A New Approach for Solving SAT by P Systems with Active Membranes*

Zsolt Gazdag and Gábor Kolonits

Department of Algorithms and their Applications
Faculty of Informatics
Eötvös Loránd University
{gazdagzs,kolomax}@inf.elte.hu

Abstract. In this paper we give two families of P systems with active membranes that can solve the satisfiability problem of propositional formulas in linear time in the number of propositional variables occurring in the input formula. These solutions do not use polarizations of the membranes or non-elementary membrane division but use separation rules with relabeling. The first solution is a uniform one, but it is not polynomially uniform. The second solution, which is based on the first one, is a polynomially semi-uniform solution.

Keywords: Membrane computing; P systems; SAT problem

1 Introduction

P systems are biologically inspired computational models introduced in [8] (for a comprehensive guide see e.g. [10]). A widely investigated variant of these systems are P systems with active membranes [9]. Here the P systems have the possibility of dividing elementary membranes which, combined with the maximal parallelism presented in these models, can yield exponential workspace in linear time. This feature is commonly used in efficient solutions of NP complete problems, e.g. in the solution of the satisfiability problem of propositional formulas (SAT). SAT is probably the best known NP-complete decision problem; the question is whether a given propositional formula in conjunctive normal form (CNF) is satisfiable.

Solving SAT efficiently by P systems with active membranes is a widely investigated area of membrane computing (see e.g. [1], [2], [4], [7], [9], and [12]). These solutions differ, for example, in the types of the rules employed, the number of possible polarizations of the membranes, and the derivation strategy (maximal or minimal parallelism - this latter concept was introduced in [3]). On the other hand, these solutions commonly work in a way where all possible truth valuations of the input formula are created and then a satisfying one (if it exists) is chosen.

* This research was supported by the project TÁMOP-4.2.1/B-09/1/KMR-2010-003 of Eötvös Loránd University.

In these works the used families of P systems are constructed in a polynomially (semi-)uniform way. This means that the P systems in these families can be constructed in polynomial time by a deterministic Turing machine from the size of the input formula (in the uniform case) or from the formula itself (in the semi-uniform case). The size of the input formula is usually described by the number n of distinct variables and the number m of clauses of the formula. (For more details on polynomially (semi-)uniform families of P systems please refer to [11] or [12].)

The P systems introduced in the above works can solve SAT in polynomial time in $n + m$. In particular, in [4] SAT is solved in linear time in n (i.e., the number of steps of the system is independent from m), but there division of non-elementary membranes is allowed, and the derivation strategy is minimally parallel instead of the commonly used maximal parallel one.

In this paper we give two families of P systems that can solve SAT in linear time in n. Our motivation was to give solutions where the number of the computation steps is independent from the number of the clauses in the input formula and the systems do not use non-elementary membrane division. Our first solution is a uniform one, but the constructed P systems have exponential number of objects and rules in n, i.e., this solution is not polynomially uniform. On the other hand, our second solution, which is based on the first one, is a polynomially semi-uniform solution.

Clearly, it is desirable that a solution of SAT by a P system be polynomially (semi-)uniform. Indeed, in a non-polynomially (semi-)uniform solution there is a possibility of computing the satisfiability of the input formula already during the construction of the P system. If this is the case, then SAT is in fact solved during the construction of the P systems, and not by the P systems itself. To demonstrate that we do not use such a "misleading" construction, we briefly describe the method that we use in our uniform solution.

Let φ be a formula in CNF over n variables. Then there is an equivalent formula φ' in CNF such that every clause of φ' contains every variable of φ negated or without negation. Such clauses are called complete clauses. It can be seen that φ' is satisfiable if and only if it does not contain every possible complete clause over n variables. We will show that our P systems can create φ' from φ and decide if φ' contains every complete clauses over n variables in linear number of steps. Clearly, the cardinality of the set of all complete clauses over n variables is exponential in n. This implies that the cardinality of the object alphabet of our P systems in our uniform solution is also exponential in n. Thus these P systems can not be constructed in polynomial time in n, even if the number m of the clauses in the input formula is polynomial in n (notice that, in general, m can be exponential in n as well).

Despite the fact that the above described systems cannot be constructed in a polynomially uniform way, we think that they are still interesting since, as we have seen above, the construction of these systems does not compute the satisfiability of the input and the solution is uniform. This latter property yields that once we have constructed our P system for a given number n, then we can use it for deciding the satisfiability of every formula having n distinct variables. Moreover, the decision is done in linear number of steps in n and to achieve this efficiency we did not have to use non-elementary membrane division.

