Decidability of termination problems for sequential P systems with active membranes

Michal Kováč

Faculty of Mathematics, Physics and Informatics, Comenius University

Abstract. We study variants of P systems that are working in the sequential mode. Basically, they are not universal, but there are possible extensions that can increase the computation power. Extensions that relate to a zero-check, are often universal. P systems with an ability to create new membranes are a rare exception as they are known to be universal even in the sequential mode without using a zero-check. In this paper we show somewhat surprising result that the existence of an infinite computation for sequential P systems with active membranes is decidable. The standard construction of coverability tree is extended to provide an algorithm for detecting infinite loops. In addition, we show that the existence of a halting computation is undecidable as it can be reduced to reachability of register machines.

1 Introduction

Membrane systems (P systems) were introduced by Păun (see [1]) as distributed parallel computing devices inspired by the structure and functionality of cells. One of the objectives is to relax the condition of using the rules in a maximally parallel way in order to find more realistic P systems from a biological point of view. In sequential P systems, only one rewriting rule is used in each step of computation. Without priorities, they are equivalent to vector addition systems [2], hence not universal. However priorities, inhibitors and other modifications can increase the computation power. It seems that there is a link between universality and ability to zero-check [3].

In this paper we study a variant where universality can be achieved without checking for zero by allowing membranes to be created unlimited number of times [2]. Such P systems are called active P systems. Contrary, if we place a limit on the number of times a membrane is created, such active P system is only equivalent to vector addition systems, hence not universal.

In the section 2 we will recall some basic notions from formal languages, multisets and graph theory. Then in the section 3 we will introduce membrane structure and formally define membrane configuration and active P system because standard definitions are not convenient for our formal proofs.

The section 4 contains two main results. The existence of an infinite computation is surprisingly[1] shown to be decidable. On the other hand, the existence of a halting computation is shown to be undecidable.

[1] At first sight it seems to relate to the Rice's theorem, but it is not the case.

2 Preliminaries

Here we recall several notions from the classical theory of formal languages.

An **alphabet** is a finite nonempty set of symbols. Usually it is denoted by Σ. A **string** over an alphabet is a finite sequence of symbols from the alphabet. We denote by Σ^* the set of all strings over an alphabet Σ. By $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ we denote the set of all nonempty strings over Σ. A **language** over the alphabet Σ is any subset of Σ^*.

The number of occurrences of a given symbol $a \in \Sigma$ in the string $w \in \Sigma^*$ is denoted by $|w|_a$. $\Psi_\Sigma(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ is called a Parikh vector associated with the string $w \in \Sigma^*$, where $\Sigma = \{a_1, a_2, \ldots a_n\}$. For a language $L \subseteq \Sigma^*$, $\Psi_\Sigma(L) = \{\Psi_\Sigma(w) | w \in L\}$ is the Parikh mapping associated with L. If FL is a family of languages, PsFL is denoted the family of Parikh images of languages in FL.

A multiset over a set Σ is a mapping $M : \Sigma \to \mathbb{N}$. We denote by $M(a), a \in \Sigma$ the multiplicity of a in the multiset M. The **support** of a multiset M is the set $supp(M) = \{a \in \Sigma | M(a) \geq 1\}$. It is the set of items with at least one occurrence. A multiset is **empty** when its support is empty. A multiset M with finite support $\{a_1, a_2, \ldots, a_n\}$ can be represented by the string $a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)}$. We say that multiset M_1 is included in multiset M_2 if $\forall a \in supp(M_1) : M_1(a) \leq M_2(a)$. We denote it by $M_1 \subseteq M_2$. If $M_1 \subseteq M_2$, the **difference** of two multisets $M_2 - M_1$ is defined as a multiset where $\forall a \in supp(M_2) : (M_2 - M_1)(a) = \max(M_2(a) - M_1(a), 0)$. The **union** of two multisets $M_1 \cup M_2$ is a multiset where $\forall a \in supp(M_1) \cup supp(M_2) : (M_1 \cup M_2)(a) = M_1(a) + M_2(a)$. The product of multiset M with natural number $n \in \mathbb{N}$ is a multiset where $\forall a \in supp(M) : (n \cdot M)(a) = n \cdot M(a)$.

