P Systems with Active Membranes and Two Polarizations

Artiom ALHAZOV
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: artiome.alhazov@estudiants.urv.es
and
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: artiom@math.md

Rudolf FREUND
Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9, A-1040 Wien, Austria
E-mail: rudi@emcc.at

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
and
Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: gpaun@us.es

Abstract. P systems with active membranes using only two electrical charges and only rules of types (a) and (c) assigned to at most two membranes are shown to be computationally complete – thus improving the previous result of this type from the point of view of the number of polarizations as well as with respect to the number of membranes. Allowing a special variant of rules of type (c) to delete symbols by sending them out, even only one membrane is needed. Moreover, we present an algorithm for deterministically deciding SAT in linear time using only two polarizations and global rules of types (a) , (c), and (e).

1 Introduction

Membrane systems are biologically motivated theoretical models of distributed and parallel computing. The most interesting questions in the area probably are completeness (solving every solvable problem) and efficiency (solving a hard problem in feasible time). In this paper we will address both problems, i.e., we shall show that when using only two polarizations in P systems

with active membranes we already obtain computational completeness in only two membranes and as well we shall give an algorithm how to use such systems to decide SAT in linear time.

The question of removing the polarizations (charges $+, -, 0$ associated with the membranes) from P systems with active membranes without diminishing their computing power or their efficiency in solving computationally hard problems in a feasible time was formulated several times and was recently considered in various contexts (with the polarizations replaced by various other features, such as label changing – see, e.g., [3], [4]). Here we present another way for improving previous results: the number of polarizations can be decreased to two, without introducing new features. It is worth mentioning that the computational completeness is obtained for systems with the same types of rules as in [15], hence without using membrane division and membrane dissolution, even decreasing the number of membranes from three (Theorem 7.2.1 in [15]) to two. It remains as an open question whether polarizations can be completely removed.

There are numerous results of solving such (mostly **NP**-complete) problems as SAT, HPP, Validity, Subset-Sum, Knapsack, Vertex Cover, Clique, QBF-SAT by P systems with active membranes with three polarizations (e.g., see [3], [4], [1], [2], [8], [9], [11], [12], [13], [14], [16], [17], [19], [20]). The ability of the system to act depending on the membrane polarizations and to change them is a powerful control feature, the use of which is not necessary if one pays the price of changing membrane labels. Another result is solving SAT in a uniform manner, without polarizations and without changing labels, but also using membrane dissolution and non-elementary membrane division. Here we show that two polarizations are enough even when restricting the types of rules to (a), (c), and (e).

2 Prerequisites

The reader is assumed to be familiar with basic elements of formal language theory. For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. The family of recursively enumerable languages is denoted by RE; NRE denotes the family of recursively enumerable sets of non-negative integers, and $PsRE$ denotes the family of recursively enumerable sets of Parikh vectors of non-negative integers. In the following we will not distinguish between a vector $(y_1, ..., y_\beta)$, its representation by a multiset or its representation by a string with Parikh vector $(y_1, ..., y_\beta)$. For more notions as well as basic results from the theory of formal languages, the reader is referred to [5] and [18].

We now also recall the definition of a graph-controlled grammar and prove a special normal form needed in the proof of the main theorem stated in this paper:

A *graph-controlled grammar* is a construct

$$G = (N, T, Lab, S, R, \{1\}, \{n\}),$$

where N denotes the set of non-terminals, T is the set of terminal symbols, $Lab = \{1, ..., n\}$ is the set of labels, S is the start symbol, R is a finite set of rules that can be represented as a function from Lab to $P \times 2^{Lab} \times 2^{Lab}$, where P denotes the set of all context-free productions over the set N of non-terminal symbols and the set of terminal symbols T. A rule in R usually is written in the form

$$(i : p(i), \sigma(i), \varphi(i)),$$

where $\sigma(i)$ is called the success field and $\varphi(i)$ is called the failure field of the rule labelled by i; the context-free production $p(i)$ is of the form $A(i) \to w(i)$, where $A(i) \in N$ and $w(i) \in (N \cup T)^*$.

Without loss of generality we not only assume that $N \cap Lab = \emptyset$ and that there is only one initial label (i.e., 1) and only one final label (i.e., n, with $\sigma(n) = \varphi(n) = \emptyset$), but we also may assume that if a computation has reached the final label n, then the obtained sentential form is terminal, i.e., it must not contain any non-terminal symbol.

As a special technical detail, without loss of generality we may assume any right-hand side $w(m)$ to contain at most one terminal symbol. Finally, again without loss of generality we may also assume that in the case of a string language, the terminal symbols are generated by G exactly in the correct sequence as they form a terminal word. All these features of a normal form for graph-controlled grammars, for example, follow from the constructions and results proved in [6], Theorem 6.

We now add one more feature to the normal form of graph-controlled grammars given above, i.e., from such a graph-controlled grammar G we now construct a graph-controlled grammar

$$G' = \left(N', T, Lab', S', R', \{0\}, \{n+1\}\right)$$

with $N' = N \cup \{S', F\}$ and $Lab' = Lab \cup \{0, n+1, n+2\}$, which has the additional feature that all failure fields and all success fields in G' are non-empty:

S' and F are new non-terminal symbols; S' is used as the new start symbol only used in the production assigned to the new label 0, where we take

$$\left(0 : S' \to SF, \{1\}, \{1\}\right);$$

F is used as a kind of trap symbol which remains for entering an infinite loop at label $n+2$ with

$$(n+2 : F \to F, \{n+2\}, \{n+2\}).$$

Moreover, if the original final label n was reached, it was clear that only terminal symbols were present in the current sentential form; hence, to erase the newly added symbol F we add the new rule

$$(n : F \to \lambda, \{n+1\}, \{n+1\}).$$

Thus, we finish with a terminal word in the new final label $n+1$, where for sake of conciseness we take

$$(n+1 : F \to F, \{n+1\}, \{n+1\}).$$

In order to obtain all failure fields and all success fields in G' to be non-empty, we construct the set of rules R'' from the set of rules R in the following way:

For every rule $(i : p(i), \sigma(i), \varphi(i))$ in R we take the rule $\left(i : p(i), \sigma'(i), \varphi(i)'\right)$, where $\sigma'(i) = \sigma(i)$ for $\sigma(i) \neq \emptyset$ and $\sigma'(i) = \{n+2\}$ for $\sigma(i) = \emptyset$ as well as $\varphi'(i) = \varphi(i)$ for $\varphi(i) \neq \emptyset$ and $\varphi'(i) = \{n+2\}$ for $\varphi(i) = \emptyset$.