Our other solution is a polynomially semi-uniform one based on the uniform solution described above. Here we implemented a method that does not create every possible complete clause but uses several copies of the original clauses of the input formula. The price of this improvement is that we could not make this solution to be uniform.

This paper is an improved version of the paper [5]. The present paper is organised as follows. In Sect. 2 we give the necessary definitions and preliminary results. Sections 3 contains our families of P systems, and Sect. 4 presents some conclusions and remarks.

2 Definitions

Alphabets, Words, Multisets. An *alphabet* Σ is a nonempty and finite set of symbols. The elements of Σ are called *letters.* Σ^* denotes the set of all finite *words* (or *strings*) over Σ, including the *empty word* ε. We will use *multisets* of objects in the membranes of a P system. As usual, these multisets will be represented by strings over the object alphabet of the P system.

The SAT Problem. Let $X = \{x_1, x_2, x_3, \ldots\}$ be a recursively enumerable set of *propositional variables* (*variables*, to be short), and, for every $n \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural numbers, let $X_n := \{x_1, \ldots, x_n\}$. An *interpretation of the variables in* X_n (or just an *interpretation* if X_n is clear from the context) is a function $\mathcal{I} : X_n \to \{true,\ false\}$.

The variables and their negations are called *literals.* A *clause* C is a disjunction of finitely many pairwise different literals satisfying the condition that there is no $x \in X$ such that both x and $\bar{x}$ occur in C, where $\bar{x}$ denotes the negation of x. The set of all clauses over the variables in X_n is denoted by $\mathcal{C}_n$. A *formula in conjunctive normal form* (CNF) is a conjunction of finitely many clauses. We will often treat formulas in CNF as finite sets of clauses, where the clauses are finite sets of literals. A clause $C \in \mathcal{C}_n$ is called a *complete clause* if, for every $x \in X_n$, $x \in C$ or $\bar{x} \in C$. Let $Form$ be the set of all formulas in CNF over the variables in X and, for every $n \in \mathbb{N}$, let $Form_n$ be the set of those formulas in $Form$ that have variables in X_n. It is easy to see that $Form$ is a recursively enumerable set (notice that, for a given $n \in \mathbb{N}$, $Form_n$ is a finite set).

Let $\varphi \in Form_n$ ($n \in \mathbb{N}$) and let $\mathcal{I}$ be an interpretation for φ. We say that $\mathcal{I}$ *satisfies* φ, denoted by $\mathcal{I} \models \varphi$, if φ evaluates to *true* under the truth assignment defined by $\mathcal{I}$. Note that $\mathcal{I} \models \varphi$ if and only if, for every $C \in \varphi$, $\mathcal{I} \models C$. We say that φ is satisfiable if there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \varphi$. The SAT *problem* (boolean satisfiability problem of propositional formulas in CNF) can be defined as follows. Given a formula φ in CNF, decide whether or not there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \varphi$.

Let $\varphi_1, \varphi_2 \in Form_n$ ($n \in \mathbb{N}$). We say that φ_1 *and* φ_2 *are equivalent*, denoted by $\varphi_1 \sim \varphi_2$, if, for every interpretation $\mathcal{I}$, $\mathcal{I} \models \varphi_1$ if and only if $\mathcal{I} \models \varphi_2$. Let $\varphi \in Form$. The set of variables occurring in φ, denoted by $var(\varphi)$, is defined by $var(\varphi) := \{x \in X \mid \exists C \in \varphi : x \in C \text{ or } \bar{x} \in C\}$. For a clause $C \in \mathcal{C}_n$ and a set $Y \subseteq X_n$ ($n \in \mathbb{N}$) such that $var(C) \cap Y = \emptyset$, let C_Y be the following set of clauses. Assume that $Y = \{x_{i_1}, \ldots, x_{i_k}\}$ ($k \leq n$, $1 \leq i_1 < \ldots < i_k \leq n$). Then let $C_Y := \{C \cup \{l_1, \ldots, l_k\} \mid 1 \leq j \leq k : l_j \in \{x_{i_j}, \bar{x}_{i_j}\}\}$. For a formula $\varphi = \{C_1, \ldots, C_m\} \in Form_n$ ($m, n \in \mathbb{N}$), let $\varphi' := \bigcup_{C \in \varphi} C_Y$, where $Y := X_n - var(C)$.