Next, we recall notions from graph theory.

A **rooted tree** is a tree, in which a particular node is distinguished from the others and called the root node. Let T be a rooted tree. We will denote its root node by r_T. Let d be a node of $T \setminus \{r_T\}$. As T is a tree, there is a unique path from d to r_T. The node adjacent to d on that path is also unique and is called a **parent node** of d and is denoted by $parent_T(d)$. We will denote the set of nodes of T by $V(T)$ and set of its edges by $E(T)$. Let T_1, T_2 be rooted trees. A bijection $f : V(T_1) \to V(T_2)$ is an **isomorphism** iff $\{(f(u), f(v)) | (u, v) \in E(V(T_1))\} = E(V(T_2))$ and $f(r_{T_1}) = r_{T_2}$.

3 Active P systems

The fundamental ingredient of a P system is the **membrane structure** (see [4]). It is a hierarchically arranged set of membranes, all contained in the **skin membrane**. Each membrane determines a compartment, also called region, which is the space delimited from above by it and from below by the membranes placed directly inside, if any exists. Clearly, the correspondence membrane – region is one-to-one, that is why we sometimes use interchangeably these terms. Membrane structure can be also viewed as a rooted tree with the skin membrane as the root node.

A P system consists of a membrane structure with a multiset of objects and a set of rewriting rules in every membrane. Objects can be transformed into other objects and sent through a membrane according to given rules.

In this paper we work with P systems with active membranes (Active P systems). The rules can modify the membrane structure by dissolving and creating new membranes. That is why we will define the configuration to include the membrane structure as well.

Let Σ be a set of objects. Recall that $\mathbb{N}^{\Sigma}$ contains all multisets of objects from Σ. **Membrane configuration** is a tuple (T, l, c), where:

- T is a rooted tree,
- $l \in \mathbb{N}^{V(T)}$ is a mapping that assigns for each node of T a number (label), where $l(r_T) = 1$, so the skin membrane is always labeled with 1,
- $c \in (\mathbb{N}^{\Sigma})^{V(T)}$ is a mapping that assigns for each node of T a multiset of objects from Σ, so it represents the contents of the membrane.

Active P system is a tuple $(\Sigma, C_0, R_1, R_2, \ldots, R_m)$, where:

- Σ is a set of objects,
- C_0 is initial membrane configuration,
- $R_1, R_2, \ldots R_m$ are finite sets of rewriting rules associated with the labels $1, 2, \ldots, m$ and can be of forms:
  - $u \to w$, where $u \in \Sigma^+$, $w \in (\Sigma \times \{\cdot, \uparrow, \downarrow_j\})^*$ and $1 \leq j \leq m$,
  - $u \to w\delta$, where $u \in \Sigma^+$, $w \in (\Sigma \times \{\cdot, \uparrow, \downarrow_j\})^*$ and $1 \leq j \leq m$,
  - $u \to [_j v]_j$, where $u \in \Sigma^+, v \in \Sigma^*$ and $1 \leq j \leq m$.

Although rewriting rules are defined as strings, u, v and w represent multisets of objects from Σ. For the first two forms, each rewriting rule may specify for each object on the right side, whether it stays in the current region (we will omit the symbol $\cdot$), moves through the membrane to the parent region ($\uparrow$) or to a specific child region ($\downarrow_j$, where j is a label of a membrane). If there are more child membranes with the same label, one is chosen nondeterministically. We denote these transfers with an arrow immediately after the symbol. An example of such rule is the following: $abb \to ab \downarrow_2 c \uparrow c\delta$. Symbol δ at the end of the rule means that after the application of the rule, the membrane is dissolved and its contents (objects, child membranes) are propagated to the parent membrane. Active P systems differs from classic (passive) P systems in ability to create new membranes by rules of the third form. Such rule will create new child membrane with a given label j and a given multiset of objects v as its contents.

