure $M_0$ is the same for all the configurations of a particular PB systems. However, it will become useful in the remainder when considering extensions of PB systems where the structure of configurations may evolve during executions.

*Remark 1* In the paper, we do not distinguish two trees $M_1$ and $M_2$ if they are isomorphic. For instance, a tree with the root node $n_1$ having one child $n_2$ is considered as the same tree than the one with $n_2$ as root node having $n_1$ as unique child. In other words, we do not distinguish two configurations $(M_1, \mu)$ and $(M_2, \mu')$ and we consider that $(M_1, \mu) = (M_2, \mu')$ if $(M_1, \mu')$ is obtained from $(M_2, \mu)$ by node permutations.

[1] We consider here a slight generalization of the model in Dal Zilio and Formenti (2004) in which we allow any kind of transformation between two membranes.

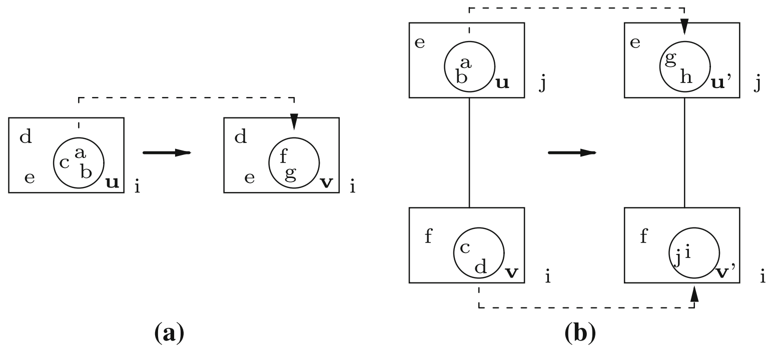

**Fig. 1** **a** An internal rule and **b** a boundary rule

The transition relation is defined as follows. A rule of the form (1) is enabled at ($M_0$, $\mu$), if there exists a membrane $n \in \mathsf{nodes}(M_0)$ with $\mathsf{type}(n) = i$ and $\mathbf{u} \preceq \mu(n)$. Its application leads to a new configuration $(M_0, \mu')$ such that

$$\mu'(n) = (\mu(n) \ominus \mathbf{u}) \oplus \mathbf{v}$$

and $\mu'(n') = \mu(n')$ for any other node $n' \in \mathsf{nodes}(M)$ such that $n \neq n'$. Figure 1a shows the effect of the internal rule $[_i\ \mathbf{u} \rightarrow [_i\ \mathbf{v}$ where $\mathbf{u} = abc$ and $\mathbf{v} = fg$.

Suppose now that a membrane $m \in \mathsf{nodes}(M_0)$ with $\mathsf{type}(m) = j$ contains as immediate successor in $M_0$ a node $n$ with $\mathsf{type}(n) = i$. A rule of the form (2) is enabled at ($M_0$, $\mu$), if $\mathbf{u} \preceq \mu(m)$ and $\mathbf{v} \preceq \mu(n)$. Its application leads to a new configuration $(M_0, \mu')$ such that

$$\mu'(m) = (\mu(m) \ominus \mathbf{u}) \oplus \mathbf{u}'$$
$$\mu'(n) = (\mu(n) \ominus \mathbf{v}) \oplus \mathbf{v}'$$

and $\mu'(m') = \mu(m')$ for any node $m' \in \mathsf{nodes}(M_0)$ such that $m' \notin \{m, n\}$. We have a transition from a configuration $c$ to $c'$, i.e., $c \rightarrow c'$, if $c'$ can be obtained from $c$ by applying a rule in $R$. Figure 1b shows the effect of the boundary rule $\mathbf{u}[_i\ \mathbf{v} \rightarrow \mathbf{u}'[_i\ \mathbf{v}'$ where $\mathbf{u} = ab, \mathbf{v} = cd, \mathbf{u}' = gh$ and $\mathbf{v}' = ij$.

The reachability and boundedness problems are decidable for PB systems with symbol objects. The proof is based on an encoding of PB systems into Petri nets (see Dal Zilio and Formenti 2004 for more details).

The quasi-order we consider to specify the coverability problem for PB systems (and their extensions) is the tree embedding (a.k.a. Kruskal) order $\leq_K$ over trees (Kruskal 1960).

**Definition 1** (*Tree Embedding*) Let $M$ and $M'$ be two trees; and assume a quasi order $\sqsubseteq$ over nodes. Then, $M \leq_K M'$ iff there exists an injection $\rho : \mathsf{nodes}(M) \mapsto \mathsf{nodes}(M')$ such that ($i$) for all $n \in \mathsf{nodes}(M), n \sqsubseteq \rho(n)$ and ($ii$) for all $n, n' \in \mathsf{nodes}(M)$, we have that $n'$ is in the sub-tree rooted by $n$ iff $\rho(n')$ is in the sub-tree rooted by $\rho(n)$.

In the case of PB configurations, the order $\sqsubseteq$ is defined as follows. Given a node $n$ of a configuration ($M$, $\mu$) and a node $n'$ of $(M', \mu'), n \sqsubseteq n'$iff $\mathsf{type}(n) = \mathsf{type}(n')$ and $\mu(n) \preceq \mu'(n')$.

From the Kruskal tree theorem [Kruskal 1960] (the version for unordered trees can be found in Bezem et al. (2003), we know that if $\sqsubseteq$ is a well-quasi ordering (wqo) then $\leq_K$ is also a wqo (see preliminaries for def. of wqo). By Dickson's lemma (Dickson 1913), the order $\sqsubseteq$ is a wqo. Thus, the order $\leq_K$ is a wqo over PB configurations.

## 3 PB systems with creation, dissolution, cloning, and fusion

In this section we define rules that can modify the structure of configurations. Hence, a configuration is now a pair $(M, \mu)$ where $M$ may be any tree.

A *dissolution rule* has the form

$$\text{Dissolution}: \quad [_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v} \cdot \delta \tag{3}$$

where $\mathbf{u}, \mathbf{v} \in \Gamma^{\otimes}, i \in N$ and $\delta$ is a symbol not in $\Gamma$. The intuitive meaning of this rule is that after applying the rule $[_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v}$ a membrane $n$ with $\mathsf{type}(n) = i$ is dissolved and its content (including its sub-membranes) is moved to the membrane $m$ that contains $n$ as immediate successor in the current membrane structure. Formally, a dissolution rule like (3) operates on a configuration $c = (M, \mu)$ as follows. For simplicity, we assume that membrane $n$ is not the root of $M$. Suppose now that $n \in \mathsf{nodes}(M)$ is an immediate successor of $m$ in $M$ and $\mathsf{type}(n) = i$. The rule is enabled if $\mathbf{u} \preceq \mu(n)$. Its application leads to a new configuration $c' = (M', v')$ such that

– $M'$ is the tree obtained by removing node $n$ and by letting all successor nodes of $n$ become successors of $m$;
– $v'$ is the mapping defined as $v'(m) = v(m) \oplus (v(n) \ominus \mathbf{u}) \oplus \mathbf{v}$ and $v'(p) = v(p)$ for all the membranes $p \in \mathsf{nodes}(M')$ such that $p \neq m$.

Figure 2 shows the effect of the dissolution rule $[_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v} \cdot \delta$ where $\mathbf{u} = ab$, $\mathbf{v} = e$.

A *creation rule* has the form

$$\text{Creation}: \quad a \rightarrow [_i\, \mathbf{v}]_i \tag{4}$$

where $a \in \Gamma, \mathbf{v} \in \Gamma^{\otimes}$, and $i \in N$. The intuitive meaning of this rule is that after applying the rule $a \rightarrow [_i\, \mathbf{v}]_i$ inside a membrane $j$, object $a$ is replaced by the new membrane $i$ containing the multiset of objects $\mathbf{v}$. A creation rule like (4) operates on a configuration $c = (M, \mu)$ as follows. Suppose that $n \in \mathsf{nodes}(M)$. The rule is enabled if $a \in \mu(n)$. Its application leads to a new configuration $c' = (M', v')$ such that

– $M'$ is the tree obtained by adding a new node $m$ with $\mathsf{type}(m) = i$ as a successor of node $n$;
– $v'$ is the mapping defined as $v'(n) = v(n) \ominus a, v'(m) = \mathbf{v}$, and $v'(p) = v(p)$ for all the membranes $p \in \mathsf{nodes}(M')$ such that $p \notin \{m, n\}$.