The correctness of the P systems that we are going to construct to solve SAT is based on the following statement which can be easily proved by standard arguments of propositional logic (see also e.g. [6] for deciding SAT by means of complete clauses).

Proposition 1. *For a formula $\varphi = \{C_1, \ldots, C_m\} \in Form_n$ ($m, n \in \mathbb{N}$), φ' contains every complete clause in $\mathcal{C}_n$ if and only if φ is unsatisfiable.*

Proof. Let $\varphi := \{C_1, \ldots, C_m\} \in Form_n$ ($m, n \in \mathbb{N}$). We prove the above statement in two steps. First, we show that $\varphi \sim \varphi'$, then we show that φ' is unsatisfiable if and only if it contains every complete clause in $\mathcal{C}_n$.

To see that $\varphi \sim \varphi'$ we show that, for every interpretation $\mathcal{I}$, $\mathcal{I} \models \varphi$ if and only if $\mathcal{I} \models \varphi'$. Let $\mathcal{I}$ be an interpretation and assume first that $\mathcal{I} \models \varphi$. Let $C \in \varphi$. Then $\mathcal{I} \models C$ and, moreover, for every $C' \in C_Y$, where $Y = X_n - var(C)$, $var(C) \subseteq var(C')$. This clearly implies that, for every $C' \in C_Y$, $\mathcal{I} \models C'$. It follows then that $\mathcal{I} \models \varphi'$ as well.

Now assume that $\mathcal{I} \models \varphi'$. We show that $\mathcal{I} \models C$, for every $C \in \varphi$, which clearly implies that $\mathcal{I} \models \varphi$. Let $C \in \varphi$ and $Y := X_n - var(C)$. Assume that $Y = \{x_{i_1}, \ldots, x_{i_k}\}$ ($k \leq n$, $1 \leq i_1 < \ldots < i_k \leq n$). Let $C' := C \cup \{l_{i_1}, \ldots, l_{i_k}\}$ be that clause in C_Y which satisfies the following property. For every $1 \leq j \leq k$, $l_{i_j} = \bar{x}_{i_j}$ if $\mathcal{I}(x_{i_j}) = true$, and $l_{i_j} = x_{i_j}$ otherwise. Clearly, $\mathcal{I} \models C'$, but $\mathcal{I} \not\models \{l_{i_1}, \ldots, l_{i_k}\}$. This implies that $\mathcal{I}$ should satisfy C.

Next, we show that φ' is unsatisfiable if and only if it contains every complete clauses in $\mathcal{C}_n$. Assume first that φ' contains every complete clauses in $\mathcal{C}_n$ and let $\mathcal{I}$ be an arbitrary interpretation of the variables in X_n. Let $C' = \{l_1, \ldots, l_n\}$ be that clause in $\mathcal{C}_n$ which satisfies the following property. For every $1 \leq i \leq n$, $l_i = \bar{x}_i$ if $\mathcal{I}(x_i) = true$, and $l_i = x_i$ otherwise. Clearly $\mathcal{I} \not\models C'$ which, since $C' \in \varphi'$, means that $\mathcal{I} \not\models \varphi'$. Thus φ' is unsatisfiable.

Assume now that φ' does not contain every complete clauses and let $C' := \{l_1, \ldots, l_n\}$ be a clause that does not occur in φ'. Let $\mathcal{I}$ be the interpretation defined as follows. For every $1 \leq i \leq n$, let $\mathcal{I}(x_i) := true$ if $l_i = \bar{x}_i$, and let $\mathcal{I}(x_i) := false$ otherwise. It can be seen that, for every $C \in \varphi'$, there is a literal $l \in C$ such that $\mathcal{I}(l) = true$. It follows then that $\mathcal{I}$ satisfies every clause in φ'. Thus φ' is satisfiable which completes the proof.

P Systems with Active Membranes. We will use P systems with active membranes to solve SAT. In particular, we will use a model where a certain kind of separation rules is allowed. These separation rules have the possibility of changing the labels of the membranes involved. On the other hand, we will not use the polarizations of the membranes, thus we leave out this feature from the definition of these systems. The following is the formal definition of the P systems we will use (see also [10]).