For active P system $(\Sigma, C_0, R_1, R_2, \ldots, R_m)$, configuration $C = (T, l, c)$, membrane $d \in V(T)$ the rule $r \in R_{l(d)}$ is **applicable** iff:

- $r = u \to w$ and $u \subseteq c(d)$ and $\forall (a, \downarrow_k) \in w \exists d_2 \in V(T) : l(d_2) = k \wedge parent(d_2) = d$,
- $r = u \to w\delta$ and $u \subseteq c(d)$ and $\forall (a, \downarrow_k) \in w \exists d_2 \in V(T) : l(d_2) = k \wedge parent(d_2) = d$ and $d \neq r_T$,
- $r = u \to [_j v]_j$ and $u \subseteq c(d)$.

In this paper we assume only sequential systems, so in each step of the computation, there is one rule nondeterministically chosen among all applicable rules in all membranes to be applied.

A **computation step** of P system is a relation $\Rightarrow$ on the set of configurations such that $C_1 \Rightarrow C_2$ holds iff there is an applicable rule in a membrane in C_1 such that applying that rule can result in C_2.

An **infinite computation** of a P system is an infinite sequence of configurations $\{C_i\}_{i=0}^{\infty}$, where $\forall i : C_i \Rightarrow C_{i+1}$.

A **finite computation** of a P system is a finite sequence of configurations $\{C_i\}_{i=0}^{n}$, where $\forall i : C_i \Rightarrow C_{i+1}$.

A **halting computation** of a P systems is a finite computation $\{C_i\}_{i=0}^{n}$, where there is no applicable rule in the last configuration C_n.

The P system can work in generating or in accepting mode. For the generating mode there are two possible ways of assigning a result of a computation:

1. By considering the multiplicity of objects present in a designated membrane in a halting configuration. In this case we obtain a vector of natural numbers. We can also represent this vector as a multiset of objects or as Parikh image of a language.
2. By concatenating the objects which leave the system, in the order they are sent out of the skin membrane (if several symbols are expelled at the same time, then any ordering of them is considered). In this case we generate a language.

The result of a single computation is clearly only one multiset or a string, but for one initial configuration there can be multiple possible computations. It follows from the fact that there can be more than one applicable rule in each configuration and they are chosen nondeterministically. For the accepting mode the input multiset is inserted in the skin membrane and it is accepted if and only if a given accepting configuration can be reached[2].

For the simplicity of proofs it is convenient to introduce a variant with a global limit upon the membrane structure. We achieve this by restricting the rule application such that if the rule would result in a structure exceeding the limit, the rule will not be applicable.

Active P system with a limit on the total number of membranes is a tuple $(\Sigma, L, C_0, R_1, R_2, \ldots, R_m)$, where $(\Sigma, C_0, R_1, R_2, \ldots, R_m)$ is an active P system and $L \in \mathbb{N}$ is a limit on the total number of membranes. Anytime during the computation, a configuration (T, l, c) is not allowed to have more than L membranes, so the following invariant holds: $|V(T)| \leq L$.

This is achieved by adding a constraint for rule of the form $r = u \rightarrow [_k v]_k$, which is defined to be applicable iff $u \subseteq c(d)$ and $|V(T)| < L$. If the number of membranes is equal to L, there is no space for newly created membrane, so in that case such rule is not applicable.

4 Termination problems

In this section we recall the halting problem for Turing machines. The problem is to determine, given a deterministic Turing machine and an input, whether the Turing machine running on that input will halt. It is one of the first known undecidable problems. On the other hand, for non-deterministic machines, there are two possible meanings for halting. We could be interested either in:

- whether there exists an infinite computation (the machine can run forever), or
- whether there exists a finite computation (the machine can halt).

We will prove the (un)decidability of these problems on active P systems with limit on the total number of membranes. The results are quite interesting, because:

Theorem 1. *Sequential active P systems with limit on the total number of membranes are universal.*

Proof. The proof of this theorem for sequential active P systems in [2] uses simulation of register machines and during the simulation, every configuration has at most three membranes. Hence the active P system with limit on the total number of membranes exists (e.g. with $L = 3$), so the universality holds. □

4.1 Existence of infinite computation

We will propose an algorithm for deciding existence of infinite computation. Basic idea is to consider the minimal coverability graph ([5]), where nodes are configurations and an edge leads from the configuration C_1 to the configuration C_2, whenever there is a rule applicable in C_1, which results in C_2. The construction in [5] is performed on Petri nets, where the configuration consists just of a vector of natural numbers. The situation is the same for single-membrane sequential P systems. We need to modify the construction for active P systems.