The only precondition for the application of rule (4) is the existence of object $a$ in a membrane, Thus, rules of type (4) can be applied inside any membrane. Furthermore, such

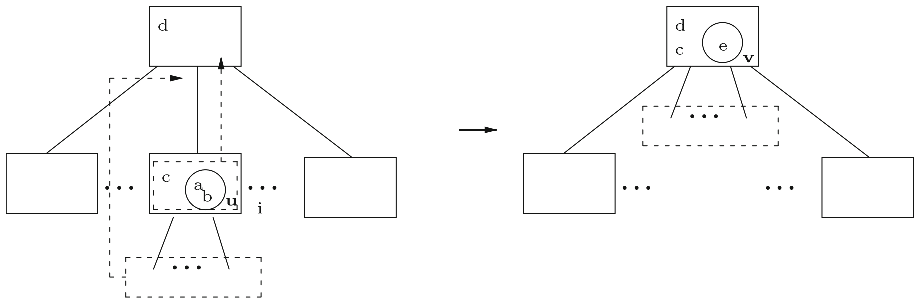

**Fig. 2** An example of dissolution rule

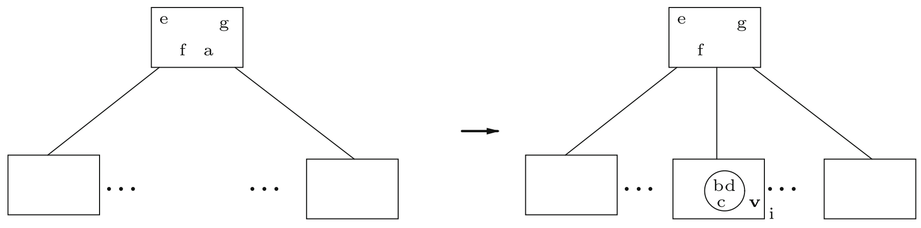

**Fig. 3** An example of creation rule

their application may create different nodes with the same membrane name. Figure 3 shows the effect of the transition $a \rightarrow [_i\ \mathbf{v}]_i$ where $\mathbf{v} = bcd$.

A *fusion rule* has the form

$$\text{Fusion :} \quad [_i\ \mathbf{u}[_j\ \mathbf{v} \rightarrow [_k\ \mathbf{w} \tag{5}$$

where $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \Gamma^{\otimes}$ and $i, j, k \in N$. The rule (5) models the fusion of a membrane $m$ with $\mathsf{type}(m) = i$ containing the multiset of objects $\mathbf{u}$ with a membrane $m'$ with $\mathsf{type}(m') = j$ containing the multiset of objects $\mathbf{v}$. Objects in $\mathbf{u}$ and $\mathbf{v}$ are consumed during this process. The fusion of the two membranes generates a new membrane $n$ with $\mathsf{type}(n) = k$ that contains $\mathbf{w}$ and the (remaining) contents of both $m$ and $m'$. Formally, a fusion rule like (5) operates on a configuration $c = (M, \mu)$ as follows. Suppose that $m$ and $m'$ are two children of a node $p$ in $M$ such that $\mathsf{type}(m) = i$ and $\mathsf{type}(m') = j$. The rule is enabled if $u \preceq \mu(m)$ and $v \preceq \mu(m')$. Its application leads to a new configuration $c' = (M', \mu')$ such that

– $M'$ is the tree obtained by removing the nodes $m$ and $m'$, adding a new node $n$ with $\mathsf{type}(n) = k$, and by letting all successor nodes of $m$ and $m'$ become successors of $n$. The parent node of $n$ is $p$, the parent of the nodes $m$ and $m'$ in the tree $M$;
– $\mu'$ is the mapping defined as

$$\mu'(n) = (\mu(m) \ominus \mathbf{u}) \oplus (\mu(m') \ominus \mathbf{v}) \oplus \mathbf{w}$$

and $\mu'(n') = \mu(n')$ for any other node $n' \in \mathsf{nodes}(M')$ such that $n' \neq n$.

Figure 4 shows the effect of the fusion rule $[_i\ \mathbf{u}[_j\ \mathbf{v} \rightarrow [_k\ \mathbf{w}$ where $\mathbf{u} = ab$, $\mathbf{v} = c$ and $\mathbf{w} = de$.

Finally, a *cloning rule* has the form

$$\text{Cloning :} \quad [_i\ \mathbf{u} \rightarrow [_i\ \mathbf{v}[_i\ \mathbf{w} \tag{6}$$

where $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \Gamma^{\otimes}, i \in N$. Such a rule clones a sub-tree rooted by a membrane $n$ with $\mathsf{type}(n) = i$ containing the multiset of objects $\mathbf{u}$. During the cloning, the objects in $\mathbf{u}$ are consumed and replaced by the multiset of objects $\mathbf{v}$ in $n$ and by the multiset of objects $\mathbf{w}$ in the clone of $n$. This definition allows us to define both perfect clones (i.e., two copies of the same membrane) or to distinguish the clone from the original membrane by using the objects in $\mathbf{w}$ and $\mathbf{v}$, respectively. The latter type of cloning can be used to disable a second application of cloning immediately after the generation of the clone (i.e., avoid cloning rules that are enabled forever and thus applied without control). Formally, a cloning rule like (6) operates on a configuration $c = (M, \mu)$ as follows. Suppose that $M$ has a node $m$ with a successor $n$ with $\mathsf{type}(n) = i$. The rule is enabled if $\mathbf{u} \preceq \mu(n)$. Its application leads to a new configuration $c' = (M', \mu')$ such that

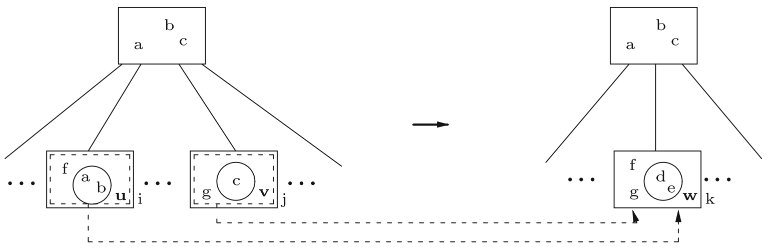

**Fig. 4** An example of fusion rule

- $M'$ is the tree obtained by adding a new copy of the tree rooted by $n$ as sub-tree of $m$;
- $\mu'$ is the mapping defined as follows. For any node $n'$ in the sub-tree rooted by $n$, let $\mathsf{Clone}(n')$ be its copy in the new sub-tree. Then, we have that
  - $\mu'(n) = (\mu(n) \ominus \mathbf{u}) \oplus \mathbf{v}$;
  - $\mu'(\mathsf{Clone}(n)) = (\mu(n) \ominus \mathbf{u}) \oplus \mathbf{w}$;
  - $\mu'(\mathsf{Clone}(n')) = \mu(n')$ for any node $n' \neq n$ in the sub-tree rooted by $n$;
  - $\mu'(n') = \mu(n')$ for the other nodes $n' \in \mathsf{nodes}(M')$.

Figure 5 shows the effect of the cloning rule $\lfloor_i\, \mathbf{u} \rightarrow \lfloor_i\, \mathbf{v} \lfloor_i\, \mathbf{w}$ where $\mathbf{u} = ab$, $\mathbf{v} = cd$ and $\mathbf{w} = ef$.

*Example 1* Let $\Gamma = \{a, b, c, d, e, f, g, h, u, v, w\}$ and $N = \{i\}$. For simplicity, configurations are represented here as terms. Specifically, objects are represented as constants, and a membrane of type $i$ containing $t_1, \ldots, t_n$ as a term of the form $[t_1, \ldots, t_n]$. Hence, $[a[b][c]]$ represents the configuration where the root node contains the object $a$ and two membranes, with objects $b$ and $c$, respectively. Now consider the initial configuration $[a\ [b \cdot d]\ [c]]$ and the following set of rules:

$$
\begin{array}{ll}
(r_1) & d \rightarrow \lfloor_i\ g \rfloor_i\ (\mathit{creation}) \\
(r_2) & b\ \lfloor_i\ g \rightarrow e\ \lfloor_i\ h\ (\mathit{boundary}) \\
(r_3) & \lfloor_i\ e \rightarrow \lfloor_i\ v\ \lfloor_i\ u\ (\mathit{cloning}) \\
(r_4) & \lfloor_i\ u \lfloor_i\ c \rightarrow \lfloor_i\ w\ (\mathit{fusion}) \\
(r_5) & \lfloor_i\ w \rightarrow \lfloor_i\ a \cdot \delta\ (\mathit{dissolution}) \\
(r_6) & \lfloor_i a \cdot a \rightarrow \lfloor_i g\ (\mathit{internal})
\end{array}
$$

Then, we have the following computation:

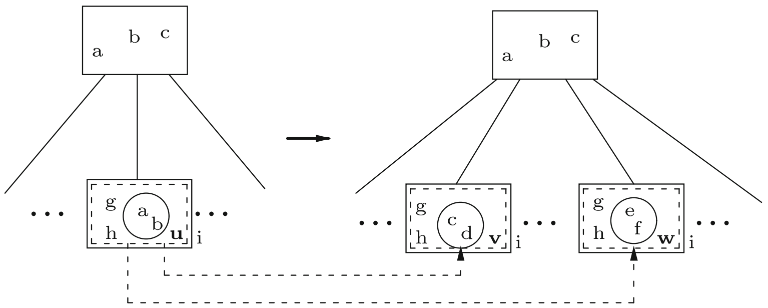

**Fig. 5** An example of cloning rule

$$[a\ [b \cdot d]\ [c]] \rightarrow_{r_1} [a\ [b\ [g]]\ [c]] \rightarrow_{r_2} [a\ [e\ [h]]\ [c]] \rightarrow_{r_3} [a\ [v\ [h]]\ [u\ [h]]\ [c]]$$
$$\rightarrow_{r_4} [a\ [v\ [h]]\ [w\ [h]]] \rightarrow_{r_5} [a \cdot a\ [v\ [h]]\ [h]] \rightarrow_{r_6} [g\ [v\ [h]]\ [h]]$$

The reachability/boundedness/coverability problems are naturally extended to PB systems extended with rules of the form (3–6).

