Solving a PSPACE-Complete Problem by Recognizing P Systems with Restricted Active Membranes

Artiom Alhazov
Institute of Mathematics and Computer Science
Academy of Science of Moldova
Str. Academiei 5, Chisinau, MD 2028, Moldova
artiom@math.md; aa2.doc@estudiantes.urv.es

Carlos Martín-Vide
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
cmv@astor.urv.es

Linqiang Pan
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, People’s Republic of China
lapan@mail.hust.edu.cn; lp@fll.urv.es

Abstract. P systems are parallel molecular computing models based on processing multisets of objects in cell-like membrane structures. Recently, Petr Sosík has shown that a semi-uniform family of P systems with active membranes and 2-division is able to solve the PSPACE-complete problem QBF-SAT in linear time; he has also conjectured that the membrane dissolving rules of the (d) type may be omitted, but probably not the (f) type rules for non-elementary membrane division. In this paper, we partially confirm the conjecture proving that dissolving rules are not necessary. Moreover, the construction is now uniform. It still remains open whether or not non-elementary membrane division is needed.

Keywords: Membrane computing, P system, PSPACE-complete problem, QBF-SAT problem

The work was supported by grant 2001CAJAL-BURV4 from Rovira i Virgili University
†Also work: Research Group on Mathematical Linguistics Rovira i Virgili University, Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
††The work was supported by grant DGU-SB2001-0092 from Spanish Ministry for Education, Culture, and Sport, National Natural Science Foundation of China (Grant No. 60373089), and Huazhong University of Science and Technology Foundation.
CAddress for correspondence: Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, Hubei, People’s Republic of China
A. Alhazov et al. / Solving a PSPACE-Complete Problem by Active P Systems

1. Introduction

The P systems are a class of distributed parallel computing devices of a biochemical type, introduced in [3], which can be seen as a general computing architecture where various types of objects can be processed by various operations. For a motivation and detailed description of various P system models we refer to [3, 5].

In [4], Păun considered P systems where the number of membranes increases during a computation by using some division rules (for example, rules of types (e), (f) which we will define in section 2), which are called P systems with active membranes. This is modelling the division of cells from nature.

In [4], it was also demonstrated that P systems with active membranes have the capability to solve some NP-complete problems in linear time. In [9] Petr Sosík has produced a semi-uniform family of P systems with active membranes and 2-division which is able to solve a PSPACE-complete problem, QBF-SAT (for definition see section 3), in linear time, which implies that such a family of P systems is at least as powerful as so-called $C_2$ class computers. The $C_2$ computer class is the fundamental class of parallel computing models containing parallel RAM machine, alternating Turing machine and others, which is of crucial importance in the computational complexity theory [1, 2]. In [9], it is conjectured that the membrane dissolving rules of the (d) type may be omitted, but probably not the (f) type rules for non-elementary membrane division. In this paper, we prove that the PSPACE-complete problem QBF-SAT can be solved in linear time by recognizing P systems with restricted active membranes (i.e., the membrane dissolving rules of the (d) type can be omitted), which gives the conjecture a partial answer.

The paper is organized as follows: in section 2 we define the notion of P systems with active membranes and of recognizing P systems; section 3 gives a membrane algorithm to solve the QBF-SAT problem in linear time by recognizing P systems with restricted active membranes. Finally, in section 4 some discussion is presented.

2. P Systems with Active Membranes

We start by introducing P systems with active membranes due to [4], where more details can also be found.

A membrane structure is represented by a Venn diagram and is identified by a string of correctly matching parentheses, with a unique external pair of parentheses; this external pair of parentheses corresponds to the external membrane, called the skin. A membrane without any other membrane inside is said to be elementary. For instance, the structure in Figure 1 contains 8 membranes; membranes 3, 5, 6 and 8 are elementary. The string of parentheses identifying this structure is


All membranes are labeled; here we have used the numbers from 1 to 8. We say that the number of membranes is the degree of the membrane structure, while the height of the tree associated in the usual way with the structure is its depth. In the example above we have a membrane structure of degree 8 and of depth 4.