A *(polarizationless) P system with active membranes* is a construct $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$, where:

- $m \geq 1$ (the *initial degree* of the system);
- O is the *alphabet of objects*;
- H is a finite set of *labels* for membranes;
- μ is a *membrane structure*, consisting of m membranes, labelled (not necessarily in a one-to-one manner) with elements of H;
- $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* (every symbol in a string representing one copy of the corresponding object) placed in the m regions of μ;
- R is a finite set of *developmental rules*, of the following forms:
  (a) $[a \to v]_h$, for $h \in H, a \in O, v \in O^*$
  (object evolution rules, associated with membranes and depending on the label 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) $a[\ ]_h \to [b]_h$, for $h \in H,\ a, b \in O$
  (communication rules, sending an object into a membrane; the label cannot be modified);
  (c) $[a]_h \to [\ ]_h b$, for $h \in H,\ a, b \in O$
  (communication rules; an object is sent out of the membrane, possibly modified during this process; the label cannot be modified);
  (d) $[a]_h \to b$, for $h \in H,\ a, b \in O$
  (dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);
  (e) $[a]_h \to [b]_h[c]_h$, for $h \in H,\ a, b, c \in O$
  (division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label; the object a specified in the rule is replaced in the two new membranes by (possibly new) objects b and c respectively, and the remaining objects are duplicated);
  (f) $[\ ]_{h_1} \to [K]_{h_2}[O - K]_{h_3}$, for $h_1, h_2, h_3 \in H,\ K \subset O$
  (2-separation rules for elementary membranes, with respect to a given set of objects; the membrane is separated into two new membranes with possibly different labels; the objects from each set of the partition of the set O are placed in the corresponding membrane).

As usual, Π works in a *maximal parallel* manner:

- In one step, any object of a membrane that can evolve must evolve, but one object can be used by only one rule of types (a)-(e);

– when some rules of type (b)-(f) can be applied to a certain membrane, then one of them must be applied, but a membrane can be the subject of only one rule of these rules during each step.

We say that Π is a *recognizing P system* if O has two designated objects *yes* and *no*, and every computation of Π halts and sends out to the environment either *yes* or *no*. We say that Π is a *recognizing P system with input* if (1) Π is a recognizing P system, (2) it has a designated input membrane i_0, and (3) a string w, called the *input of* Π, can be added to the system by placing it into the region i_0 in the initial configuration. A recognizing P system Π (with input) is called *deterministic* if it has only a single computation from its initial configuration to its unique halting configuration.

We say that *SAT can be solved in linear time by a uniform family* $\Pi := (\Pi(i))_{i\in\mathbb{N}}$ *of recognizing P systems with input*, if the following holds:

(1) for every $n \in \mathbb{N}$, $\Pi(n)$ can be constructed from n by a deterministic Turing machine in polynomial time in n;
(2) for a given formula $\varphi \in Form$ with size n ($n \in \mathbb{N}$), starting $\Pi(n)$ with a polynomial time encoding of φ in its input membrane, $\Pi(n)$ sends out to the environment *yes* if and only if φ is satisfiable;
(3) for every $n \in \mathbb{N}$, the computation of $\Pi(n)$ always halts in linear number of steps in n.

If in the above definition in condition (1) we do not require the Turing machine to be a polynomial time one, then we say that *SAT can be solved in weak linear time by* Π.

Now we give a similar definition corresponding semi-uniform families of recognizing P systems. We say that *SAT can be solved in linear time by a semi-uniform family* $\Pi := (\Pi(\varphi))_{\varphi\in Form}$ *of recognizing P systems* if, for every $\varphi \in Form$, the following holds:

(1) $\Pi(\varphi)$ can be constructed from φ by a deterministic Turing machine in polynomial time in the size of φ;
(2) $\Pi(\varphi)$ sends out to the environment *yes* if and only if φ is satisfiable;
(3) the computation of $\Pi(\varphi)$ always halts in linear number of steps in the size of φ.

For more details on complexity classes defined by (semi-)uniform families of P systems see e.g. [11] or [12].

3 The Main Results

Here we give our solutions for deciding SAT by P systems with active membranes. The first solution is a uniform but non-polynomial one; the second solution is a polynomially semi-uniform one. First, we discuss how we will encode the formulas in our P systems.