## 4 Reachability for extended PB systems

In this section we investigate the reachability problem for PB systems extended with creation, dissolution, fusion, and cloning. Given $S \subseteq \{3, 4, 5, 6\}$, we use the notation $S$-PB system to denote a PB system extended with all rules in $S$. For instance, a {3,4}-PB system is obtained by extending PB systems with rules (3) and (4).

### 4.1 An undecidability result for extended PB systems

We first show that the reachability problem is decidable in PB systems extended with rules that can both increase and decrease the number of membranes.

**Theorem 4** *The reachability problem for $\{i, j\}$-PB systems with $i \in \{3, 5) \}$ and $j \in \{4, 6\}$ is undecidable (where (3)=dissolution, (4)=creation, (5)=fusion, (6)=cloning).*

*Proof* Let us first recall the undecidability proof given for {5,6}-PB systems in Delzanno and Van Begin (2008). We reduce the reachability problem for two counter machines. Our reduction uses the following types of membrane: *cs* (for control structure), $c_1$, $c_2$, and *trash*. Each configuration has a root node of type *cs*. Counters are encoded with membranes of type $c_1$ and $c_2$, that, together with a *trash* membrane, are all contained into the *cs* membrane. The trash membrane is used to validate executions. The set of objects contains the control states of the counter machine and their primed version, i.e. the root membrane containing control state $l$ means that the counter machine is in control state $l$. The primed versions correspond to intermediate states. Furthermore, we use objects of type $o$ to encode the value of counters. A membrane of type $c_1$ with $k$ objects $o$ represents the value $k$ for the first counter. Finally, we use objects of type $active_1$, $active_2$, $clone_1$, $clone_2$, *fusion* and *fused* to guide the simulation steps.

The initial configuration with control state $l_0$ and both counters equal to 0 is encoded with a configuration where the root (of type cs) contains $l_0$, the child of type $c_i$ contains the object $active_i$ and the trash membrane is empty. An increment of counter $i$ from location $l_1$ to location $l_2$ is simulated by a rule

$$l_1\ [_{c_i}\ active_i \rightarrow l_2 [_{c_i}\ active_i \cdot o$$

A decrement of counter $i$ from location $l_1$ to location $l_2$ is simulated by a rule

$$l_1 [_{c_i}\ active_i \cdot o \rightarrow l_2 [_{c_i}\ active_i$$

Finally, a zero test on counter $i$ and a move from $l_1$ to $l_2$ is simulated by four rules. The first rules are used to clone membrane $c_i$:

$$l_1 [_{c_i}\ active_i \rightarrow l'_1 [_{c_i}\ clone_i$$
$$[_{c_i} clone_i \rightarrow [_{c_i}\ active_i [_{c_i}\ fusion$$

The next rule can be fired only after the preceding ones and fuses the trash membrane with the copy of membrane $c_i$ containing the *fusion* object:

$$[_{trash}\epsilon[_{c_i} fusion \rightarrow [_{trash} fused$$

Finally, after the fusion the control state moves to $l_2$ by applying the following rule:

$$l'[_{trash} fused \rightarrow l_2[_{trash}\ \epsilon$$

Notice that if we simulate a test for zero on counter $i$ and the membrane $c_i$ contains at least one object $o$ then the *trash* membrane contains at least one object $o$ after the simulation. Furthermore there is no rule that decreases the number of objects $o$ in *trash*. *trash* remains empty while the system simulates correctly the counter machine. Thus, the configuration with control state $l$ and counter $c_i$ equal to $v_i$ is reachable if and only if the configuration where the root node contains $l$, its child $c_i$ contains $v_i$ instances of $o$ (and one object $active_i$) and an empty trash membrane is reachable.

The construction can be easily adapted to the other cases. Indeed, cloning of a membrane $c_i$ can be replaced by the creation of a membrane $c_i$ containing the unique object $active_i$, and fusion can be replaced by a dissolution. In the last case, the *trash* membrane becomes useless (since with dissolution rules the parent inherits the objects of the discarded node) and can be removed. □

## 4.2 Decidability results for restricted classes

We now investigate the cases where PB systems are extended with either rules that increase the number of nodes or rules that decrease the number of nodes. In both cases, the number of tree structures that we have to consider in configurations to decide the reachability problem is finite. As a consequence, the reachability problem for extended PB systems can be reduced to the reachability problem for Petri nets.

First, we consider PB systems extended with fusion and dissolution rules.

**Theorem 5** *The reachability problem for* $\{3,5\}$*-PB systems, i.e. PB systems extended with dissolution and fusion rules, is decidable.*

*Proof* We reduce the reachability problem for Petri nets. Let $(\Gamma, N, M, R, \mu_0)$ be a PB-system extended with dissolution and fusion rules. First, notice that for any configuration $(M', \mu')$ reachable from $(M, \mu_0)$ we have that $|\mathsf{nodes}(M')|$, the cardinality of $\mathsf{nodes}(M')$, is smaller than or equal to $|\mathsf{nodes}(M)|$, the cardinality of $\mathsf{nodes}(M)$. Hence, the number of membranes in reachable configurations is bounded by $|\mathsf{nodes}(M)|$. Let $S$ be the finite set of trees with at most $|\mathsf{nodes}(M)|$ nodes. We associate to the nodes $n$ of those trees a value $1 \leq \mathsf{val}(n) \leq |\mathsf{nodes}(M)|$. Let $(M, \mathsf{val})$ denotes a tree where a value $\mathsf{val}(n)$ is associated to each nodes $n \in \mathsf{nodes}(M)$ and let $S_{tree}$ be the set of labeled trees $(M, \mathsf{val})$ where $M \in S$. Notice that $S_{tree}$ may contain two elements $(M, \mathsf{val})$ and $(M', \mathsf{val}')$ such that $M = M'$ but $\mathsf{val} \neq \mathsf{val}'$. The values $\mathsf{val}(n)$ are used as names in the following and, when considering nodes of a tree $(M, \mathsf{val})$, we do not distinguish them from their name, i.e., in the following we consider that the nodes are values $i$ such that $1 \leq i \leq |\mathsf{nodes}(M)|$.

The Petri net $(P, T, \mathbf{m}_0)$ is defined as follows. The encoding takes care of the current tree structure $M'$ by using places $(M', \mathsf{val})$, i.e. the set $P$ of places contains a place $(M', \mathsf{val})$ for any $(M', \mathsf{val}) \in S_{\text{tree}}$. The content of a node $i$ $(1 \leq i \leq |\mathsf{nodes}(M)|)$ is tracked with the places $p_{i,a} \in P$ where $a \in \Gamma$; i.e. for all $1 \leq i \leq |\mathsf{nodes}(M)|$, for all $a \in \Gamma$, we have a place $p_{i,a}$ in $P$. Furthermore, we have the places $p_i^{source}, p_i^{target} \in P$ for all $1 \leq i \leq |\mathsf{nodes}(M)|$. Those places are used when simulating dissolution or fusion. In both cases, a membrane disappear from the configuration and its content moves to another membrane. To achieve that goal, we use transitions $t_a$ with $a \in \Gamma$ that moves tokens one by one from a place $p_{i,a}$ to a place $p_{j,a}$ when $p_i^{source}$ and $p_j^{target}$ contains one token. Finally, dissolution and fusion are simulated in several steps in the Petri net. Hence, we have a place $lock \in P$ used to ensure that only one rule is simulated at a time.