A configuration $C_2 = (T_2, l_2, c_2)$ **covers** configuration $C_1 = (T_1, l_1, c_1)$ iff $\exists$ isomorphism $f : T_1 \to T_2$ preserving membrane labels and contents: $\forall d \in T_1$ the following properties hold: $l_1(d) = l_2(f(d)) \wedge c_1(d) \subseteq c_2(f(d))$.

We will denote this with $C_1 \leq C_2$.

Lemma 1. *For sequential active P system with limit on the total number of membranes, if* $C_2 = (T_2, l_2, c_2)$ **covers** *configuration* $C_1 = (T_1, l_1, c_1)$*, then there is an isomorphism* $f : T_1 \to T_2$ *such that if a rule* r *is applicable in membrane* $d \in T_1$*, then* r *is applicable in* $f(d)$*.*

Proof. Suppose r is applicable in d. Then the left side u of the rule r is contained within the contents of the membrane $u \subseteq c_1(d)$. Because $C_1 \leq C_2$, then there is an isomorphism $f : T_1 \to T_2$ such that $c_1(d) \subseteq c_2(f(d))$ and then $u \subseteq c_2(f(d))$.

There are three possible forms of the rule r.

- If $r = u \to w$, then because r is applicable in d, $\forall(a, \downarrow_k) \in w \exists d_2 \in V(T_1) : l_1(d_2) = k \wedge parent_{T_1}(d_2) = d$. Because $C_1 \leq C_2$, then for $f(d_2) \in V(T_2)$ the following holds: $l_2(f(d_2)) = l_1(d_2) = k$ and $parent_{T_2}(f(d_2) = f(d)$. Hence r is applicable in $f(d)$.
- If $r = u \to w\delta$, then $d \neq r_{T_1}$. Since f is an isomorphism, then also $f(d) \neq r_{T_2}$. Other properties follows from the previous case.
- If $r = u \to [_k v]_k$, then $|V(T_1)| < L$. Isomorphism preserves number of nodes, hence $|V(T_2)| = |V(T_1)| < L$ and r is applicable in $f(d)$. □

Now, we will define the encoding of a configuration $C = (T, l, c)$ into a tuple of integers.

A membrane $d \in T$ will be encoded as $(n+m)$-tuple $enc(d) \in \mathbb{N}^{(n+m)}$, where first n numbers will be actual counts of objects and next m numbers will encode the membrane label:

$$enc(d)_i = \begin{cases} c(d)(a_i) & \text{if } i \leq n \\ 0 & \text{if } n < i \leq m \wedge i - n \neq l(d) \\ 1 & \text{if } n < i \leq m \wedge i - n = l(d) \end{cases}$$

.

The entire tree will be encoded into concatenated sequences of encoded nodes in the preorder traversal order. This sequence is then padded with zeroes to have length $(n + m)L$ as that is the maximal length of encoded tree.

Example 1. Suppose skin membrane with label 1 and contents $a_1^2 a_2$ with a child membrane with label 2 and contents a_2^2. Then the encoding will be 21100201.

Since there are only finitely many non-isomorphic trees with at most L nodes ([6]), there is a constant z such that we can uniquely assign the tree an order number $o(T) \leq z$.

The entire configuration will be encoded in tuple which consists of z parts. All but the part with index $o(T)$ will contain just zeros. The part with index $o(T)$ will contain the encoding of the tree.