In what follows, the membranes can be marked with $+$ or $-$, and this is interpreted as an “electrical charge”, or with 0, and this means “neutral charge”. We will write $[i]_t^+$, $[i]_t^-$, $[i]_t^0$ in the three cases, respectively.
The membranes delimit regions, precisely identified by the membranes (the region of a membrane is delimited by the membrane and all membranes placed immediately inside it, if any such a membrane exists). In these regions we place objects, which are represented by symbols of an alphabet. Several copies of the same object can be present in a region, so we work with multisets of objects. A multiset over an alphabet \( V \) is represented by a string over \( V \): the number of occurrences of a symbol \( a \in V \) in a string \( x \in V^* \) (\( V^* \) is the set of all strings over \( V \); the empty string is denoted by \( \lambda \) ) is denoted by \( |x|_a \) and it represents the multiplicity of the object \( a \) in the multiset represented by \( x \).

A \( P \) system with active membranes and 2-division is a construct

\[
\Pi = (O, H, \mu, w_1, \ldots, w_m, R),
\]

where:

(i) \( m \geq 1 \) (the initial degree of the system);

(ii) \( O \) is the alphabet of objects;

(iii) \( H \) is a finite set of labels for membranes;

(iv) \( \mu \) is a membrane structure, consisting of \( m \) membranes, labelled (not necessarily in a one-to-one manner) with elements of \( H \);

(v) \( w_1, \ldots, w_m \) are strings over \( O \), describing the multisets of objects placed in the \( m \) regions of \( \mu \);

(vi) \( R \) is a finite set of developmental rules, of the following forms:

(a) \( [h(a \rightarrow v)]^n_h \),
for \( h \in H, \alpha \in \{+, -, 0\}, a \in O, v \in O^* \)
(object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);

(b) \( a[h]^\alpha_1 \rightarrow [h b]^\alpha_2 \),
for \( h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a, b \in O \)
(communication rules; an object is introduced in the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);
A sequence of transitions is a computation. A computation is halting if no other rules can be applied in types (b)-(f). In this way, we get transition from a configuration of the system to the next configuration.

For a detailed description of using these rules we refer to [4, 5]. Here we only mention that the rules are of types (a), (b), (c), (e), and (f) only (i.e., a P system with active membranes not using membrane dissolving rules of type (d)).

A P system with restricted active membranes is a P system with active membranes where the rules are of types (a), (b), (c), (e), and (f) only (i.e., a P system with active membranes not using membrane dissolving rules of type (d)).

A P system with restricted elementary active membranes is a P system with active membranes where the rules are of types (a), (b), (c), and (e) only (i.e., a P system with active membranes not using membrane dissolving rules of type (d) and the (f) type rules for non-elementary membrane division).

To understand what it means that a problem can be solved in polynomial time by P systems, it is necessary to recall some complexity measure for P systems as described in [7].

Consider a decision problem \( A \) and denote by \( A(n) \) an instance of \( A \) of size \( n \). Given a class \( X \) of membrane systems and a total function \( f : \mathbb{N} \to \mathbb{N} \) (for example, linear and polynomial functions), we
say that problem \( A \) belongs to \( MC_X(f) \) if a family of membrane systems \( \Pi_A = (\Pi_A(1), \Pi_A(2), \ldots) \) of type \( X \) exists such that:

1. \( \Pi_A \) is a \textit{uniform} family: there is a Turing machine which constructs \( \Pi_A(n) \) in polynomial time starting from \( n \).
2. Each \( \Pi_A(n) \) is \textit{confluent}: there is a distinguished object \( \text{yes} \) such that either in every computation of \( \Pi_A(n) \) the object \( \text{yes} \) is send out from the system, or this happens in no computation.
3. \( \Pi_A(n) \) is \textit{sound}: that is, \( \Pi_A(n) \) sends out the object \( \text{yes} \) if and only if the answer to \( \Pi_A(n) \) is “\text{yes}”.
4. \( \Pi_A \) is \textit{\( f \)-efficient}: that is, \( \Pi_A \) always halts in at most \( f(n) \) steps.