In sum, we obtain

$$\begin{aligned} R' &= R'' \setminus \{(n : p(n), \emptyset, \emptyset)\} \\ &\cup \{(0 : S' \to SF, \{1\}, \{1\}), (n : F \to \lambda, \{n+1\}, \{n+1\})\} \\ &\cup \{(n+1 : F \to F, \{n+1\}, \{n+1\}), (n+2 : F \to F, \{n+2\}, \{n+2\})\}. \end{aligned}$$

We also assume the reader to be familiar with the basic elements of membrane computing, e.g., from [15] (details can be found at `http://psystems.disco.unimib.it`), in particular, with P systems with active membranes.

For the sake of completeness, we recall the definition of P systems with active membranes for the case when only rules of types (a), (b), and (c) or (c_λ) as well as possibly (e) are used;

in a more general way as in the original definition, we allow the polarizations to be arbitrary non-negative integers.

A *P system system with active membranes* (of degree $m \geq 1$) is a construct of the form

$$\Pi = (O, E, \mu, w_1, \ldots, w_m, e_1, \ldots, e_m, R),$$

where O is the alphabet of objects, $E = \{0, ..., n-1\}$ with $n \geq 1$ is the set of electrical charges (polarizations), μ is the membrane structure (with m membranes, bijectively labelled with $1, 2, \ldots, m$; by H we denote the set of labels $\{1, 2, \ldots, m\}$), $w_1, \ldots, w_m$ are strings over O indicating the multisets of objects at the beginning present in the m regions of μ, $e_1, \ldots, e_m$ are the polarizations at the beginning assigned to the membranes $1, \ldots, m$, and R is a finite set of rules of the following forms:

(a) $[a \to v]_h^i$, $a \in O$, $v \in O^*$, $h \in H$, $i, j \in E$

(evolution rules, used in parallel in the region of membrane h, provided that the polarization of the membrane is i);

(b) $a\,[\ ]_h^i \to [b]_h^j$, $a, b \in O$, $h \in H$, $i, j \in E$

(communication rule, sending an object into a membrane, possibly changing the polarization of the membrane);

(c) $[a]_h^i \to [\ ]_h^j\, b$, $a, b \in O$, $h \in H$, $i, j \in E$

(communication rule, sending an object out of a membrane, possibly changing the polarization of the membrane).

We shall also consider the following variant of rule type (c):

(c_λ) $[a]_h^i \to [\ ]_h^j\, b$, $a \in O$, $b \in O \cup \{\lambda\}$, $h \in H$, $i, j \in E$

(communication rule, sending an object out of a membrane or "killing" it by sending it through the membrane, possibly changing the polarization of the membrane).

(e) $[a]_h^i \to [b]_h^j\, [c]_h^k$, $a, b, c \in O$, $h \in H$, $i, j, k \in E$

(division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations, and the object specified in the rule is replaced in the two new membranes by possibly new objects)

Throughout this paper, we shall even use only communication rules $[a]_h^i \to [\ ]_h^j\, b$ with $a = b$ or $b = \lambda$.

The rules of types (b), (c), and (c_λ) are considered as involving the membrane, hence, we assume at most one of such a rule to be used for each membrane in a given step; the use of rules is maximally parallel, with the rules chosen in a non-deterministic manner. An output is associated with a halting computation – and only with halting computations – in the form of the objects sent into the environment during the computation; for the following definitions, we assume $\emptyset \subsetneq D \subseteq \{a, b, c, c_\lambda, e\}$:

- If we consider only the number of symbols sent out during a halting computation, then the set of all such numbers computed by a system Π is denoted by $N(\Pi)$. By $NOP_m\,(active_n, D)$ we denote the family of all sets $N(\Pi)$ computed by P systems with at most m membranes allowing for n polarizations, using rules of the types contained in D.

- If we distingish the different symbols sent out during a halting computation, the set of all such vectors of numbers computed by a system Π is denoted by $Ps(\Pi)$. By $PsOP_m\left(active_n, D\right)$ we denote the family of all sets $Ps(\Pi)$ computed by P systems with at most m membranes allowing for n polarizations, using rules of the types contained in D.

- If we consider the sequence of symbols sent out during a halting computation and interpret this sequence as a string, then the set of all such strings computed by a system Π is denoted by $L(\Pi)$. By $LOP_m\left(active_n, D\right)$ we denote the family of all languages $L(\Pi)$ computed by P systems with at most m membranes allowing for n polarizations, using rules of the types contained in D.

In this paper, we will use only two polarizations, 0 and 1, and this restriction will be indicated by writing $active_2$ in the notations defined above.

3 Completeness/Universality with Two Polarizations

Stated in the notations of this paper, Theorem 1 from [10] says that $PsOP_3\left(active_3, \{a, b, c\}\right) = PsRE$. As announced above, we here not only improve this result with respect to the number of electrical charges (polarizations), but even with respect to the number of membranes, especially when allowing rules of type (c_λ) instead of rules (c) :

Theorem 3.1 $PsOP_1\left(active_2, \{a, c_\lambda\}\right) = PsRE$.

Proof. We only prove that any recursively enumerable set of vectors of non-negative integers can be generated by a P system with active membranes using only one membrane, two polarizations, and rules of the types (a) and (c_λ).

We start with a graph-controlled grammar

$$G' = \left(N', T, Lab', S', R', \{0\}, \{n+1\}\right)$$

which is in the normal form constructed above and represents the given recursively enumerable set L of Parikh vectors.