Lemma 2. *For configurations $C_1 = (T_1, l_1, c_1)$ and $C_2 = (T_2, l_2, c_2)$, $enc(C_1) \leq enc(C_2) \Rightarrow C_1 \leq C_2$.*

Proof. Both $enc(C_1)$ and $enc(C_2)$ contains z parts and exactly one part contains non-zero values. Non-zero part of $enc(C_1)$ must be non-zero also in $enc(C_2)$, because $enc(C_1) \leq enc(C_2)$. Then $o(T_1) = o(T_2)$, so the trees are isomorphic. Suppose there is an isomorphism $f : T_1 \to T_2$. For every membrane $d \in T_1$, $l_1(d) = l_2(f(d))$ and $c_1(d) \subseteq c_2(f(d))$. Hence, $C_1 \leq C_2$. □

Lemma 3. *For sequential active P system with limit on the total number of membranes L for every infinite sequence of configurations $\{C_i\}_{i=0}^{\infty} \exists i < j : C_i \leq C_j$.*

Proof. Suppose an infinite sequence $\{enc(C_i)\}_{i=0}^{\infty}$. We use a variation of Dickson's lemma ([7]): Every infinite sequence of tuples from $\mathbb{N}^k$ contains an increasing pair. Applied to our sequence, there are two positions $i < j : enc(C_i) \leq enc(C_j)$. From lemma 2, $C_i \leq C_j$. □

Theorem 2. *Existence of infinite computation for active P systems with limit on the total number of membranes is decidable.*

Proof. The algorithm for deciding the problem will traverse the reachability graph. When it encounters a configuration that covers another configuration, from lemma 1 follows that the same rules can be applied repeatedly, so the algorithm will halt with the answer YES. Otherwise, the algorithm will answer NO. Algorithm will always halt, because if there was an infinite computation, from lemma 3 there would be two increasing configurations which is already covered in the YES case. □

4.2 Existence of halting computation

In this subsection we will focus on the opposite problem: whether there is a computation that is halting. Recall that halting computation has no applicable rule in the last configuration. First, we will reduce this problem to the reachability. It is a problem of determining, for a given configuration C, whether there exists a computation from C_0 to C. The reachability of active P systems can be then reduced to the reachabililty of register machines, which is undecidable.

For a given P system Π and a target configuration C we will construct a P system Π' such that there is a halting computation of $\Pi' \Leftrightarrow$ C is reachable for Π. Suppose $\Pi = (\Sigma, C_0, R_1, \dots R_m)$ and $C = (T, l, c)$. Then we will construct $\Pi' = (\Sigma', C_0', R_1', \dots R_m')$, where:

- $\Sigma' = \Sigma \cup \{\xi_d | d \in V(T)\}$,
- $C_0' = (T, l, c')$, where $\forall d \in V(T) \backslash r_T : c'(d) = c(d)$ and $c'(r_T) = c(r_T) \cup \{\xi_{r_T}\}$,
- $\forall i \in \{1 \dots m\} : R_i' = R_i \cup \{\xi_d c(d) \rightarrow \xi_{d'} \downarrow_{l(d')} | d, d' \in V(T), l(d) = i, parent(d') = d\}$.

The ξ_d objects are called verifiers, they are intended to verify if the contents of the membrane corresponds to the contents in the target configuration C. After this verification it descends down into child membranes for the verification of other parts of the membrane structure. Initially, there is an object ξ_{r_T} in the skin membrane. Verification is performed in the rule $\xi_d c(d) \rightarrow \xi_{d'} \downarrow_{l(d')}$, where on the right side there is $\xi_{d'}$ object for every child membrane d' in the target configuration C.

The construction is not complete. The system should not be able to halt unless the verification takes place. That is why we introduce a new object ω to each membrane with a rule $\omega \rightarrow \omega$ and the verifier will erase them with rule $\xi_d \omega c(d) \rightarrow \xi_{d'} \downarrow_{l(d')}$. We also need to ensure that newly created membranes contain the ω object, so we replace every rule for membrane creation $u \rightarrow [_k v]_k$ with $u \rightarrow [_k v \omega]_k$.

There is still a problem that we actually check, whether a target configuration is contained within the current configuration. If there are additional objects which cannot be erased, so C cannot be reached, we need to ensure Π' will not halt. We will add a rule $a \to a$ to each membrane for each object $a \in \Sigma$, so Π' can halt only if all objects are erased.