Given the initial marking $\mathbf{m}_0$, we have that the encoding of $(M, \mu_0)$ is such that $\mathbf{m}_0((M, \mathsf{val})) = 1$ for a non-deterministically chosen $\mathsf{val}$, for all $i \in \mathsf{nodes}(M)$, for all $a \in \Gamma, \mathbf{m}_0(p_{i,a}) = \mu_0(i)(a)$ and $\mathbf{m}_0(lock) = 1$. For all the other places $p \in P$, we have that $\mathbf{m}_0(p) = 0$.

The set of transitions $T$ is defined as follows.

- For any internal rule $[_i\, \mathbf{u} \to [_i\, \mathbf{v}$, and any $(M', \mathsf{val}) \in S_{\text{tree}}$, we have the following transitions in $T$. If the node $j \in \mathsf{nodes}(M')$ is such that $\mathsf{type}(j) = i$, then we have a transition $t \in T$ such that ${}^\bullet t((M', \mathsf{val})) = 1, {}^\bullet t(lock) = 1, {}^\bullet t(p_{j,a}) = \mathbf{u}(a)$ for all $a \in \Gamma, {}^\bullet t(p) = 0$ for all the other places in $P$; and $t^\bullet((M', \mathsf{val})) = 1, t^\bullet(lock) = 1, t^\bullet(p_{j,a}) = \mathbf{v}(a)$ for all $a \in \Gamma$ and $t^\bullet(p) = 0$ for all the other places in $P$;
- For any boundary rule $\mathbf{u}[_i\, \mathbf{v} \to \mathbf{u}'[_i\, \mathbf{v}'$ and any $(M', \mathsf{val}) \in S_{\text{tree}}$, we have the following transitions in $T$. If the node $j \in \mathsf{nodes}(M')$ is such that $\mathsf{type}(j) = i$ and $l \in \mathsf{nodes}(M')$ is the parent of $j$, then we have a transition $t \in T$ such that ${}^\bullet t((M', \mathsf{val})) = 1, {}^\bullet (lock) = 1, {}^\bullet t(p_{j,a}) = \mathbf{v}(a)$ and ${}^\bullet t(p_{l,a}) = \mathbf{u}(a)$ for all $a \in \Gamma, {}^\bullet t(p) = 0$ for all the other places in $P$; and $t^\bullet((M', \mathsf{val})) = 1, t^\bullet(lock) = 1, t^\bullet(p_{j,a}) = \mathbf{v}'(a)$ and $t^\bullet(p_{l,a}) = \mathbf{u}'(a)$ for all $a \in \Gamma$ and $t^\bullet(p) = 0$ for all the other places in $P$;
- For any dissolution rule $[_i\, \mathbf{u} \to [_i\, \mathbf{v} \cdot \delta$ and any $(M', \mathsf{val}) \in S_{\text{tree}}$, we have the following transitions in $T$. If the node $l \in \mathsf{nodes}(M')$ is such that $\mathsf{type}(l) = i$ and $f$ is the parent of $l$ in $M'$ then we have the transitions $t_1$, $t_2$ and $t_a$ for all $a \in \Gamma$ defined as follows:
  - ${}^\bullet t_1((M', \mathsf{val})) = 1, {}^\bullet t_1(lock) = 1, {}^\bullet t_1(p_{l,a}) = \mathbf{u}(a)$ for all $a \in \Gamma, {}^\bullet t_1(p) = 0$ for all the other places in $P$ and $t_1^\bullet(p_l^{source}) = 1, t_1^\bullet\left(p_f^{target}\right) = 1, t_1^\bullet\left((M'', \mathsf{val}')\right) = 1$ where $(M'', \mathsf{val}')$ is obtained from $(M', \mathsf{val})$ by removing the node $l$ and letting all its sub-trees become sub-trees of $f$. For all the other place $p \in P$ we have that $t_1^\bullet(p) = 0$;
  - for all $a \in \Gamma, {}^\bullet t_a(p_l^{source}) = 1, {}^\bullet t_a\left(p_f^{target}\right) = 1, {}^\bullet t_a(p_{l,a}) = 1$ and ${}^\bullet t_a(p) = 0$ for all the other places $p \in P, t_a^\bullet\left(p_l^{source}\right) = 1, t_a^\bullet\left(p_f^{target}\right) = 1, t_a^\bullet(p_{f,a}) = 1$ and $t_a^\bullet(p) = 0$ for all the other places $p \in P$;
  - ${}^\bullet t_2\left(p_l^{source}\right) = 1, {}^\bullet t_2\left(p_f^{target}\right) = 1 {}^\bullet t_2(p) = 0$ and for all the other places $p \in P, t_2^\bullet(lock) = 1, t_2^\bullet(p_{f,a}) = \mathbf{v}(a)$ for all $a \in \Gamma$, and $t_2^\bullet(p) = 0$ for all the other places $p \in P$;
- For any fusion rule $[_i\, \mathbf{u}[_j\, \mathbf{v} \to [_k\, \mathbf{w}$ and any $(M', \mathsf{val}) \in S_{\text{tree}}$, we have the following transitions in $T$. If the nodes $l, f \in \mathsf{nodes}(M')$ are such that $\mathsf{type}(l) = i$ and $\mathsf{type}(f) = j$ and $l, f$ have the same parent $e$ in $M'$ then we have in $T$ the transitions $t_1$, $t_2$, and $t_a$ for all $a \in \Gamma$ defined in a similar fashion than in the dissolution case, except that $(M'', \mathsf{val}')$ is obtained from $(M', \mathsf{val})$ by removing the node $l$ and letting the subtrees of $l$ become subtrees of $f$, and updating $\mathsf{type}(f)$ to $k$.

The transition $t_2$ in the case of dissolution and fusion non-deterministically stops the transfer of tokens from the places encoding the content of the discarded membrane. If a token is not transferred from a place $p_{l,a}$ associated to a dissolved membrane, then it remains in that place forever. In other words, all the markings $\mathbf{m}$ with $\mathbf{m}((M', \mathsf{val})) > 0$ for some place $(M', \mathsf{val})$ and which is reachable from $\mathbf{m}_0$ by firing a sequence of transitions that do not simulate the PB system is such that there is some $j \notin \mathsf{nodes}(M')$ and $a \in \Gamma$ such that $\mathbf{m}(p_{j,a}) > 0$. As a consequence, we can check if a configuration $(M', \mu)$ is reachable by testing if one of the markings $\mathbf{m}$ that satisfy the following properties is reachable from $\mathbf{m}_0$ in the Petri net: $\mathbf{m}((M', \mathsf{val})) = 1$ for some $\mathsf{val}$, $\mathbf{m}(lock) = 1$, for all $i \in \mathsf{nodes}(M')$, for all $a \in \Gamma, \mathbf{m}(p_{i,a}) = \mu(i)(a)$, and for the other places $p \in P$ we have that $\mathbf{m}(p) = 0$. □

Second, we consider PB systems extended with creation and cloning rules.

**Theorem 6** *The reachability problem for* $\{4{,}6\}$*-PB systems, i.e. PB systems extended with creation and cloning rules, is decidable.*

*Proof* Given a PB system $(\Gamma, N, M, R, \mu_0)$ extended with creation and cloning rules, we know that all the configurations in an execution from $(M, \mu_0)$ to a configuration $(M', \mu)$ have a tree structure with at most $|\mathsf{nodes}(M')|$ nodes, where $|\mathsf{nodes}(M')|$ is the cardinality of $\mathsf{nodes}(M')$. Indeed, there is no rule that allows to decrease the number of membranes in configurations. Hence, once a membrane is created it stays forever.

Hence, we have a bound on the number of nodes for the configurations we must consider when testing the reachability of a configuration, as in the proof of Theorem 2. As a consequence, the proof of Theorem 2 can be directly adapted to obtain the theorem. □

To conclude the section, notice that we can easily reduce (in polynomial time) the reachability problem for Petri net into the reachability problem for PB-systems. Indeed, each transition $t$ of a Petri net can be simulated with an internal rule $[_i^\bullet t \to [_i t^\bullet$. As a consequence, the reachability problem for PB-system extended with fusion/dissolution rules or creation/cloning rules has at least the complexity of the reachability problem for Petri nets, which is known to be EXPSPACE-hard and for which only non-primitive recursive algorithms are known.

## 5 Boundedness for extended PB systems

in this section we investigate the boundedness problem for extended PB systems. First, we show that undecidability holds if creation rules are allowed. Then, we show that the boundedness problem for extended PB systems without creation rules is decidable. That result is obtained by using the theory of well-structured transition systems.

The next theorem states that adding creation rules to PB systems leads to the undecidability of the boundedness problem.