The linear and polynomial complexity classes associated with a family of membrane systems, \( X \), are defined as follows:

\[
\text{LMC}_X = \bigcup_{f \text{ linear}} \text{MC}_X(f), \quad \text{PMC}_X = \bigcup_{f \text{ polynomial}} \text{MC}_X(f).
\]

In [5], the definition of these complexity classes is based on a \textit{semi-uniform} construction of P systems solving a problem \( A \): one starts not from \( n \), but from an instance \( A(n) \) and we construct a system \( \Pi_{A(n)} \), which is said to solve \( A(n) \) if it is confluent and sound in the sense of the previous definition. If for all \( n \geq 1 \) the system \( \Pi_{A(n)} \) is \( f \)-efficient, then we say that \( A \) belongs to \( \text{MC}^S_X(f) \). For a clearer description of the difference between uniform P systems and semi-uniform P systems, please refer to [6].

Then, the classes \( \text{LMC}^S_X, \text{PMC}^S_X \) are defined in the same way as \( \text{LMC}_X, \text{PMC}_X \), starting from \( \text{MC}^S_X(f) \) for \( f \) linear functions and polynomials, respectively. Obviously, \( \text{LMC}_X \subseteq \text{LMC}^S_X, \quad \text{PMC}_X \subseteq \text{PMC}^S_X \), but it is not yet known whether the inclusions are proper.

In what follows, we use recognizing P systems. First of all, following [6, 7] we consider P systems with input. Such a device is a tuple \( (\Pi, \Sigma, i_0) \), where:

- \( \Pi \) is a P system, with the alphabet of objects \( \Gamma \) and initial multisets \( w_1, \ldots, w_m \) (associated with membranes labelled by \( 1, \ldots, m \), respectively).
- \( \Sigma \) is an (input) alphabet strictly contained in \( \Gamma \) and such that \( w_1, \ldots, w_m \) are multisets over \( \Gamma - \Sigma \).
- \( i_0 \) is the label of a distinguished membrane (of input).

If \( w \) is a multiset over \( \Sigma \), then the initial configuration of \( (\Pi, \Sigma, i_0) \) with input \( w \) is \( (\mu, w'_1, \ldots, w'_m) \), where \( w'_i = w \) for \( i \neq i_0 \), and \( w'_{i_0} = w_{i_0} \cup w \).

The computations of a P system with input are defined in a natural way. Note that the initial configuration is obtained by adding the input multiset \( w \) over \( \Sigma \) to the initial configuration of the system \( \Pi \).

Now, a recognizing P system is a P system with input, \( (\Pi, \Sigma, i_0) \), such that:

1. The alphabet of objects contains two distinguished elements \( \text{yes}, \text{no} \).
2. All computations of the system halt.
3. If \( C \) is a computation of \( \Pi \), then either the object \( \text{yes} \) or the object \( \text{no} \) (but not both) have to be sent out to the environment, and only in the last step of the computation.

We say that \( C \) is an accepting (respectively, rejecting) computation if the object \( \text{yes} \) (respectively, \( \text{no} \) appears in the environment in the halting configuration of \( C \).
3. Solving QBF-SAT Problem in Linear Time by Recognizing P Systems with Restricted Active Membranes

The QBF-SAT problem (satisfiability of quantified propositional formulas, QBF, in short, in the conjunctive normal form) is a well-known PSPACE-complete problem [8]. It asks whether or not a given quantified boolean formula in the conjunctive normal form assumes the value true.

A formula as above is of the form

$$\gamma = Q_1 x_1 Q_2 x_2 \cdots Q_n x_n (C_1 \land C_2 \land \cdots \land C_m),$$

where each $Q_i$, $1 \leq i \leq n$, is either $\forall$ or $\exists$, and each $C_j$, $1 \leq i \leq m$, is a clause of the form of a disjunction

$$C_j = y_1 \lor y_2 \lor \cdots \lor y_r,$$

with each $y_k$ being either a propositional variable, $x_s$, or its negation, $\neg x_s$. For example, let us consider the propositional formula

$$\beta = Q_1 x_1 Q_2 x_2 [(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2)].$$

It is easy to see that it is true when $Q_1 = \forall$ and $Q_2 = \exists$, but it is false when $Q_1 = \exists$ and $Q_2 = \forall$.