We now construct a P system with active membranes of degree one

$$\Pi = \left(O, \{0,1\}, [_1\ ]_1, (S,0)\,(F,0)\,(1,0), 0, R_\Pi\right)$$

using only two polarizations 0 and 1 and rules of the form (a) and (c_λ) such that $PsP\left(\Pi\right) = L$.

The simulation of derivations in the graph-controlled grammar G' by derivations in the P system Π uses a colouring technique opening a "window" of length three for the application of the current rule $\left(i : p\left(i\right), \sigma\left(i\right), \varphi\left(i\right)\right)$ to be applied. Basically, the labels $k \in Lab$ occur in the variants (k, l) and the non-terminal symbols $B \in N'$ occur in the variants $\left(B, l\right)$, $0 \leq l \leq 3n$.

In the following, the label m runs from 1 to n.

- As long as membrane 1 (the skin membrane) has polarization 0, the index l of the non-terminal symbols $B \in N'$ in (B, l) may be incremented:

  $\left[(B, l) \rightarrow (B, l+1)\right]_1^0$, $B \in N'$, $0 \leq l \leq 3n$.

- For each m, the index l of $m \in Lab$ in (m, l) is incremented until the index $3m-3$ is reached:

  $\left[(m, l) \rightarrow (m, l+1)\right]_1^0$, $0 \leq l < 3m-3$.

- Then we check whether $p(m)$ can be applied to the current contents of membrane 1; by polarizing the membrane we first prohibit the incrementation of the index l in the variables of the form (B, l) :

  $[(m, 3m-3) \to (m, 3m-2)\, E]_1^0$

  In the next step, all objects $(B, 3m-2)$, $B \in N'$, in membrane 1 evolve to $(B, 3m-1)$, whereas $(m, 3m-2)$ evolves to $(m, 3m-1)$ by applying the following rule:

  $[(m, 3m-2) \to (m, 3m-1)]_1^0$

  At the same time, E passes the skin membrane thereby changing its polarization from 0 to 1 :

  $[E]_1^0 \to [\ ]_1^1 \lambda$

- With the polarization of the membrane being 1, the symbols now remain unchanged, only one suitable object – if possible – has to pass the membrane resetting the polarization to 0:

  $[(A(m), 3m-1)]_1^1 \to [\ ]_1^0 \lambda$

  At the same time, the object $(m, 3m-1)$ evolves according to the following rule:

  $[(m, 3m-1) \to m']_1^1$

- In the next step m' evolves according to the polarization of the skin membrane (the polarization has stored the one-bit information whether $A(m)$ was present or not):

  If the polarization is still 1, then m' evolves in two further steps to m''', where the symbol E generated in the first step then resets the polarization to 0 in the second step by passing the skin membrane:

  $[m' \to m''E]_1^1$,

  $[m'' \to m''']_1^1$,

  $[E]_1^1 \to [\ ]_1^0 \lambda$

- As the polarization is 0 again, the symbols $(B, 3m-1)$, $B \in N'$, may evolve to $(B, 3m)$, whereas m' and m''', respectively, evolve to different symbols with index $3m$:

  $[m' \to (\bar{m}, 3m)]_1^0$

  $[m''' \to (\hat{m}, 3m)]_1^0$

- The symbols $(\bar{m}, 3m)$ and $(\hat{m}, 3m)$ until the end of a simulation cycle evolve in the same way as the basic objects (m, l) by incrementing the second parameter:

  $[(\bar{m}, l) \to (\bar{m}, l+1)]_1^0$, $3m \le l < 3n$;

  $[(\hat{m}, l) \to (\hat{m}, l+1)]_1^0$, $3m \le l < 3n$.

- At the end of a complete cycle, we finally extract the information stored in the symbols $\bar{m}$ and $\hat{m}$, respectively, and start a new cycle:

  $[(B, 3n) \to (B, 3n+1)]_1^0$ and

  $[(B, 3n+1) \to (B, 0)]_1^0$ for all $B \in N'$;

  $[(\hat{m}, 3n) \to (\hat{m}, 3n+1)]_1^0$ or

  $[(\bar{m}, 3n) \to (\bar{m}, 3n+1)\, h(w(m))]_1^0$, respectively, where

$h : N' \cup T \to (N', 3n+1) \cup T$ is the morphism with

$h(B) = (B, 3n+1)$ for all $B \in N'$ and

$h(a) = a$ for all $a \in T$;

observe that due to our assumptions about G', $h(w(m))$ cannot contain more than one terminal symbol a, which may leave the skin membrane by using the following rule:

$[a]_1^0 \to [\ ]_1^0 a$

The next cycle of simulating a derivation in G' by Π starts after the application of one of the following rules:

$[(\bar{m}, 3n+1) \to (k, 0)]_1^0$ for every $k \in \sigma(m) \setminus \{n+1, n+2\}$;

$[(\hat{m}, 3n+1) \to (k, 0)]_1^0$ for every $k \in \varphi(m) \setminus \{n+1, n+2\}$.

In case that the label of the "trap" $n+2$ is reached then we can immediately enter an infinite loop due to the fact that the additional symbol F then still will be present in its indexed variants (F, l), $0 \leq l \leq 3n+1$:

$[(\bar{m}, 3n+1) \to \lambda]_1^0$ for every m with $n+2 \in \sigma(m)$;

$[(\hat{m}, 3n+1) \to \lambda]_1^0$ for every m with $n+2 \in \varphi(m)$.

- The simulation of a derivation in G' by Π may successfully end if we can apply

  $[(\bar{m}, 3n+1) \to \lambda]_1^0$ for $n+1 \in \sigma(m)$ or

  $[(\hat{m}, 3n+1) \to \lambda]_1^0$ for $n+1 \in \varphi(m)$.

  Due to our assumptions for G', after applying such a rule in Π no non-terminal symbol can appear any more (observe that in that case the additional symbol F has disappeared, too); hence, in case of termination we finish with an empty membrane.

The construction given above completely describes the set of rules R_Π of the P system with active membranes Π.