**Theorem 7** (From Delzanno and Van Begin 2007) *The boundedness problem for* $\{4\}$*-PB systems, i.e. PB systems extended with creation rules, is undecidable.*

We now prove that the boundedness problem is decidable for extensions of PB systems if we disallow creation rules. We first notice that extended PB-systems with the Kruskal order are not well-structured. Indeed, consider only one type of membrane $i$ and a boundary rule $r = a[_i\ b \to c[_i\ d$. Now consider two configurations $(M, \mu)$ and $(M', \mu')$. The first one is composed of two nodes, the root and its unique child. The root contains the object $a$ and its child contains $b$. Hence, $r$ is applicable. The second configuration is similar to the first one except that there is an intermediate membrane between the root and its child. That intermediate membrane contains the object $c$, hence $r$ is not applicable and the condition $(ii)$ of the definition of wsts (monotonicity) does not hold.

Thus, we cannot directly use the theory of well-quasi ordering for extended PB systems and the order $\leq_K$ to solve the boundedness problem. Instead, we use another order, noted $\leq_D$, for which {3,5,6}-PB systems are strictly monotonic.

Assume two trees $M$ and $M'$ with $r$ and $r'$ as root, respectively. Assume also a quasi order $\sqsubseteq$ on nodes of trees.

**Definition 2** (*The order* $\leq_D$) We say that $M \leq_D M'$ iff there exists an injection $\rho : \mathsf{nodes}(M) \mapsto \mathsf{nodes}(M')$ such that $\rho(r) = r'$, for all $n \in \mathsf{nodes}(M), n \sqsubseteq \rho(n')$ and for all $n, n' \in \mathsf{nodes}(M)$, we have that $n'$ is a child of $n$ iff $\rho(n')$ is a child of $\rho(n)$.

In the case of configurations, the order $\sqsubseteq$ between labels of nodes is multiset inclusion as for $\leq_K$. For any pairs of configurations $c$ and $c'$, we use $c <_D c'$ to denote that $c \leq_D c'$ and $c' \not\leq_D c$.

Now, we say that a configuration $(M, \mu)$ is $b$-depth bounded if the depth (= max number of nested membranes) $d$ of $M$ is less of equal than $b$. Given an extended PB system $(\Gamma, N, M, R, \mu_0)$, let $b \in \mathbb{N}$ be the depth of $M$. Then, it is easy to check that all the reachable configurations are $b$-depth bounded. In the following, we say that the extended PB system is $b$-depth bounded and we restrict the set of configurations to $b$-depth bounded configurations. Based on this observation, we prove that $\leq_D$ is a wqo for the set of $b$-depth bounded configurations.

**Lemma 1** *Given an extended (b-depth bounded) PB system $\Pi$, the quasi order $\leq_D$ over (b-depth bounded) configurations of $\Pi$ is a wqo.*

*Proof* To prove the lemma, we annotate each node $n$ by its depth $\mathsf{Depth}(n)$. Then, $\leq_D$ corresponds to the Kruskal order where the order $\sqsubseteq$ on nodes is defined as follows: $n \sqsubseteq n'$ iff $\mathsf{Depth}(n) = \mathsf{Depth}(n'), \mathsf{type}(n) = \mathsf{type}(n')$ and $\mu(n) \preceq \mu(n')$. Since $\preceq$ is a wqo, the size of the set $N$ of possible membrane names/types and the depth of configurations are bounded, we get that $\sqsubseteq$ is a wqo. Hence, applying the Kruskal theorem (Kruskal 1960), we get that $\leq_D$ is a wqo. □

Thus, when reasoning on $b$-depth bounded configurations we can replace the tree embedding order $\leq_K$ with the tree immersion $\leq_D$. In Delzanno and Van Begin (2008) we have shown that {5,6}-PB systems are strictly monotonic w.r.t. $\leq_D$. We next show that rule (3) is also strictly monotonic.

**Lemma 2** {3,5,6}-*PB systems are strictly monotonic with respect to* $\leq_D$.

*Proof* Assume a $b$-depth bounded {3,5,6}-PB system and two ($b$-depth bounded) configurations $c = (M, \mu)$ and $c' = (M', \mu')$ such that $c \leq_D c'$. Assume also that the configuration $c'' = (M'', \mu'')$ is reachable from $c$ by applying a rule. We show that, by applying the same rule on $c'$ we get a configuration $c''' = (M''', \mu''')$ such that $c'' \leq_D c'''$. More precisely, from an injection $\rho$ that shows that $c \leq_D c'$ we build an injection $\rho'$ that shows that $c'' \leq_D c'''$. Furthermore, if $c <_D c'$ then $c'' <_D c'''$. In Delzanno and Van Begin (2008, Lemma 2, Sect. 4) we have shown that {5,6}-PB systems are strictly monotonic w.r.t. $\leq_D$. We next show that rule (3) is also strictly monotonic. □

*Dissolution rule.* Assume that a dissolution rule $[_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v} \cdot \delta$ is applied on a node $n \in \mathsf{nodes}(M)$ with $\mathsf{type}(n) = i$. Assume also that the membrane $m \in \mathsf{nodes}(M)$ is the father of $n$ in $M$. Then, the rule can also be applied on $\rho(n)$ since $(a)$ $i = \mathsf{type}(n) = \mathsf{type}(\rho(n))$ and $(b)$ $\mu(n) \preceq \mu'(\rho(n))$. Moreover, following the definition of $\rho$, we get that $(c)$ $\rho(m)$ is the father of $\rho(n)$ and $(d)$ $\mu(m) \preceq \mu'(\rho(m))$. Hence, from $(b)$ and $(d)$, we get that $\mu(m) \oplus (\mu(n) \ominus \mathbf{u}) \oplus \mathbf{v} \preceq \mu'(\rho(m) \oplus (\mu'(n) \ominus \mathbf{u}) \oplus \mathbf{v}$. As a consequence, we define the injection $\rho'$ as follows: $\rho'\ (n') = \rho(n')$ for all the nodes $n' \in \mathsf{nodes}(M)$ such that $n' \neq n$. Notice that the number of nodes always decreases by 1 when applying a fusion rule. Hence, if the number of nodes in $M$ is strictly smaller than in $M'$ then the number of nodes in $M''$ is strictly smaller than in $M'''$.. Furthermore, if $c$ contains a node $n' \neq n$ such that $\mu(n') \prec \mu'(\rho(n'))$ then $\mu''(n') \prec \mu'''(\rho'(n'))$ and if $\mu(n) \prec \mu'(\rho(n))$ then $\mu''(m) \prec \mu'''(m)$. Hence, we conclude that $c <_D c'$ implies that $c'' <_D c'''$..

From Lemma 1 and Lemma 2, we conclude that PB systems extended with dissolution, fusion, and cloning rules are strictly monotonic wsts.

**Proposition 1** *$\{3,5,6\}$-PB systems, i.e. PB-systems extended with dissolution, fusion and cloning rules, together with the order $\leq_D$ are strictly monotonic wsts.*

The algorithm of Finkel and Schnoebelen (2001) that solves the boundedness problem for wsts computes a finite prefix of the reachability tree. The unfolding of a branch is stopped as soon as from the current configuration $c$ either there is no configuration $c'$ with $c \rightarrow c'$, i.e., $c$ is a deadlock, or there is another configuration $c'$ in the same branch such that $c' \leq_D c$. The set of reachable configurations is infinite if and only if there is a leaf $c$ with a predecessor $c'$ such that $c' <_D c$. Since we can compute the successors for any configuration of extended PB systems and the order $\leq_D$ is decidable, we can instantiate that algorithm to PB systems extended with dissolution/fusion/cloning rules.

**Theorem 8** *The boundedness problem for $\{3,5,6\}$-PB systems, i.e. PB systems extended with dissolution, fusion and cloning rules, is decidable.*

Following Delzanno and Van Begin (2008), we can show that Transfer Petri nets can be encoded (in polynomial time) into PB systems extended with fusion and cloning rules. The following corollary then holds.

**Corollary 1** *The boundedness problem for $\{3,5,6\}$-PB systems has non-primitive complexity.*

In Delzanno and Van Begin (2008) we only give the intuition of the encoding of Transfer nets into PB-systems with fusion and cloning. For sake of completeness, we given next a formal definition of the encoding.

*Encoding Transfer nets* For each place $p$ we have a membrane name/type $p$. We also have a membrane name/type $\mathcal{N}$. A marking $m$ is encoded into a configuration composed of a root membrane of name/type $\mathcal{N}$ which has, for each place $p$, two children of name/type $p$. The first child of type $p$ contains only an object $w_p$ and is used to simulate transfers from the place $p$. The second child contains one object $ok_p$ and as many objects $\bullet$ as the number of tokens assigned by $m$ to $p$, i.e., it is used to encode the content of the place $p$. The root membrane contains an object that describes the state of the simulation: the system is ready to simulate a new transition or it is simulating a transition. Figure 6 shows how a transfer from place $p$ to $p'$ is simulated: the membrane of type $p$ encoding the content of the place $p$ is fused with the membrane of type $p'$ encoding the content of the place $p'$. Moreover, the other membrane of name/type $p$ is cloned and the new copy is used to encode the content of the place $p$ after the transfer.