We will now consider a normal form for QBFs. For any QBF as above with $n$ variables there exists an equivalent QBF in the form $\gamma' = \exists x_1' \forall x_2' \cdots \exists x_n' \forall x_2', (C_1' \land C_2' \land \cdots \land C_m')$ with $2n$ variables, where each clause $C_j'$ is obtained from $C_j$ by replacing every variable $x_i$ by $x_{i+1}$ if $Q_i = \exists$, or by $\neg x_i$ if $Q_i = \forall$. For example, for the propositional formula $\beta_1 = \forall x_1 \exists x_2 [(x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2)]$, we have the normal form $\beta_1' = \exists x_1' \forall x_2' \exists x_3' [(x_1' \lor x_3') \land (\neg x_1' \lor \neg x_3')]$; for the propositional formula $\beta_2 = \exists x_1 \forall x_2 [(x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_2)]$, we have the normal form $\beta_2' = \exists x_1' \forall x_2' \exists x_4' [(x_1' \lor x_4') \land (\neg x_1' \lor \neg x_4')]$.

The following theorem shows that the QBF-SAT problem can be solved in linear time by a uniform family of recognizing P systems with restricted active membranes, thus partially solving the problem from [9].

**Theorem 3.1.** There exists a uniform family of recognizing P systems with restricted active membranes, solving QBF-SAT problem in a linear time with respect to the number of variables and the number of clauses.

**Proof:**

Let us consider a propositional formula

$$\gamma = \exists x_1 \forall x_2 \cdots \exists x_{2n-1} \forall x_{2n} (C_1 \land C_2 \land \cdots \land C_m).$$

The instance $\gamma$ is encoded as a (multi)set over $\Sigma((n, m)) = \{x_{i, j}, j \mid 1 \leq i \leq m, 1 \leq j \leq 2n\}$. The objects $x_{i, j}$ represent the variables $x_j$ appearing in the clauses $C_i$ without negation, and the objects $x'_{i, j}$ represent the variables $x_j$ appearing in the clauses $C_i$ with negation. Thus, the input multiset is

$$w = \{x_{i, j} \mid x_j \text{ in } C_i, 1 \leq i \leq m, 1 \leq j \leq 2n\} \cup \{x'_{i, j} \mid \neg x_j \text{ in } C_i, 1 \leq i \leq m, 1 \leq j \leq 2n\}.$$
where

\[ O(n,m) = \{x_{i,k,j}, x'_{i,k,j} \mid 1 \leq i \leq m, 1 \leq k \leq j \leq 2n\} \cup \{c_i \mid 0 \leq i \leq 2n\} \]
\[ \cup \{d_i \mid 1 \leq i \leq m + 1\} \cup \{r_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq 2n\} \]
\[ \cup \{f_i \mid 0 \leq i \leq 7n + 2m + 4\} \cup \{t, s, u, \text{yes, no}\}, \]

\[ H = \{0, 1, e, a\}, \]
\[ \mu = 1_{[a_{+}e \cdots a_{+}e]^0}1_{[a_{+}e \cdots a_{+}e]^0}^+ (2n + 2 \text{ nested membranes}), \]
\[ w_0 = c_0, \]
\[ w_1 = f_{7n+2m+4}, \]
\[ w_\alpha = w_e = \lambda, \]

and the set \( R \) contains the following rules (we also give explanations about the use of these rules during the computations):

1. \( [\alpha_{\varepsilon_{j+1}}]^0 \rightarrow [\alpha_{\varepsilon_{j+1}}]^0_0 [\alpha_{\varepsilon_{j+1}}]^0, \) for all \( 0 \leq j < 2n. \)

   In membrane with label 0, when it is “electrically neutral”, object \( c_j \) causes the membrane to divide and to subsequently choose for a variable \( x_j, 1 \leq j \leq 2n, \) both values \textit{true} and \textit{false}, in form of charges + and − of the two created membranes with the label 0.