The last issue to solve is that it is still possible that in the middle of the verifying, some of already verified membranes got dissolved and all yet unverified membranes will be successfully verified causing Π' to halt, although without that dissolution it would be unable to reach C. We need to ensure all dissolution happen before the verification takes place. We will add a new object σ, which stands as a footprint object. It will be created as a result of the verification rule $\xi_d \omega c(d) \to \sigma \xi_{d'} \downarrow_{l(d')}$. If a membrane is dissolved after it was verified, then two σs will meet in the same membrane, because the parent membrane also contains σ as it had been verified before. We will add a rule $\sigma\sigma \to \sigma\sigma$ to prevent Π' from halting.

The final construct is:

- $\Sigma' = \Sigma \cup \{\omega, \sigma\} \cup \{\xi_d | d \in V(T)\}$,
- $C'_0 = (T, l, c')$, where $\forall d \in V(T) \setminus r_T : c'(d) = c(d) \cup \{\omega\}$ and $c'(r_T) = c(r_T) \cup \{\omega, \xi_{r_T}\}$,
- $\forall i \in \{1 \dots m\} : R'_i = \{r | r \in R_i, r = u \to w \vee r = u \to w\delta\} \cup \{u \to [_k v\omega]_k | u \to [_k v]_k \in R_i\} \cup \{a \to a | a \in \Sigma\} \cup \{\sigma\sigma \to \sigma\sigma, \omega \to \omega\} \cup \{\xi_d \omega c(d) \to \sigma \xi_{d'} \downarrow_{l(d')} | d, d' \in V(T), l(d) = i, parent(d') = d\}$.

We need to prove two implications in order to formally prove correctness of this construction.

Lemma 4. *If C is reachable for Π then there is a halting computation of Π'.*

Proof. Consider computation Π with C as the last configuration. For Π' there is a corresponding computation as the rules of Π are included in Π' with an exception of the rule $u \to [_k v]_k$, which has a corresponding rule with the same left side: $u \to [_k v\omega]_k$. The corresponding computation of Π' will result in a configuration, where in every membrane d, the contents will be $\omega c(d)$, and the skin membrane will contain an additional ξ_{r_T}. Then the cascade of applications of rule $\xi_d \omega c(d) \to \sigma \xi_{d'} \downarrow_{l(d')}$ can be applied, starting in the skin membrane and cascading down to children until all the membranes applied that rule. The objects $\omega c(d)$ will be replaced by σ and the computation will halt. □

The other direction of the implication is more complicated, so following lemmas will state some properties about computations of Π and Π'.

Lemma 5. *For all halting computations of Π' there is no rule of form $u \to w\delta$ applied in the membrane d' after the application of rule $\xi_d \omega c(d) \to \sigma \xi_{d'} \downarrow_{l(d')}$ in membrane $d = parent(d')$.*

Proof. After the application of the rule $\xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$, the object σ remains in the membrane. There is no possible interaction of σ with other objects, only with another σ. If a child membrane d' is dissolved with a rule $u \rightarrow w\delta$, then two objects σ will meet in the membrane d and the computation will not halt, which contradicts the fact that the computation is halting. □

Lemma 6. *For all halting computations of Π' there is no rule of form $u \rightarrow [_k v\omega]_k$ applied in the membrane d' after the application of rule $\xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$.*

Proof. After the application of rule $r = \xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$ in membrane d, there will not be any of ξ objects present in the membrane, because they are only sent into child membranes, which cannot be dissolved because of lemma 5. So there will be no more of rule r applied in membrane d. The newly created membrane will never receive any of ξ objects, so the object ω will never be erased and the computation will not halt, which contradicts the fact that the computation is halting. □

Lemma 7. *If a halting computation of Π' exists then there is a halting computation, where for every membrane $d \in V(T)$ in the target configuration $C = (T, l, c)$ the last rule used is $r = \xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$.*

Proof. Consider membrane d: let C_1 be the configuration before the application of r, C_2 the configuration after the application of r and C_3 the halting configuration.