For sake of completeness, we finally list all objects from O obtained according to the construction given above:

- E;
- a for all $a \in T$;
- (B, l) for all $B \in N$, $0 \leq l \leq 3n+1$;
- (m, l) for all $1 \leq m \leq n$, $0 \leq l < 3m$;
- $(\bar{m}, l), (\hat{m}, l)$ for all $3m \leq l \leq 3n+1$, $1 \leq m \leq n$;
- m', m'', m''' for all $1 \leq m \leq n$.

From the explanations given above, it is obvious that the P system with active membranes Π defined above exactly generates the same set of objects as the given graph-controlled grammar in the special normal form exhibited in this section, which observation completes the proof. □

We could also consider P systems with extensions, i.e., in the constructions above we could read every λ there representing the empty word as a special non-terminal symbol not being taken

into account when considering the resulting Parikh sets; in such a case, using rules of the form (c) only instead of rules of the form (c_λ) would already yield computational completeness. Yet we do not follow this direction in the following, as the related results are obvious and directly follow from the proofs given in this section. Instead, we prove that even with the original types of rules (a) and (c) we only need one additional membrane being only used for filtering out the non-terminal symbols having passed the inner membrane:

Theorem 3.2 $PsOP_2\left(active_2, \{a, c\}\right) = PsRE.$

Proof. Again we only prove that any recursively enumerable set of numbers can be generated by a P system with active membranes now using two membranes, two polarizations, and rules of the types (a) and (c).

We again start with a graph-controlled grammar

$$G' = \left(N', T, Lab', S', R', \{0\}, \{n+1\}\right)$$

in the normal form as constructed above which generates the given recursively enumerable set L of Parikh vectors.

We now construct a P system with active membranes of degree two

$$\Pi = \left(O, \{0,1\}, [_1[_2\ ]_2]_1, (S,0)\,(F,0)\,(1,0), \lambda, 0, 0, R_\Pi\right)$$

using only two polarizations 0 and 1 and rules of the form (a) and (c) which generates L.

The simulation of derivations in the graph-controlled grammar G' by derivations in the P system Π again uses the same colouring technique as described in the previous proof; the simulation of a derivation step is carried out in membrane 2 by opening a "window" of length three for the application of the current rule $\left(i : p\left(i\right), \sigma\left(i\right), \varphi\left(i\right)\right)$ to be applied. The non-terminal symbols sent out through membrane 2 remain unchanged in the region enclosed by the skin membrane. On the other hand, the terminal symbols having passed membrane 2 in the succeeding step leave the system by immediately passing through the skin membrane.

In the following, the labels $k \in Lab$ occur in the variants (k, l) and the non-terminal symbols $B \in N'$ occur in the variants $\left(B, l\right)$, $0 \leq l \leq 3n$; the label m runs from 1 to n :

- As long as the second membrane has polarization 0, the index l of the non-terminal symbols $B \in N'$ in (B, l) may be incremented:

  $\left[(B, l) \rightarrow (B, l+1)\right]_2^0$, $B \in N'$, $0 \leq l \leq 3n$.

- For each m, the index l of $m \in Lab$ in (m, l) is incremented until the index $3m-3$ is reached:

  $\left[(m, l) \rightarrow (m, l+1)\right]_1^0$, $0 \leq l < 3m-3$.

- Then we check whether $p\left(m\right)$ can be applied to the current contents of membrane 2; by polarizing membrane 2 we first prohibit the incrementation of the index l in the variables of the form $\left(B, l\right)$:

  $\left[(m, 3m-3) \rightarrow (m, 3m-2)\, E\right]_2^0$

  In the next step, all objects $\left(B, 3m-2\right)$, $B \in N'$, in membrane 2 evolve to $\left(B, 3m-1\right)$, whereas $(m, 3m-2)$ evolves to $(m, 3m-1)$ by applying the following rule:

  $\left[(m, 3m-2) \rightarrow (m, 3m-1)\right]_2^0$

  At the same time, E passes membrane 2 thereby changing its polarization from 0 to 1 :

  $[E]_2^0 \rightarrow [\ ]_2^1 E$

- With the polarization of membrane 2 being 1, the symbols now remain unchanged, only one suitable object – if possible – has to pass the membrane resetting the polarization to 0:

  $[(A(m), 3m-1)]_2^1 \rightarrow [\ ]_2^0 (A(m), 3m-1)$

  At the same time, the object $(m, 3m-1)$ evolves according to the following rule:

  $[(m, 3m-1) \rightarrow m']_2^1$

- In the next step m' evolves according to the polarization of the membrane (the polarization has stored the one-bit information whether $A(m)$ was present or not):

  If the polarization is still 1, then m' evolves in two further steps to m''', where the symbol E generated in the first step then resets the polarization to 0 in the second step by passing the membrane:

  $[m' \rightarrow m''E]_2^1$,

  $[m'' \rightarrow m''']_2^1$,

  $[E]_2^1 \rightarrow [\ ]_2^0 E$

- As the polarization is 0 again, the symbols $(B, 3m-1)$, $B \in N'$, may evolve to $(B, 3m)$, whereas m' and m''', respectively, evolve to different symbols with index $3m$:

  $[m' \rightarrow (\bar{m}, 3m)]_2^0$

  $[m''' \rightarrow (\hat{m}, 3m)]_2^0$

- The symbols $(\bar{m}, 3m)$ and $(\hat{m}, 3m)$ until the end of a simulation cycle evolve in the same way as the basic objects (m, l) by incrementing the second parameter:

  $[(\bar{m}, l) \rightarrow (\bar{m}, l+1)]_2^0$, $3m \leq l < 3n$;

  $[(\hat{m}, l) \rightarrow (\hat{m}, l+1)]_2^0$, $3m \leq l < 3n$.