Encoding SAT Instances. Usually, when SAT is solved by a computation device, the formulas are encoded appropriately so that the model can process the formula. Clearly, the used encoding should be carried out efficiently, otherwise it is not ensured that the encoding phase does not compute also the satisfiability of the formulas. According to this, the encoding we use is rather trivial: we use symbols that are in one-to-one correspondence with the clauses in $\mathcal{C}_n$ ($n \in \mathbb{N}$). For every $n \in \mathbb{N}$, let O_n be an alphabet with a bijection between $\mathcal{C}_n$ and O_n. For a symbol $c \in O_n$, we denote the corresponding clause in $\mathcal{C}_n$ by $\hat{c}$. Thus, a formula $\varphi = \{C_1, \ldots, C_m\}$ ($m \in \mathbb{N}$) will be encoded in our membrane systems by the set of objects $cod(\varphi) := \{c_1, \ldots, c_m\} \subseteq O_n$, where, for every $1 \leq i \leq m$, $\hat{c}_i = C_i$. We will need a copy of the symbols in $cod(\varphi)$ thus we will also use the set $cod'(\varphi) := \{c' \mid c \in cod(\varphi)\}$.

The Uniform Solution. Here we define a uniform family $\mathbf{\Pi} := (\Pi(i))_{i \in \mathbb{N}}$ of recognizer P systems with input that solves SAT in weak linear time.

Definition 1. *For every $n \in \mathbb{N}$, let $\Pi(n) := (O, H, \mu, w_1, w_2, w_3, R)$, where:*

- $O := O_n \cup \{d_1, \ldots, d_{n+3}, yes, no\}$;
- $H := \{1, \ldots, n+3\}$;
- $\mu := [[[\ ]_3]_2]_1$, *where the input membrane is* $[\ ]_3$;
- $w_1 := \varepsilon, w_2 := d_1$ *and* $w_3 := \varepsilon$;
- *R is the set of the following rules (in some cases we also give explanations of the presented rules):*
  - *(a)* $[c \to c_1 c_2]_{i+2}$, *for every* $1 \leq i \leq n$ *and* $c, c_1, c_2 \in O_n$ *with* $x_i, \bar{x}_i \notin \hat{c}$, $\hat{c}_1 = \hat{c} \cup \{x_i\}$ *and* $\hat{c}_2 = \hat{c} \cup \{\bar{x}_i\}$
    (for every $1 \leq i \leq n$, these rules will replace those clauses in membrane $i+2$ that do not contain x_i or $\bar{x}_i$ by two other clauses, a clause that additionally contains x_i, and another one that contains $\bar{x}_i$);
  - *(b)* $[\ ]_{i+2} \to [K_i]_{i+3}[O - K_i]_{i+3}$, *for every* $1 \leq i \leq n$ *and* $K_i = \{c \in O_n \mid x_i \in \hat{c}\}$
    (for every $1 \leq i \leq n$, these rules will separate the objects in membranes with label $i+2$ according to that whether the clauses represented by the objects contain x_i or not; the new membranes will have label $i+3$);
  - *(c)* $[d_i \to d_{i+1}]_2$, *for every* $1 \leq i \leq n+2$;
  - *(d)* $[c]_{n+3} \to \varepsilon$, *for every* $c \in O_n$ *such that $\hat{c}$ is a complete clause in* $\mathcal{C}_n$;
  - *(e)* $d_{n+2}[\ ]_{n+3} \to [yes]_{n+3}$,
    $[yes]_{n+3} \to [\ ]_{n+3} yes$,
    $[yes]_2 \to [\ ]_2 yes$,
    $[yes]_1 \to [\ ]_1 yes$;
  - *(f)* $[d_{n+2}]_2 \to [d_{n+3}]_2[no]_2$,
    $[no]_2 \to [\ ]_2 no$,
    $[no]_1 \to [\ ]_1 no$.

Next we give an example to demonstrate how the P systems defined above create new clauses from the input and separate them into new membranes.

Example 1. We show the working of $\Pi(3)$ on a formula in $Form_3$. For the better readability, we denote the variables x_1, x_2, x_3 by x, y and z, respectively. Moreover, the objects in O_3 are denoted by sequences of literals occurring in the corresponding clauses of the formula, i.e., the symbols in O_3 are now strings over the set of literals.