Formally, the simulation works as follows. Let $p_1, \ldots, p_k$ be the set of places of $\mathcal{N}$. Then, for any transition $t$ of $\mathcal{N}$, to simulation transition $t$ we use objects: ${}^\bullet t_{p_1}, \ldots, {}^\bullet t_{p_{k+1}}, t_{\mathsf{T},1}, t_{\mathsf{T},2}, t_{\mathsf{T},3}, t_{\mathsf{T},4}, t^\bullet_{p_1}, \ldots, t^\bullet_{p_k}, t^\bullet_{p_{k+1}}$. Moreover, we have the object *idle* meaning that the system is ready to simulate a transition.

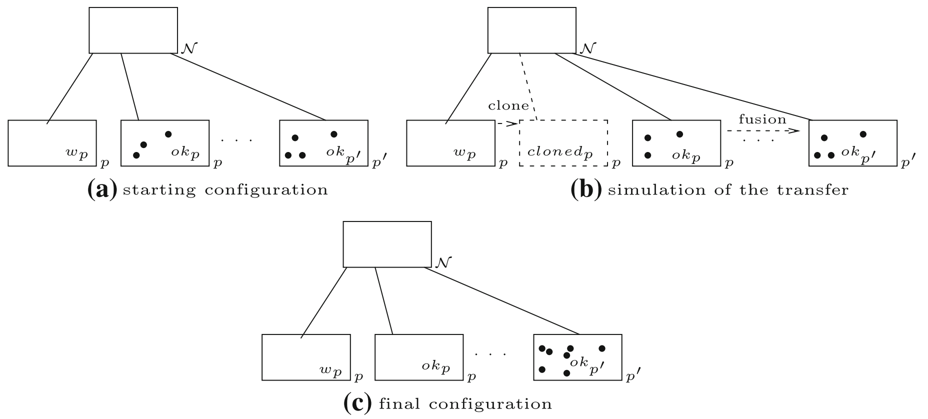

**Fig. 6** Simulation of a transfer with a PB system extended with fusion/cloning rules. *Rectangles* represents membranes

Consider a transition $t$. The first part of the simulation consists to remove the objects in ${}^\bullet t$. The next rule initializes the simulation of $t$:

$$[_{\mathcal{N}}\ idle \rightarrow [_{\mathcal{N}}\ {}^\bullet t_{p_1}$$

Then, for any $i$ such that $1 \leq i \leq k$ we have the following rule that removes ${}^\bullet t(p(i))$

$${}^\bullet t_{p_i} [_{p_i}\ ok_{p_i} \cdot \bullet^{{}^\bullet t(p_i)} \rightarrow^\bullet t_{p_{i+1}} [_{p_i}\ ok_{p_i}$$

where $\bullet^{{}^\bullet t(p_i)}$ is the concatenation of ${}^\bullet t(p_i)$ symbols $\bullet$.

If there is no transfer, i.e. $\mathsf{T}(t)$ is undefined, then we have the rule

$$[_{\mathcal{N}}\ {}^\bullet t_{p_{i+1}} \rightarrow t^\bullet_{p_1}$$

that allows to start the simulation of adding the tokens in $t^\bullet$. Otherwise, the transfer part is simulated in several steps with the following rules. Assume that $\mathsf{T}(t) = (p_j, p_l)$. Then, we also have the objects $c_{p_j}$, $cloned_{p_j}$,$fused_{p_l}$ and $f_{p_j p_l}$. The first two rules clone the membrane of type $p_j$ containing the object $w_{p_j}$:

$${}^\bullet t_{p_{k+1}} [_{p_j}\ w_{p_j} \rightarrow t_{\mathsf{T},1} [_{p_j}\ c_{p_j}$$

$$[_{p_j}\ c_{p_j} \rightarrow [_{p_j}\ w_{p_j} [_{p_j}\ cloned_{p_j}$$

The three next rules fuse the membranes encoding the content of the places $p_j$ and $p_l$ into a unique membrane of type $p_l$ (encoding the content of the place $p_l$ after the transfer). Those rules can be applied only after the two previous ones.

$$t_{\mathsf{T},1} [_{p_j}\ cloned_{p_j} \rightarrow t_{\mathsf{T},2} [_{p_j}\ cloned_{p_j}$$

$$t_{\mathsf{T},2}\ [_{p_j}\ ok_{p_j} \rightarrow t_{\mathsf{T},3} [_{p_j}\ f_{p_j p_l}$$

$$[_{p_j}\ f_{p_j p_l} [_{p_l}\ ok_{p_l} \rightarrow [_{p_l}\ fused_{p_l}$$

Finally, the cloned membrane of name/type $p_j$ becomes the membrane encoding the content of the place $p_j$ and the result of the fusion encodes the place $p_l$.

$$t_{\top,3}\big[_{p_l}\, fused_{p_l} \to t_{\top,4}\big[_{p_l}\, ok_{p_l}$$
$$t_{\top,4}\big[_{p_j}\, cloned_{p_j} \to t^{\bullet}_{p_1}\big[_{p_j}\, ok_{p_j}$$

Adding the tokens in $t^{\bullet}$ is simulated by the following rules. For $1 \leq i \leq k+1$, we have the rule

$$t^{\bullet}_{p_i}\big[_{p_i}\, ok_{p_i} \to t^{\bullet}_{p_{i+1}}\big[_{p_i}\, ok_{p_i} \cdot \bullet^{t^{\bullet}(p_i)}$$

The simulation of $t$ is finished when firing the rule

$$\big[_{\mathcal{N}}\, t^{\bullet}_{p_{k+1}} \to \big[_{\mathcal{N}}\, idle$$

By construction, we have that $\mathcal{N}$ has a finite set of reachable configurations if and only if its encoding into PB systems extended with fusion and cloning rules has a finite set of reachable configurations. Finally, following Theorem 2 and since the encoding of $\mathcal{N}$ is computable in polynomial time, we conclude that the theorem holds.

## 6 Coverability for extended PB systems

We show in this section that, similarly to the boundedness problem, the $\leq_K$-coverability problem for PB systems extended with creation rules is undecidable and it is decidable for extended PB systems without creation rules. Again, the decidability result is obtained by using the theory of well-structured transition systems.

**Theorem 9** *The $\leq_K$-coverability problem is undecidable for $\{4\}$-PB systems; i.e. PB systems extended with creation rules.*

*Proof* The proof is similar to the proof of Theorem 5 and we reuse the same encoding of counter machines into PB systems extended with creation rules. We reduce the $\sqsubseteq$-coverability problem for a counter machine and a location $\ell$ to the $\leq_K$-coverability problem where the configuration $(M, \mu)$ to cover is composed of one node of type 0 containing the only symbol $\ell$. □

We now apply the theory of well-structured transition systems (Abdulla et al. 1996; Finkel and Schnoebelen 2001) to solve the $\leq_K$-coverability problem for $\{3,5,6\}$-PB systems. Since $\{3,5,6\}$-PB systems together with wqo $\leq_K$ do not form a wsts, we cannot apply directly the results of Abdulla et al. (1996) and Finkel and Schnoebelen (2001). Instead, we reuse the wqo $\leq_D$ introduced into the previous section. In the remainder of the section, we show how to instantiate the algorithm proposed in Abdulla et al. (1996) and Finkel and Schnoebelen (2001) to solve the $\leq_D$-coverability problem. Then, we show how to use that result to solve the $\leq_K$-coverability problem.

### 6.1 Solving the $\leq_D$-coverability problem

The algorithm in Abdulla et al. (1996) and Finkel and Schnoebelen (2001) that solves the coverability problem manipulates *upward-closed sets* of configurations. Remember that we restrict the set of configurations of a $b$-depth bounded $\{3,5,6\}$-PB system to $b$-depth bounded configurations. A set of configurations $U$ is upward-closed if and only if for any pair of configurations $c$, $c'$, we have that $c \in U$ and $c \leq_D c'$ implies that $c' \in U$. Given a set of $b$-depth bounded configurations $F, F\uparrow$ denotes the upward-closed set of $b$-depth bounded configurations $\{c \mid \exists c' \in F : c' \leq_D c\}$. If $F$ is finite then we say that it is a *finite basis* of $F\uparrow$. Notice that a finite basis may contain useless elements, ie. configurations $c$ such that there exists another configuration $c'$ with $c' <_D c$. Since $\leq_D$ is a wqo over $b$-depth bounded configurations, the following proposition holds:

**Proposition 2** *Any upward-closed set of b-depth bounded configurations have at least one finite basis.*

More precisely, the algorithm of Abdulla et al. (1996) and Finkel and Schnoebelen (2001) manipulates upward-closed sets towards one of their finite basis. To instantiate that algorithm we need thus to show how to compute the operations needed by that algorithm on that symbolic representation. Given an upward-closed set $U$, let $\mathsf{Pre}(U) = \{c \mid c \rightarrow c', c' \in U\}$. Roughly speaking, the algorithm computes the sequence of upward-closed sets $(\mathcal{P}_i)_{i\geq 0}$ such that $\mathcal{P}_0 = U$ and $\forall i \geq 1, \mathcal{P}_i = \mathcal{P}_{i-1} \cup \mathsf{Pre}(\mathcal{P}_{i-1})$ until it converges. Hence, we need an algorithm that computes, for any upward-closed set $U$, a finite basis of the set $\mathsf{Pre}(U)$. Before exhibiting such an algorithm, we show that a finite basis of the intersection of two upward-closed sets is computable[2].