- At the end of a complete cycle, we finally extract the information stored in the symbols $\bar{m}$ and $\hat{m}$, respectively, and start a new cycle:

  $[(B, 3n) \rightarrow (B, 3n+1)]_2^0$ and

  $[(B, 3n+1) \rightarrow (B, 0)]_2^0$ for all $B \in N'$;

  $[(\hat{m}, 3n) \rightarrow (\hat{m}, 3n+1)]_2^0$ or

  $[(\bar{m}, 3n) \rightarrow (\bar{m}, 3n+1)\, h(w(m))]_2^0$, respectively, where

  $h : N' \cup T \rightarrow (N', 3n+1) \cup T$ is the morphism with

  $h(B) = (B, 3n+1)$ for all $B \in N'$ and

  $h(a) = a$ for all $a \in T$;

  observe that due to our assumptions about G', $h(w(m))$ cannot contain more than one terminal symbol a, which may pass membrane 2 by using the rule

  $[a]_2^0 \rightarrow [\ ]_2^0 a$

  and, in the immediately following step, then leaves the skin membrane by using the following rule:

  $[a]_1^0 \rightarrow [\ ]_1^0 a$

The rules of the form $[a]_1^0 \to [\ ]_1^0 a$ are the only rules affecting the skin membrane (without changing its polarity); moreover, there are no evolution rules in region 1, i.e., the other (non-terminal) symbols coming through membrane 2 are never changed in region 1, they remain there as a kind of "garbage".

The next cycle of simulating a derivation in G' by Π starts after the application of one of the following rules:

$[(\bar{m}, 3n+1) \to (k, 0)]_2^0$ for every $k \in \sigma(m) \setminus \{n+1, n+2\}$;

$[(\hat{m}, 3n+1) \to (k, 0)]_2^0$ for every $k \in \varphi(m) \setminus \{n+1, n+2\}$.

In case that the label of the "trap" $n+2$ is reached then we can immediately enter an infinite loop due to the fact that the additional symbol F then still will be present in its indexed variants (F, l), $0 \leq l \leq 3n+1$:

$[(\bar{m}, 3n+1) \to \lambda]_2^0$ for every m with $n+2 \in \sigma(m)$;

$[(\hat{m}, 3n+1) \to \lambda]_2^0$ for every m with $n+2 \in \varphi(m)$.

- The simulation of a derivation in G' by Π may successfully end if we can apply

  $[(\bar{m}, 3n+1) \to \lambda]_2^0$ for $n+1 \in \sigma(m)$ or

  $[(\hat{m}, 3n+1) \to \lambda]_2^0$ for $n+1 \in \varphi(m)$.

  Due to our assumptions for G', after applying such a rule in Π no non-terminal symbol can appear any more (observe that in that case the additional symbol F has disappeared, too); hence, in case of termination we finish with region 2 being empty, whereas region 1 still contains all the "garbage" (of non-terminal symbols passed through membrane 2).

- Obviously, we could remove this "garbage" by using the following evolution rules in region 1 :

  $[E \to \lambda]_1^0$ and $[(A(m), 3m-1) \to \lambda]_1^0$

The construction given above completely describes the set of rules R_Π of the P system with active membranes Π.

For sake of completeness, we finally define the set of objects O obtained according to the construction given above (in fact, it is identical with the set O constructed in the proof of the preceding theorem):

$$\begin{aligned} O &= \{E\} \cup T \cup \{(B, l) \mid B \in N, 0 \leq l \leq 3n+1\} \\ &\cup \{(B, l) \mid B \in N, 0 \leq l \leq 3n+1\} \\ &\cup \{(m, l) \mid 1 \leq m \leq n, , 0 \leq l < 3m\} \\ &\cup \{(\bar{m}, l), (\hat{m}, l) \mid 1 \leq m \leq n, 0 \leq l < 3m\} \\ &\cup \{m'm'', m''' \mid 1 \leq m \leq n\} \end{aligned}$$

From the explanations given above, it is obvious that the P system with active membranes Π defined above exactly generates the same set of objects as the given graph-controlled grammar in the normal form constructed in this section, which observation completes the proof. □

The following two corollaries are immediate consequences of the two preceding theorems, i.e., obviously also when considering only the number of symbols sent out through the skin membrane without distinguishing between different symbols we obtain the corresponding results:

Corollary 3.1 $NOP_1(active_2, (a, c_\lambda)) = NRE$.

Proof. The result directly follows from Theorem 3.1. □

Even when taking the original definitions from [15], we can considerably improve the result stated in Theorem 7.2.1 there, which in the notations defined in this paper says $NOP_3\,(active_3, \{a, b, c\}) = NRE$, i.e., we can improve the result with respect to the number of polarizations as well as to the number of membranes, too.

Corollary 3.2 $NOP_2\,(active_2, \{a, c\}) = NRE$.

Proof. The result directly follows from Theorem 3.2. □

So far, the terminal symbols – as the result of a successful computation – have been sent out the skin membrane without regarding the order of their appearance; regarding the sequence of symbols sent out during a successful (i.e., halting) computation as a string we obtain languages of strings:

Corollary 3.3 $LOP_1\,(active_2, \{a, c_\lambda\}) = RE$.

Proof. We only prove that any recursively enumerable language can be generated by a P system with active membranes using only one membrane, two polarizations, and rules of the form (a) and (c_λ). The proof follows from Theorem 3.1 by going into the details of the proof of Theorem 6 in [6]: There a graph-controlled graph grammar is constructed in such a way that the symbols of a terminal string are generated symbol by symbol in the same order as they form this string. Hence, in the simulating P system the terminal symbols pass the skin membrane in just the same sequence as they are generated by the graph-controlled grammar. This observation already completes the proof. □

Corollary 3.4 $LOP_2\,(active_2, \{a, c\}) = RE$.