First consider elementary membranes, lets have a membrane d. As it has no child membranes, C_2 is contained within C_1 (with the exception of the σ object). Therefore the sequence of steps from C_2 to C_3 can be applied starting from C_1 instead of C_2. It would result in a configuration C_4, where contents of d is exactly $\xi_d \omega c(d)$, assuming d has not been dissolved, what is ensured by lemma 5. So the rule $r = \xi_d \omega c(d) \rightarrow \sigma$ can be applied, resulting in configuration, where d contains exactly one σ and nothing else, so no more rule will be applied in d.

Consider a membrane d, which is not elementary, so it has child membranes. In C_1 the verifier is in d and no child membrane contains the verifier. In C_2 every child membrane contains the verifier. We cannot use the same argument, because C_2 is not contained within C_1.

We will proceed by induction, starting from the elementary membranes to parent membranes. Assume the lemma holds for child membranes. Then we have a computation, which contains a configuration C_5, after which only verification rules are applied in child membranes. So the sequence of steps from C_2 to C_5 does not contain the verification rule in child membranes. Hence this sequence can be used starting from C_1 instead of C_2, resulting in a configuration C_6, where application of verification rule in d results in C_5. So again, after C_6 only verification rules are applied. □

Lemma 8. *If there is a halting computation of Π' then C is reachable for Π.*

Proof. According to lemma 7 there is also a halting computation where in every membrane d the last used rule is $r = \xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$. So the corresponding

computation in Π will result in the configuration C_5, where every membrane d contains exactly $c(d)$. If C_5 contains a membrane not present in C, it will contain the object ω, which will not be reached by any of objects ξ so the computation will not halt. If C contains a membrane d' not present in C_5, then the membrane $d = parent(d')$ will never get rid of ω because the rule $r = \xi_d \omega c(d) \rightarrow \sigma \xi_{d'} \downarrow_{l(d')}$ cannot be applied due to lack of the child membrane d'. Hence $C = C_5$ and there is a computation of Π that will result in C. □

Theorem 3. *Existence of halting computation for active P systems with limit on the total number of membranes is undecidable.*

Proof. For a given P system Π and a target configuration C we have constructed a P system Π' such that there is a halting computation of Π' iff the Π can reach configuration C. The two directions of the equivalence have been proven in lemmas 4 and 8. Using this construction, we can reduce the existence of halting computation to reachability of register machines [2], which is known to be undecidable. □

5 Conclusion

We have studied the termination problems for active sequential P systems. Unlike deterministic systems, the termination problems cannot be simply reduced to the halting problem. We have shown that active P systems with limit on the number of membranes have decidable existence of infinite computation and undecidable existence of halting computation. It is currently unknown whether the same results apply also for a variant without the limit on the number of membranes, so it could be a subject for the future study.

Regarding the open problem stated in [2] about sequential active P systems with hard membranes (without communication between membranes), it could be interesting to find a connection between the universality and decidability of these termination problems.

References

1. Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1) (2000) 108 – 143
2. Ibarra, O., Woodworth, S., Yen, H.C., Dang, Z.: On sequential and 1-deterministic p systems. In Wang, L., ed.: Computing and Combinatorics. Volume 3595 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2005) 905–914
3. Alhazov, A.: Properties of membrane systems. In Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S., eds.: Membrane Computing. Volume 7184 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 1–13
4. Păun, G.: Introduction to membrane computing. In Ciobanu, G., Păun, G., Pérez-Jiménez, M., eds.: Applications of Membrane Computing. Natural Computing Series. Springer Berlin Heidelberg (2006) 1–42

5. Finkel, A.: The minimal coverability graph for petri nets. In Rozenberg, G., ed.: Advances in Petri Nets 1993. Volume 674 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1993) 210–243
6. Cayley, P.: On the analytical forms called trees. American Journal of Mathematics **4**(1) (1881) pp. 266–268
7. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermannian and primitive-recursive bounds with dickson's lemma. In: Proceedings of the 2011 IEEE 26th Annual Symposium on Logic in Computer Science. LICS '11, Washington, DC, USA, IEEE Computer Society (2011) 269–278