**Lemma 3** *There exists an algorithm that builds, given two b-depth bounded configurations c and $c'$ of a b-depth bounded extended PB system $\Pi$, a finite basis of the upward-closed set of b-depth bounded configurations $\{c\}\uparrow \cap \{c'\}\uparrow$.*

*Proof* Given two $b$-depth bounded configurations $c = (M, \mu)$ and $c' = (M', \mu')$, we show here how to build a finite set of configurations $\mathcal{M}$ such that $\mathcal{M}\uparrow = \{(M,\mu)\}\uparrow \cap \{(M',\mu')\}\uparrow$. We assume that $\mathsf{nodes}(M) \cap \mathsf{nodes}(M') = \emptyset$.

Let $r$ and $r'$ be the root node of $M$ and $M'$, respectively. We define $\Pi$ as the set of partial injection $\rho : \mathsf{nodes}(M) \mapsto \mathsf{nodes}(M')$ such that $(i)$ $\rho(r) = r'$ and $(ii)$ for all $n, n' \in \mathsf{nodes}(M)$, $n$ is the parent of $n'$ and $\rho(n')$ is defined implies that $\rho(n)$ is defined and $\rho(n)$ is the parent of $\rho(n')$. Notice that the set $\Pi$ is finite and computable.

Then, for all $\rho \in \mathcal{M}$, we defined the $b$-depth bounded configuration $(M_\rho, \mu_\rho)$ as follows. The set of nodes of the tree $M_\rho$ is

$$\begin{aligned}\mathsf{nodes}(M_\rho) = & \{\{n\} \mid n \in \mathsf{nodes}(M), \rho(n) \text{ is undefined}\} \\ & \cup \left\{\{n\} \mid n \in \mathsf{nodes}(M'), \rho^{-1}(n) \text{ is undefined}\right\} \\ & \cup \{\{n, n'\} \mid n \in \mathsf{nodes}(M), n' \in \mathsf{nodes}(M'), \rho(n) = n'\}\end{aligned}$$

where $\rho^{-1}$ is the inverse of $\rho$.

A node $n' \in \mathsf{nodes}(M_\rho)$ is a child of a node $n \in \mathsf{nodes}(M_\rho)$ iff there exist $m \in n, m' \in n'$ and $m'$ is a child of $m$ either in $M$ or in $M'$.

Finally, $\mu_\rho$ is defined as follows. For all $m \in \mathsf{nodes}(M_\rho)$, one of the following holds:

- $m = \{n\}$ with $n \in \mathsf{nodes}(M)$ and $\mu_\rho(m) = \mu(n)$;
- $m = \{n\}$ with $n \in \mathsf{nodes}(M')$ and $\mu_\rho(m) = \mu'(n)$;
- $m = \{n, n'\}$ with $n \in \mathsf{nodes}(M), n' \in \mathsf{nodes}(M')$ and for all $\gamma \in \Gamma$ we have that $\mu_\rho(m)(\gamma) = max(\mu(n)(\gamma), \mu'(n')(\gamma))$.

Again, notice that for any $\rho \in \mathcal{M}$ the $b$-depth bounded configuration $(M_\rho, \mu_\rho)$ is computable. Finally, notice that we have that $\cup_{\rho\in\mathcal{M}}\{(M_\rho, \mu_\rho)\}\uparrow = \{(M,\mu)\}\uparrow \cap \{(M',\mu')\}\uparrow$. □

[2] Notice that the intersection of upward-closed sets is always an upward-closed set.

In Delzanno and Van Begin (2008) we have shown how to compute a finite basis of $\mathsf{Pre}(U)$ for {5,6}-PB system. We next show how to extend the construction to handle the dissolution rule.

**Proposition 3** *There exists an algorithm that builds, given a b-depth bounded {3}-PB system Π and a finite basis of an upward closed set U of b-depth bounded configurations, a finite basis of the set* $\mathsf{Pre}(U)$.

*Proof* Let $F$ be a finite basis of $U$ and $F'$ be the finite basis of $\mathsf{Pre}(U)$ computed by the algorithm. The algorithm is defined by cases as follows.

Assume a dissolution rule $[_i\, \mathbf{u} \rightarrow [_i\, \mathbf{v} \cdot \delta$ and a configuration $(M, \mu) \in F$. We consider two cases:

- the parent of the dissolved membrane appears explicitly. Since there is no constraint on the parent, it may be any node in $\mathsf{nodes}(M)$ (with a depth at most $b - 1$). Hence, for any node $n \in \mathsf{nodes}(M)$, we have in $F'$ the configurations $(M', \mu')$ such that $M'$ is obtained from $M$ by adding a new node $n'$ as a child of $n$ with $\mathsf{type}(n') = i$ and each subtree of $n$ in $M$ either stay a subtree of $n$ in $M'$ or becomes a subtree of $n'$. The function $\mu'$ is defined as follows: $\mu'(n)$ and $\mu'(n')$ are such that $\mathbf{u} \preceq \mu'(n')$, $\mu'(n) \oplus (\mu'(n') \ominus \mathbf{u}) = \mu(n) \ominus \mathbf{v}$; and for all $n'' \in \mathsf{nodes}(M')$ such that $n'' \notin \{n, n'\}$ we have that $\mu'(n'') = \mu(n)$;
- the parent of the dissolved membrane does not appear explicitely. In that case a new membrane containing the objects of $\mathbf{u}$ must appear in the predecessor configurations and that membrane may appear at any depth, bounded by $b$. In other words, $F'$ contains all the $b$-depth bounded configurations $(M', \mu')$ such that $M'$ is obtained from $M$ by adding nodes $n_1, \ldots, n_k (k \geq 2)$ such that for some node $n_0 \in \mathsf{nodes}(M)$ we have that for all $1 \leq i \leq k - 1$, $n_i$ is a child of $n_{i\text{-}1}$ and $\mathsf{nodes}(n_i)$ is any type, $n_k$ is the child of $n_{k-1}, \mathsf{type}(n_k) = i$. The function $\mu'$ is defined as follows: $\mu'(n_k) = \mathbf{u}$, for all $1 \leq i < k, \mu'(n_i) = \epsilon$, and $\mu'(n) = \mu(n)$ for all $n \in \mathsf{nodes}(M)$.

Notice that the union of upward-closed sets is computed by making the union of their finite basis, and the convergence of the sequence $(\mathcal{P}_i)_{i \geq 0}$ can be decided by testing $\leq_D$ on elements of finite basis as follows: given two finite set of $b$-depth bounded configurations $B_1$ and $B_2$, we have that

$$B_1 \uparrow \subseteq B_2 \uparrow \text{ iff } \forall c_2 \in B_2, \exists c_1 \in B_1, c_1 \leq_D c_2.$$

Since $\leq_D$ is decidable, from Lemma 5 and Proposition 1 we get the decidability of the $\leq_D$-coverability problem.

**Proposition 4** *The $\leq_D$-coverability problem for* {3,5,6}*-PB systems, i.e. PB systems extended with dissolution/fusion/cloning rules, is decidable.*

By reusing the encoding of Transfer nets into PB systems with fusion and cloning rules given in the proof of Theorem 5 we can reduce (in polynomial time) the $\preceq$-coverability problem for Transfer nets to the $\leq_D$-coverability problem for PB systems extended with fusion and cloning rules. Then, following Theorem 2 we conclude that the next theorem holds. The $\leq_D$-coverability problem for {3,5,6}-PB systems has a nonprimitive recursive complexity.