Proof. The proof follows from Theorem 3.2 in the same way as the proof of Corollary 3.3 followed from Theorem 3.1: In the same sequence as they are generated by the graph-controlled grammar, the terminal symbols pass the second membrane of the simulating P system and one step later are sent out through the skin membrane. □

In addition to the generative P systems with active membranes sending out the results of their (halting) computations through the skin membrane, we could also define the following variants:

- P systems with active membranes with internal output keep the output in the innermost elementary membrane; in this case, the results proved in Theorems 3.1 and 3.2 as well as Corollaries 3.1 and 3.2 still remain valid.
- Accepting P systems with active membranes accept multisets of objects given in a specified input membrane by halting computations; again similar results as proved in Theorems 3.1 and 3.2 as well as Corollaries 3.1 and 3.2 hold true for accepting P systems with active membranes.
- Computing P systems with active membranes start with multisets of objects given in a specified input membrane and then (in halting computations) compute functions, the results appearing as external or internal output in halting computations. Again similar results as those presented so far in this section hold true.

Without going into the very details of the proofs we just mention that for accepting P systems with active membranes as well as for computing P systems with active membranes we may take the universal model of *deterministic* graph controlled grammars to be simulated by the P systems with active membranes. In deterministic graph controlled grammars each success field and each failure field contains exactly one element. The input values are given as multisets over an input alphabet which is considered to be a subset of the non-terminal alphabet of the deterministic graph controlled grammar. According to the constructions given in the proofs of Theorems 3.1 and 3.2, the simulation of the deterministic graph-controlled grammar by the corresponding (accepting, computing) P systems with active membranes is deterministic, too, which is a very important feature in the area of P systems (e.g., see [7]).

At the end of this section, we list the problems left open despite the possibly optimal results (with respect to computational completeness) proved above:

- What happens if we allow only rules of the types (a) and (c) in one membrane, but possibly an unbounded number of polarizations, i.e., how can we characterize $PsOP_1\left(active_n, \{a, c\}\right)$ for $n \geq 1$?
- How can we characterize $PsOP_m\left(active_1, \{a, c_\lambda\}\right)$, $m \geq 1$?
- Can we at least characterize $PsOP_1\left(active_1, \{a, c\}\right)$ or $PsOP_1\left(active_1, \{a, c_\lambda\}\right)$?

4 Deterministically Solving SAT in Linear Time

In this section we now show that only two polarizations are needed for P systems with active membranes and global rules of types (a), (c), and (e). As we have the same set of rules for all membranes, in this section we shall omit the membrane labels. Moreover, throughout this section we consider recognizing P systems with active membranes, i.e., the initial input in addition is put into membrane 1.

Theorem 4.1 *SAT can be deterministically solved in linear time by P systems with active membranes with two polarizations and global rules of types (a), (c), and (e), constructed in a uniform manner.*

Proof. Let us consider a propositional formula in conjunctive normal form:

$$\begin{aligned} \beta &= C_1 \wedge \cdots \wedge C_m, \\ C_i &= y_{i,1} \vee \cdots \vee y_{i,l_i},\ 1 \leq i \leq m, \text{ where} \\ y_{i,k} &\in \{x_j, \neg x_j \mid 1 \leq j \leq n\},\ 1 \leq i \leq m, 1 \leq k \leq l_i. \end{aligned}$$

The instance β (to which the size (m, n) is associated) is encoded as a multiset over

$$V(n, m) = \left\{x_{i,j,j}, x'_{i,j,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\right\}.$$

The object $x_{i,j,j}$ represents the variable x_j appearing in the clause C_i without negation, and object $x'_{i,j,j}$ represents the variable x_j appearing in the clause C_i with negation. Thus, the input multiset is

$$\begin{aligned} w &= \{x_{i,j,j} \mid x_j \in \{y_{i,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq m, 1 \leq j \leq n\} \\ &\cup \left\{x'_{i,j,j} \mid \neg x_j \in \{y_{i,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq m, 1 \leq j \leq n\right\}, \end{aligned}$$

which has to be put into membrane 2 in addition to the initial symbol d_0 in the recognizing P system $\Pi(n,m)$we are going to construct for any given $(n,m) \in \mathbb{N}^2$ (where $\mathbb{N}$ denotes the set of positive integers):

$$\begin{aligned}
\Pi(n,m) &= (O(n,m), \{0,1\}, [_1[_2\ ]_2]_1, t_0, d_0, 0, 0, R), \\
O(n,m) &= \left\{x_{i,j,k}, x'_{i,j,k} \mid 1 \le i \le m, 0 \le k \le j \le n\right\} \cup \{z, o, \texttt{yes}, \texttt{no}\} \\
&\cup \{c_{i,j} \mid 0 \le i \le m, 0 \le k \le n\} \cup \{c_i \mid 0 \le i \le m\} \\
&\cup \{d_i, e_i \mid 0 \le i \le n+1\} \cup \{t_i \mid 0 \le i \le n+2m+4\};
\end{aligned}$$

R contains the following rules (we also give explanations about the use of these rules; again observe that we omit the labels of the membranes, because the rules are global):

Global control in skin membrane

- $[\ t_i \to t_{i+1}\ ]^0$, $0 \le i \le n+2m+2$;

the control variables t_i only occur in exactly one copy in the skin membrane. As we shall see at the end of the description of the whole algorithm, after $n+2m+3$ derivation steps in the corresponding P system $\Pi(n,m)$ the answer *yes* appears outside the skin membrane if the given satisfiability problem has a solution, whereas in the case that no solution exists, one step later the answer *no* appears in the environment.

The main task of the algorithm is accomplished in the generation phase of the algorithm where for each possible truth assignment to the n variables one elementary membrane is generated which after $n+1$ steps will contain all the informations needed to decide whether it represents a solution to the given problem or not:

Generation phase

- $[\ d_i\ ]^e \to [\ d_{i+1}\ ]^0[\ d_{i+1}\ ]^1$, $e \in \{0,1\}$, $0 \le i < n-1$;
- $[\ x_{i,j,k} \to x_{i,j,k-1}\ ]^e$,
  $[\ x'_{i,j,k} \to x'_{i,j,k-1}\ ]^e$, $e \in \{0,1\}$, $1 \le i \le m$, $1 \le k \le j \le n$;
- $[\ x_{i,j,0} \to \lambda\ ]^0$,
  $[\ x_{i,j,0} \to c_{i,j}\ ]^1$,
  $[\ x'_{i,j,0} \to c_{i,j}\ ]^0$,
  $[\ x'_{i,j,0} \to \lambda\ ]^1$, $1 \le i \le m$, $1 \le j \le n$;
- $[\ c_{i,j} \to c_{i,j+1}\ ]^1$, $e \in \{0,1\}$, $1 \le i \le m$, $1 \le j < n$;
- $[\ d_n \to d_{n+1}z\ ]^1$,
  $[\ d_n \to d_{n+1}\ ]^0$.