2. \( [\alpha_{\varepsilon_{i,1,j}} \rightarrow r_{i,j}]^0_0, [\alpha_{\varepsilon_{i,1,j}} \rightarrow r_{i,j}]^0, \)
\[ [\alpha_{\varepsilon_{i,1,j}} \rightarrow \lambda]^0_0, [\alpha_{\varepsilon_{i,1,j}} \rightarrow \lambda]^0_0, \) for all \( 1 \leq i \leq m, 1 \leq j \leq 2n. \)

   These rules create objects \( r_{i,j} \) for each clause \( C_i \) satisfied by the \textit{true} value of variable \( x_j \) in a positively charged membrane, and by the \textit{false} value of \( x_j \) in a negatively charged membrane.

3. \( [\alpha_{\varepsilon_{i,k,j}} \rightarrow x_{i,k-1,}]^0_0, \) \( [\alpha_{\varepsilon_{i,1,j}} \rightarrow x_{i,k-1,}]^0_0, \alpha \in \{+ , -\}, \) for all \( 1 \leq i \leq m, 2 \leq k \leq j \leq 2n; \)
\( [\alpha_{\varepsilon_{i,j}} \rightarrow r_{i,j+1}]^0_0, \alpha \in \{+ , -\}, \) for all \( 1 \leq i \leq m, 1 \leq j \leq 2n - 1. \)

   These rules shift the variables, so that in rules 2, variables \( x_1, \cdots, x_{2n} \) are substituted one by one.

4. \( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0, \)
\( [a_{+}e]^0_0 \rightarrow [a_{+}e]^0_0. \)

These are division rules for membranes with labels \( a, e. \) The opposite polarization introduced when dividing a membrane 0 is propagated from lower levels to upper levels of the membrane structure and the membranes are continuously divided until also the child of the skin has been divided; this membrane remains polarized and hence may be never divided again; in the following cycle of the division process, the same holds for its children and so on, resulting in the structure as shown in Figure 2 after \( 4n \) steps. Note that after \( 4n \) steps, the polarization of the membranes with label 0 is neutral.

5. \( [\alpha_{\varepsilon_{j+1}}]^0_0 \rightarrow [\alpha_{\varepsilon_{j+1}}]^0_0. \)

At step \( 4n + 1, \) the counter object \( c_{2n} \) evolves to a new counter \( d_1. \) We now pass to the next phase of the computation – checking the satisfiability of clauses. At the same time, we look for the clauses satisfied by the truth–assignments from each membrane. This is done in parallel for all truth values present in each membrane (the rules are used in parallel, they do not actively involve
the membranes). After completing this step, the (only) membranes with label 0 which contain all symbols \( r_{1,2n}, r_{2,2n}, \ldots, r_{m,2n} \) meaning that the truth–assignments from those membranes satisfy all clauses. We will let those membranes send a suitable signal to the membranes with label 1 directly out of them. This will be done by the following rules of types 6-10.

6. \([0]^{r_{1,2n}}_0 \rightarrow [0] r_{1,2n}^{0}\).  
   For all \(2^n\) membranes with label 0, we check whether or not \(r_{1,2n}\) is present in each membrane. If this is the case, then \(r_{1,2n}\) is sent out of the membrane where it is present (one copy of \(r_{1,2n}\) exits the membrane where it is present, the other copies of \(r_{1,2n}\) will evolve to \(r_{0,2n}\) by rule 8, which will never evolve again), changing in this way the polarization of that membrane, to negative. The membranes which do not contain the object \(r_{1,2n}\) remain neutrally charged and they will no longer evolve, as no further rule can be applied to them.

7. \([0] d_i \rightarrow d_{i+1}^{0}\), 1 ≤ \(i\) ≤ \(m\).

8. \([0]^{r_{k,2n}} \rightarrow r_{k-1,2n}^{0}\), 1 ≤ \(k\) ≤ \(m\).