Let the input formula be $\varphi := \{\{x, y, z\}, \{\bar{x}\}, \{\bar{y}\}, \{\bar{z}\}\}$. Then $\Pi(3)$ is started with symbols $xyz, \bar{x}, \bar{y}, \bar{z}$ in the input membrane, thus at the beginning the initial configuration looks as follows:

$$[d_1\,[[xyz, \bar{x}, \bar{y}, \bar{z}]_3]_2]_1.$$

In the first step, the system creates $x\bar{y}$ and $\bar{x}\bar{y}$ from $\bar{y}$, and $x\bar{z}$ and $\bar{x}\bar{z}$ from $\bar{z}$. Moreover, two new membranes with label 4 are created and the system puts $xyz, x\bar{y}$ and $x\bar{z}$ into the first new membrane and $\bar{x}, \bar{x}\bar{y}$ and $\bar{x}\bar{z}$ into the second one. Thus, after the first step the configuration of the system looks as follows:

$$[d_2\,[[xyz, x\bar{y}, x\bar{z}]_4, [\bar{x}, \bar{x}\bar{y}, \bar{x}\bar{z}]_4]_2]_1.$$

Then, in the next step, the system creates the clauses $xy\bar{z}$, $x\bar{y}\bar{z}$ from $x\bar{z}$, $\bar{x}y$, $\bar{x}\bar{y}$ from $\bar{x}$, and $\bar{x}y\bar{z}$, $\bar{x}\bar{y}\bar{z}$ from $\bar{x}\bar{z}$. Moreover, two new membranes with label 5 are created from each membranes with label 4, and the symbols are separated into these new membranes. Thus, after the second step, the system has the following configuration:

$$[d_3\,[[xyz, xy\bar{z}]_5, [x\bar{y}, x\bar{y}\bar{z}]_5, [\bar{x}y, \bar{x}y\bar{z}]_5, [\bar{x}\bar{y}, \bar{x}\bar{y}, \bar{x}\bar{y}\bar{z}]_5]_2]_1.$$

Finally, after the third step, the configuration of the system is:

$$[d_4\,[[xyz]_6, [xy\bar{z}]_6, [x\bar{y}z]_6, [x\bar{y}\bar{z}, x\bar{y}\bar{z}]_6,$$
$$[\bar{x}yz]_6, [\bar{x}y\bar{z}, \bar{x}y\bar{z}]_6, [\bar{x}\bar{y}z, \bar{x}\bar{y}z]_6, [\bar{x}\bar{y}\bar{z}, \bar{x}\bar{y}\bar{z}, \bar{x}\bar{y}\bar{z}]_6]_2]_1.$$

In general, the computation of $\Pi(n)$ for some $n \in \mathbb{N}$, when the membrane with label 3 contains the string $c_1 \dots c_m$ encoding a formula $\varphi = \{\hat{c}_1, \dots, \hat{c}_m\} \in Form_n$ $(m \in \mathbb{N})$ can be described as follows:

– During the first step, rules in (a) replace in the membrane with label 3 every object c with the property that $\hat{c}$ does not contain x_1 or $\bar{x}_1$ with two objects representing the clauses $\hat{c} \cup \{x_1\}$ and $\hat{c} \cup \{\bar{x}_1\}$. In parallel to this step, a rule in (b) separates the resulting objects into new membranes with label 4, depending on whether the clauses represented by the objects contain x_1 or not. Moreover, in membrane with the label 2, the object d_1 evolves to d_2 by the corresponding rule in (c).

– After n steps, the membrane system contains 2^n membranes with label $n+3$. Each such membrane can contain an object in O_n corresponding to a complete clause in $\mathcal{C}_n$. At this point the computation can continue in two different cases.
Case 1:

- If each of the membranes with label $n+3$ contains at least one object $c \in O_n$ such that $\hat{c}$ is a complete clause, then the system dissolves these membranes in one step by using the rules in (d). In parallel, d_{n+1} evolves to d_{n+2}.
- In the next step, using the first rule in (f), the system divides the membrane with label 2, and introduces the symbol *no*.
- In the last two steps, the symbol *no* goes out to the environment, and the computation halts.

Case 2:

- If there is at least one membrane with label $n+3$ that does not contain an object $c \in O_n$ such that $\hat{c}$ is a complete clause, then only the first rule in (e) can be applied, introducing the symbol *yes* (notice that the division rule in (f) cannot be applied as the membrane with label 2 is not elementary in this case).
- In the last three steps of the system, the symbol *yes* goes out to the environment, and the computation halts.