During each of the first n steps, every elementary membrane is duplicated, in order to examine all possible truth assignments to the variables $x_1, \cdots, x_n$.

Now let us consider step i of the generation phase: One of the membranes resulting from the application of the rule

$$[\ d_i\ ]^e \to [\ d_{i+1}\ ]^0[\ d_{i+1}\ ]^1$$

is with polarization 0, corresponding to assigning the truth value `false` to x_i (and in this case the clauses where $\neg x_i$ appears are satisfied), and the other membrane is with polarization 1, corresponding to assigning the truth value `true` to x_i (and in this case those clauses where x_i appears without negation are satisfied). Rather important for the correct answer to the decision problem is the application of the rules

$$[\ x_{i,j,0} \to \lambda\ ]^0, [\ x_{i,j,0} \to c_{i,j}\ ]^1, [\ x'_{i,j,0} \to c_{i,j}\ ]^0, [\ x'_{i,j,0} \to \lambda\ ]^1,$$

which in the corresponding step of the derivation act according to the truth value assigned to x_i in the underlying elementary membrane, i.e., only those variables "survive" which correspond to the correct truth assignment at the moment the last index has reached the ground level 0.

After the end of this first phase of the algorithm, 2^n elementary membranes (each of them with label 2) have been produced, each of them containing d_{n+1} and objects $c_{i,n}$ for all clauses C_i that are satisfied. Every membrane with polarization 1 also contains an object z. This procedure described so far in total takes $n+1$ step.

Transition phase

- $[\ z\ ]^1 \to [\ \ ]^0 o$;
- $[\ d_{n+1} \to e_1\ ]^e$, $e \in \{0,1\}$;
- $[\ c_{i,n} \to c_i\ ]^e$, $e \in \{0,1\}$, $1 \leq i \leq n$.

The objects z are only needed to reset the polarization of the membranes polarized by 1 to zero again by passing through the surrounding membrane; the application of the rule $[\ z\ ]^1 \to [\ \ ]^0 o$ yields the "garbage" symbol o within the skin membrane. After this single step of the transition phase all the elementary membranes now have the polarization 0 and contain e_1 as well as c_i for every satisfied clause C_i.

Checking phase

- $[\ c_1\ ]^0 \to [\ \ ]^1 o$;
- $[\ e_i \to e_{i+1} z\ ]^0$, $1 \leq i < m$;
- $[\ c_1 \to \lambda\ ]^1$;
- $[\ c_i \to c_{i-1}\ ]^1$, $2 \leq i \leq m$;
- $[\ e_m \to e_{m+1}\ ]^0$;
- $[\ e_{m+1}\ ]^1 \to [\ \ ]^1$`yes`.

All clauses are satisfied if and only if all objects $c_1, \cdots, c_m$ are present in some membrane. If in some odd step of the procedure described in the following, no symbol c_1 is present, then the polarization of the membrane will not change to 1, so finally e_{m+1} will appear, but the polarization of the elementary membrane will still be 0, so the rule $[\ e_{m+1}\ ]^1 \to [\ \ ]^1$`yes` will not be applicable. If in some odd step of the checking phase c_1 is present, then one copy exits into the outer region as o, changing the polarization of the membrane from 0 to 1. At the same time, the index of e_i is incremented and (for $i < m$) z is produced. In the succeeding even step, other copies of c_1 are erased, the indices i of all objects c_i with $1 < i \leq m$ are decremented, preparing the system for checking whether the next clause is satisfied or not. At the same time,

z exits the membrane (thereby resetting the polarization to 0 again) and appears as o in the skin membrane, and then the process continues.

In the case that all clauses are satisfied, then, at the end all objects c_i, $1 \leq i \leq m$, have been sent out into the skin membrane. While checking the last clause, no object z is produced from e_m by applying the rule

$$[\ e_m \rightarrow e_{m+1}\ ]^0,$$

hence, e_{m+1} will be present in a membrane with polarization 1 thus allowing for the application of the rule

$$[\ e_{m+1}\ ]^1 \rightarrow [\ ]^1\texttt{yes}$$

indicating that the corresponding elementary membrane represented a solution of the given satisfiability problem. In total, this phase takes $2m$ steps.

Output phase

- $[\ \texttt{yes}\ ]^0 \rightarrow [\ ]^1\texttt{yes}$;
- $[\ t_{n+2m+3}\ ]^0 \rightarrow [\ ]^0\texttt{no}$.

Every elementary membrane which after the first $n+1$ steps had represented a solution of the given satisfiability problem, after $n+1+1+2m$ steps has sent a copy of `yes` into the skin membrane, and in the next step one of these copies exits into the environment by using the rule