9. \(r_{1,2n}[0]^{0} \rightarrow [0] r_{0,2n}^{0}\).
   In the copies of membrane 0 with a negative polarization, hence those where we have found \(r_{1,2n}\) in the previous step, we perform two operations in parallel: we count the number of satisfied clauses, by means of increasing the subscript of \(d_k\), and we decrease the subscripts of all objects \(r_{j,2n}\) from that membrane. Thus, if the second clause was satisfied by the truth–assignment from a membrane, that is \(r_{2,2n}\) was present in a membrane, then \(r_{2,2n}\) becomes \(r_{1,2n}\) – hence the rule of type 6 can be applied again. Note the important fact that passing from \(r_{2,2n}\) to \(r_{1,2n}\) is possible only in membranes where we already had \(r_{1,2n}\), hence we check whether the second clause is satisfied only after knowing that the first clause was satisfied. At the same time, the object \(r_{1,2n}\) from the membrane with label \(e\) returns to membrane with label 0, changing to \(r_{0,2n}\), and returning the polarization of the membrane to neutral (this makes possible the use of rules of type 6).

The rules of types 6, 7, 8, 9 are applied as many times as possible (in one step rules of type 6, in the next one rules of types 7, 8, and 9, and then we repeat the cycle). Clearly, if a membrane does not contain an object \(r_{1,2n}\), then that membrane will stop evolving at the time when \(r_{1,2n}\) is supposed to reach the subscript 0, \(2n\). In this way, after \(2n\) steps we can find those membranes which contain all objects \(r_{1,2n}, r_{2,2n}, \ldots, r_{m,2n}\). The membranes with this property, and only they, will get the object \(d_{m+1}\). Note that at that moment the last object \(r_{1,2n}\), originating in \(r_{m,2n}\), enters the corresponding membrane, and changes its polarization to neutral.

10. \([0] d_{m+1}^{0} \rightarrow [0]^{0}\).
   The object \(d_{m+1}\) exits the membrane that has contained a truth–assignment that has satisfied all clauses, producing the signal object \(t\). This object \(t\) is then sent to the membranes with label \(e\). The fact that the membranes with label \(e\) contain \(t\) means that the corresponding truth–assignments satisfied all clauses. Now we pass to the next phase of the computation – checking the satisfiability of quantifiers.

11. \([e] t^{0} \rightarrow [e]^{0} t\), \(\alpha \in \{+, -, \}\).
A. Alhazov et al. / Solving a PSPACE-Complete Problem by Active P Systems 9

Figure 2. The membrane structure of the system after 4n steps.

12. \[ [a^t]^\alpha_a \rightarrow [a^0]^\alpha_a, \alpha \in \{+, -\}, [a^t]^0 \rightarrow [a^0]. \]
   The object \( t \) exits the membranes, directly including the elementary membranes. Membranes with label \( a \) correspond to quantifications \( Q_{2j} = \forall \). The object \( t \) is passed to the upper level only if it comes from both lower level membranes, i.e., the respective clauses are satisfied for both truth values of \( x_j \). Membranes with label \( e \) (except those directly containing membranes with label 0) correspond to quantifications \( Q_{2j-1} = \exists \), hence a single object \( t \) coming from lower level membrane is enough.

13. \[ [1]^+ \rightarrow [1]^0 \gamma \text{es}. \]
   The object \( t \) exits the skin producing an object \( \gamma \text{es} \), telling us that the formula \( \gamma \) is satisfiable, and the computation halts. Note that rule 13 changes the polarization of the skin membrane to neutral in order that the object \( t \) (if exists) remaining in it is not able to continue evolving.

14. \[ [1]^+ \rightarrow [1]^0 \gamma \text{es}. \]
   At step \( 7n + 2m + 4 \), the skin membrane has object \( f_0 \), originating in \( f_{7n+2m+4} \). Note that if the formula \( \gamma \) is not satisfiable, then at this moment the polarization of the skin membrane is positive. In the next step, \( f_0 \) produces the object no, which is sent to the environment, telling us that the formula \( \gamma \) is not satisfiable, and the computation stops.