## 6.2 Decidability of the $\leq_K$-coverability problem

We now show how to reduce the $\leq_K$-coverability problem to the $\leq_D$-coverability problem. To achieve that goal we define for any $b$-depth bounded configuration $c = (M, \mu)$ the set

$\mathcal{M}(c,b)$. It is the smallest set of $b$-depth bounded configurations that satisfies the following conditions:

1. $c \in \mathcal{M}(c,b)$;
2. if the $b$-depth bounded configuration $(M',\mu') \in \mathcal{M}(c,b)$, then any $b$-depth configurations ($M''$, $\mu''$) built from $\mathcal{M}(,c,b)$ such that $M''$ is obtained from $M'$ by chosing a node $n \in Nodes(M')$ with father $n'$ and adding an intermediate node $m$ between $n'$ and $n$, ie. $n'$ is the father of $m$ and $n$ is the only son of $m$. The mapping $\mu''$ is defined such that $\mu''(m) = \emptyset$ and $\mu''(m') = \mu'$ $(m')$ for all node $m' \in Nodes(M)$;
3. if $(M',\mu') \in \mathcal{M}(c,b)$ is a $(b-1)$-depth bounded configuration with root node $r$, then the configuration $(M'',\mu'') \in \mathcal{M}(c,b)$ where $M''$ is built from $M'$ by adding a new node $m$ that has $r$ as son; ie. $m$ is the root node of $M''$. The mapping $\mu''$ is defined such that $\mu''(m) = \emptyset$ and $\mu''(m') = \mu'(m')$ for all node $m' \in Nodes(M)$.

Notice that the points 2 and 3 increase the depth of generated configurations. Furthermore they are applied on configurations with a depth lesser than $b$. Hence, we conclude to the next lemma.

**Lemma 4** *For any b-depth bounded configuration c, the set $\mathcal{M}(c,b)$ is finite and contains only configurations with a depth bounded by b.*

In other words, the configurations in $\mathcal{M}(c,b)$ have a similar structure than $c$ except that the nodes of $M$ are potentially at a larger depth (bounded by $b$). Hence the following lemma holds.

**Lemma 5** *For any b-depth bounded configurations c and $c'$, we have that $c' \leq_K c$ if and only if there exists a b-depth bounded configuration $c'' \in \mathcal{M}(c,b)$ such that $c' \leq_D c''$.*

Since the set $\mathcal{M}(c,b)$ is finite and computable, we can solve the $\leq_K$-coverability problem for a $b$-depth bounded {3,5,6}-PB system $\Pi$ and a $b$-depth bounded configuration ($M$, $\mu$) by solving $\leq_D$-coverability problems for $\Pi$ and configurations in $\mathcal{M}(c,b)$.

**Theorem 11** *The $\leq_K$-coverability problem for {3,5,6}-PB systems, i.e. PB systems extended with dissolution/fusion/cloning rules, is decidable.*

By reusing the encoding of Transfer nets into PB systems with fusion and cloning rules given in the proof of Theorem 5 we can reduce in polynomial time the $\preceq$-coverability problem for Transfer nets to the $\leq_K$-coverability problem for PB systems extended with fusion and cloning rules. Then, following Theorem 2 we conclude that the next theorem holds.

**Theorem 12** *The $\leq_K$-coverability problem for {3,5,6}-PB systems has a nonprimitive recursive complexity.*

## 7 Conclusion

In this paper we have studied verification problems for extensions of PB-systems with creation, dissolution, fusion and cloning of membranes. In the resulting model reachability of a configuration turns out to be undecidable while coverability of a configuration (w.r.t. tree embedding) and boundedness of the state-space are both decidable. Creation rules make coverability and boundedness undecidable. However, the addition of creation rules is not sufficient to make reachability undecidable. Reachability becomes undecidable when combining creation with rules that delete membranes like dissolution and fusion. As future work we plan to investigate the impact of movement operations, i.e., rules that move membranes across configurations, on the decidability results for the extensions of PB-systems we have studied in this paper.

## References

Abdulla PA, Čerāns K, Jonsson B, Yih-Kuen T (1996) General decidability theorems for infinite-state systems. In: Proceedings of the 11th annual IEEE symposium on logic in computer science, LICS '96, New Brunswick, New Jersey, 27–30 July 1996. IEEE Computer Society Press, Westbury, pp 313–321

Alhazov A, Freund R, Riscos-Núñez A (2006) Membrane division, restricted membrane creation and object complexity in P systems. Comput Math 83(7):529–547

Bernardini F, Manca V (2003) P systems with boundary rules. In: Proceedings of the international workshop on membrane computing, WMC '03, Curtea de Arges, Romania, August 19–23, 2002, Lecture notes in computer science 2597. Springer, Berlin, pp 107–118

Besozzi D, Zandron C, Mauri G, Sabadini N (2001) P systems with gemmation of mobile membranes. In: Proceedings of the 7th Italian conference on theoretical computer science, ICTCS '01, Torino, Italy, October 4–6, 2001. Lecture notes in computer science 2202. Springer, Berlin, pp 136–153

Besozzi D, Mauri G, Păun G, Zandron C (2003) Gemmating P systems: collapsing hierarchies. Theor Comput Sci 296(2):253–267

Bezem M, Klop JW, de Vrijer R (2003) Term rewriting systems. Cambridge University Press, Cambridge

Dal Zilio S, Formenti E (2004) On the dynamics of PB systems: a Petri net view. In: Proceedings of the international workshop on membrane computing, WMC 2003, Tarragona, Spain, July 17–22, 2003, Revised Papers. Lecture notes in computer science 2933. Springer, Berlin, pp 153–167

Delzanno G, Montagna R (2007) On reachability and spatial reachability in fragments of bioambients. Electron Notes Theor Comput Sci 171(2):69–79

Delzanno G, Van Begin L (2007) On the dynamics of PB systems with volatile membranes. In: Proceedings of the International Workshop on Membrane Computing, WMC 2007, Tarragona, Spain, July 17–22, 2007. Lecture notes in computer science 4860. Springer, Berlin, pp 240–256

Delzanno G, Van Begin L (2008) A Biologically inspired model with fusion and clonation of membranes. In: Proceedings of the 7th international conference on unconventional computing, UC '08, Vienna, Austria, August 25–28, 2008. Lecture notes in computer science 5204. Springer, Berlin, pp 64–82

Dickson LE (1913) Finiteness of the odd perfect and primitive abundant numbers with distinct factors. Am J Math 35:413–422

Finkel A, Schnoebelen Ph (2001) Well-structured transition systems everywhere! Theor Comput Sci 256(1–2):63–92

Franco G, Manca V (2004) A membrane system for the leukocyte selective recruitment. In: Proceedings of the international workshop on membrane computing, WMC '03, Tarragona, Spain, July 17–22, 2003. Lecture notes in computer science 2933. Springer, Berlin, pp 181–190

Ibarra OH, Dang Z, Egecioglu Ö (2004) Catalytic P systems, semilinear sets, and vector addition systems. Theor Comput Sci 312(2-3):379–399

Kosaraju SR (1982) Decidability of reachability in vector addition systems. In: Proceedings of the 14th annual ACM symposium on theory of computing, STOC '82, May 5–7, 1982. ACM, San Francisco, CA, pp 267–281

Kruskal JB (1960) Well-quasi ordering, the tree theorem, and Vazsonyi's conjecture. Trans Am Math Soc 95:210–225

Li C, Dang Z, Ibarra OH, Yen H-C (2005) Signaling P systems and verification problems. In: Proceedings of the 32nd international colloquium on automata, languages and programming, ICALP '05. Lisbon, Portugal, July 11–15, 2005. Lecture notes in computer science 3580. Springer, Berlin, pp 1462–1473

Mayr EW (1984) An algorithm for the general Petri net reachability problem. SIAM J Comput 13(3):441–460

Minsky M (1967) Computation: finite and infinite machines. Prentice-Hall, Englewood Cliffs

Păun Gh (2000) Computing with membranes. J Comput Syst Sci 61(1):108–143

Păun Gh (2001) P systems with active membranes: attacking NP-complete problems. J Autom Lang Comb 6(1):75–90

Păun A, Popa B (2006) P systems with proteins on membranes. Fundam Inform 72(4):467–483

Păun Gh, Suzuki Y, Tanaka H, Yokomori T (2004) On the power of membrane division in P systems. J Theor Comput Sci 324(1):61–85

Petre I, Petre L (1999) Mobile ambients and P-systems. J Univers Comput Sci 9:588–598

Petri CA (1962) Kommunikation mit Automaten. Ph.D. Thesis, University of Bonn

Regev A, Panina EM, Silverman W, Cardelli L, Shapiro E (2004) BioAmbients: an abstraction for biological compartments. J Theor Comput Sci 325(1):141–167

Schnoebelen Ph (2002) Verifying lossy channel systems has nonprimitive recursive complexity. Inform Process Lett 83(5):251–261

Reisig W (1985) Petri nets—an introduction. Springer, Berlin

The P Systems Webpage http://www.ppage.psystems.eu/

Zavattaro G (2009) Reachability analysis in bioambients. Electron Notes Theor Comput Sci 227:179–193