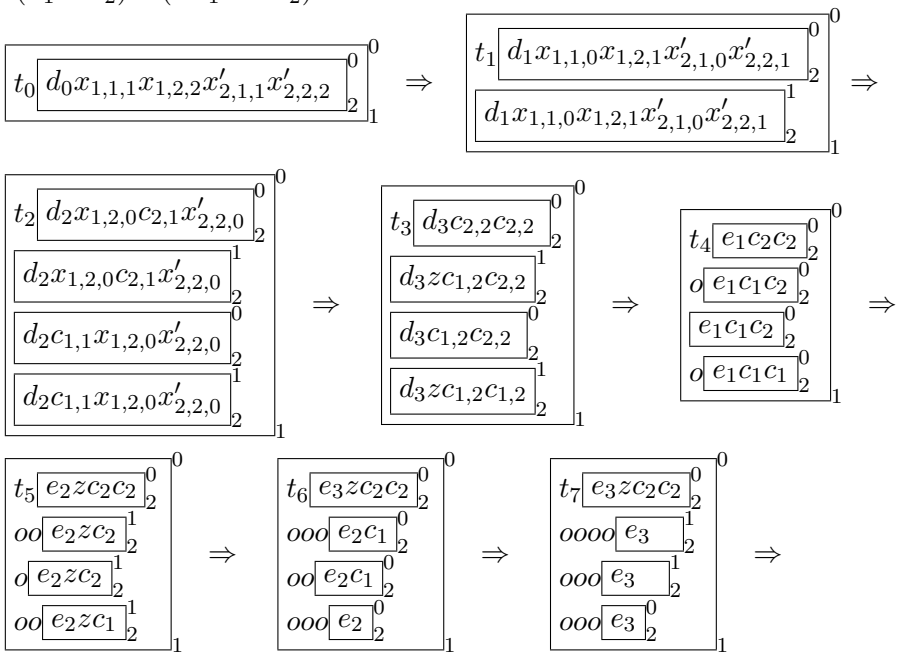
$$[\ \texttt{yes}\ ]^0 \rightarrow [\ ]^1\texttt{yes}$$

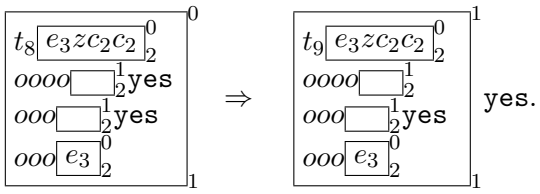
thus giving the positive result `yes` and changing the skin polarization to 1 in order to prevent further output. If, on the other hand, the given satisfiability problem has no solution, after $n+2m+3$ steps the polarization of the skin membrane will still be 0, hence, the rule

$$[\ t_{n+2m+3}\ ]^0 \rightarrow [\ ]^0\texttt{no}$$

sends out the correct answer `no`.

We illustrate the construction elaborated above by the following example:
$\gamma = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$





Due to the explanations given above one can easily verify that in any case the given algorithm will correctly decide a given satisfiability problem in n variables and m clauses in $n + 2m + 4$ steps, i.e., in linear time. □

In Theorem 4.1 we have shown that the NP-complete problem SAT can be decided by a P system with active membranes in linear time with only two polarizations and global rules of types (a), (c), and (e). There remains the open question whether the general form of these rules can be further restricted, e.g., is it possible to replace the rules of type (c) by rules of type (c_0), where the application of such a rule does not depend on the polarization of the involved membrane and as well the application of such a rule never changes the polarization of the membrane?

References

[1] A. Alhazov, C. Martín-Vide, L. Pan, Linear Time Solutions of NP-Complete Graph Problems by P Systems with Restricted Elementary Active Membranes, submitted 2003.

[2] A. Alhazov, C. Martín-Vide, L. Pan, Solving a PSPACE-complete Problem by P Systems with Restricted Active Membranes, *Fundamenta Informaticae*, **58**, 2 (2003), 67–77.

[3] A. Alhazov, L. Pan, Polarizationless P Systems with Active Membranes, *Grammars*, **7**, 1 (2004).

[4] A. Alhazov, L. Pan, Gh. Păun, Trading Polarizations for Labels in P Systems with Active Membranes, submitted 2003.

[5] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989.

[6] R. Freund, C. Martín-Vide, Gh. Păun, From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies, *Theoretical Computer Science*, **312** (2004), 143–188.

[7] R. Freund, Gh. Păun, Deterministic P systems, submitted 2004.

[8] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Solving Numerical NP-Complete Problems Using P Systems with Active membranes: the Partition Problem and Beyond, *EMCC Workshop,* Vienna (2003).

[9] S.N. Krishna, R. Rama, A Variant of P Systems with Active Membranes: Solving NP-Complete Problems, *Romanian J. of Information Science and Technology*, **2**, 4 (1999), 357–367.

[10] M. Madhu, K. Krithivasan, Improved Results about the Universality of P Systems, *Bulletin of the EATCS*, **76** (February 2002), 162–168.

[11] A. Obtułowicz, Deterministic P Systems for Solving SAT Problem, *Romanian J. of Information Science and Technology*, **4**, 1-2 (2001), 195–202.

[12] A. Obtułowicz, On P Systems with Active Membranes Solving Integer Factorizing Problem in a Polynomial Time, in *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View* (C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), Lecture Notes in Computer Science **2235**, Springer-Verlag, 2001, 267–286.

[13] A. Păun, On P Systems with Global Rules, *Proc. 7th Intern. Meeting on DNA Based Computers* (N. Jonoska, N.C. Seeman, eds.), Tampa (2001), 43–52.

[14] Gh. Păun, Computing with Membranes - A Variant: P Systems with Polarized Membranes, *Intern. J. of Foundations of Computer Science*, **11**, 1 (2000), 167–182, and CDMTCS TR **098**, Univ. of Auckland (1999).

[15] Gh. Păun, *Computing with Membranes: An Introduction*, Springer-Verlag, Berlin, 2002.

[16] M.J. Pérez-Jiménez, A. Romero Jiménez, F. Sancho Caparrini, *Teoría de la complejidad en modelos de computación celular con membranas*. Kronos Editorial, Sevilla, 2002.

[17] M.J. Pérez Jiménez, A. Romero Jiménez, F. Sancho Caparrini, Solving VALIDITY Problem by Active Membranes with Input, *Brainstorming Week on Membrane Computing*, Tarragona, 2003, Rovira i Virgili Univ., *Tech. Rep.* **26/03** (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), 279–290.

[18] A. Salomaa, G. Rozenberg (eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.

[19] P. Sosík, Solving a PSPACE-Complete Problem by P Systems with Active Membranes, *Brainstorming Week on Membrane Computing*, Tarragona, 2003, Rovira i Virgili Univ., *Tech. Rep.* **26/03** (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), 305–312.

[20] Cl. Zandron, Cl. Ferretti, G. Mauri, Solving NP-Complete Problems Using P Systems with Active Membranes, in *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 289–301.