From the previous explanation of the use of rules, one can easily see how this P system works. It is clear that the object \( \gamma \text{es} \) is sent to the environment if and only if the formula \( \gamma \) is satisfiable. This is achieved in \( 7n + 2m + 4 \) steps in \( 4n \) steps we create the membrane structure in Figure 2 (as well as the \( 2^{2n} \) different truth-assignments), then we create new counter \( d_1 \) and objects \( r_{1,2n}, \ldots, r_{m,2n} \) (1 further step) and we check the satisfiability of clauses, in parallel in the \( 2^{2n} \) sub-structures (this takes further \( 2m + 1 \) steps); then object \( t \) exits the membranes containing the elementary membranes (one further step) and further \( 3n \) steps are necessary to check whether all quantifiers are satisfied by propagating object \( t \) through the indicated binary tree structure. One step is needed for each \( \exists \) quantifier, while two steps are necessary for \( \forall \); finally the system sends out the object \( \gamma \text{es} \) telling us that the formula \( \gamma \) is
satisfiable (one further step). If formula $\gamma$ is not satisfiable, then at step $7n + 2m + 5$ the system sends the object no to the environment. Therefore, the family of membrane systems we have constructed is sound, confluent, and linearly efficient.

To prove that the family is uniform in the sense of section 1, we have to show that for a given size, the construction of P systems described in the proof can be done in polynomial time by a Turing machine. We omit the detailed construction due to the fact that it is straightforward but cumbersome as explained in the proof of Theorem 7.2.3 in [5], although P systems in [5] are semi-uniform. There is one point that needs to be mentioned: the reduction of the QBF-SAT problem to the normal form can be also done in polynomial time. So QBF-SAT problem was decided in linear time $(7n + 2m + 5)$ by recognizing P systems with restricted active membranes, and this concludes the proof. $\square$

4. Conclusions

We have shown that QBF-SAT problem can be solved in linear time by a uniform family of recognizing P systems with restricted active membranes. So we give a partial answer to the conjecture [9] that QBF-SAT problem can be solved by P systems with active membranes and 2-division omitting the membrane dissolving rules of the (d) type, but probably not the (f) type rules for non-elementary membrane division.

It is open whether QBF-SAT problem can be solved in linear time by P systems with restricted elementary active membranes.

The solution presented here differs from other solutions to NP-complete problems given by Gh. Păun [4], P. Sosík [9], et al. in the following sense: a family of recognizing P systems with active membranes is constructed, associated with the problem that is being solved, in such a way that all the instances of the problem that have the same size are processed by the same P system (to which an appropriate input, that depends on the concrete instance, is supplied). In [4, 9] one works with semi-uniformly constructed P systems, associated with each one of the instances of the problem.

Let us denote by $\mathcal{AM}$ the class of restricted recognizing P systems with active membranes and with 2-division. Then from Theorem 3.1 we have QBF-SAT $\in \mathcal{PMC}_{\mathcal{AM}}$. Because the class $\mathcal{PMC}_{\mathcal{AM}}$ is stable under polynomial time reduction, we have $\text{PSPACE} \subseteq \mathcal{PMC}_{\mathcal{AM}}$, which strengthens $\text{PSPACE} \subseteq \mathcal{PMC}^S_{\mathcal{AM}}$ [9]. Notice that both classes $\text{NP}$ and $\text{co-NP}$ are contained in $\text{PSPACE}$, although it is not known whether these inclusions are strict or not. So our result generalizes the following two results: $\text{NP} \subseteq \mathcal{PMC}_{\mathcal{AM}}$ [6], $\text{co-NP} \subseteq \mathcal{PMC}_{\mathcal{AM}}$ [7].

Finally, we point out two open problems:

1. Does $\text{PSPACE} = \mathcal{PMC}_{\mathcal{AM}}$ hold?

2. Find a class $\mathcal{F}$ of recognizing P systems such that $\text{NP} \cup \text{co-NP} = \mathcal{PMC}_{\mathcal{F}}$ or $\text{PSPACE} = \mathcal{PMC}_{\mathcal{F}}$.

References