Notice that the membranes with label $n+3$ can contain objects representing complete clauses only.

It is not difficult to see that $\Pi(n)$ works correctly. Indeed, $\Pi(n)$ sends in every computation to the environment either the symbol *no* or the symbol *yes*. The symbol *no* can be introduced only in *Case 1* above, but in this case φ' must contain every complete clause in $\mathcal{C}_n$, and it follows from Proposition 1 that φ is not satisfiable. On the other hand, *yes* can be introduced only in *Case 2*, but in this case there is a complete clause in $\mathcal{C}_n$ that does not occur in φ', which, again by Proposition 1 means that φ is satisfiable. Moreover, it is easy to see that, for every formula $\varphi \in Form_n$, $\Pi(n)$ halts in $n+5$ steps. Thus we have the following theorem.

Theorem 1. *The SAT problem can be solved in weak linear time by a uniform family* $\mathbf{\Pi} := (\Pi(i))_{i \in \mathbb{N}}$ *of polarizationless recognizing P systems with input with the following properties: the elements of* $\mathbf{\Pi}$ *are deterministic, do not use non-elementary membrane division, and the size of an input formula is described by the number of variables occurring in the formula.*

The Semi-Uniform Solution. Here we give a polynomially semi-uniform family of recognizer P systems that solves SAT in linear time. This solution is strongly based on the family of P systems defined in Definition 1. Clearly, the main issue with a P system $\Pi(n)$ $(n \in \mathbb{N})$ of that family is that it can not be constructed in polynomial time in n. As we have mentioned, the reason is that $\Pi(n)$ creates complete clauses from the clauses of the input formula, and the number of these complete clauses can be exponential in n. On the other hand, one can note that the answer of $\Pi(n)$ depends only on whether or not every membrane with label $n+3$ contains at least one object regardless of whether the set of these objects contains every complete clause or not. Thus, one way to turn $\Pi(n)$ into a polynomially semi-uniform solution of SAT is to modify $\Pi(n)$ such that it does not create new objects representing clauses but reuses appropriately the original clauses of the input formula in every step of the computation. The following is the formal definition of a family of P systems where we implemented the above described idea.

Definition 2. *Let* $\mathbf{\Pi} := (\Pi(\varphi))_{\varphi \in Form}$*, where* $\Pi(\varphi)$ *for some* $\varphi \in Form$ *is defined as follows.* $\Pi(\varphi) := (O, H, \mu, w_1, w_2, w_3, R)$*, where:*

- $O := cod(\varphi) \cup cod'(\varphi) \cup \{d_1, \dots, d_{n+3}, yes, no\}$;
- $H := \{1, \dots, n+3\}$;
- $\mu := [[[\ ]_3]_2]_1$;
- $w_1 := \varepsilon, w_2 := d_1$ *and* $w_3 := cod(\varphi)$;
- *R is the set of the following rules:*
  - *(a1)* $[c \to c']_{i+2}$*, for every* $1 \le i \le n$ *and* $c \in cod(\varphi)$ *with* $\bar{x}_i \in \hat{c}$
    (for every $1 \le i \le n$*, these rules replace in membrane* $i+2$ *every symbol* c *representing a clause which contains* $\bar{x}_i$ *by its primed version* c'*;*
  - *(a2)* $[c' \to c]_{i+2}$*, for every* $1 \le i \le n$ *and* $c' \in cod'(\varphi)$ *with* $x_i \in \hat{c}$
    (for every $1 \le i \le n$*, these rules replace in membrane* $i+2$ *every symbol* c' *representing a clause which contains* x_i *by the symbol* c*;*
  - *(a3)* $[c \to cc']_{i+2}$ *and* $[c' \to cc']_{i+2}$ *for every* $1 \le i \le n$ *and* $c \in cod(\varphi)$ *with* $x_i, \bar{x}_i \notin \hat{c}$
    (for every $1 \le i \le n$*, these rules duplicate those symbols in membrane* $i+2$ *that represent such clauses which do not contain* x_i *or* $\bar{x}_i$*;*
  - *(b)* $[\ ]_{i+2} \to [K]_{i+3}[O-K]_{i+3}$*, for every* $1 \le i \le n$*, where* $K = cod(\varphi)$
    (for every $1 \le i \le n$*, these rules will separate the objects in membranes with label* $i+2$ *according to that whether they are primed or not;*
  - *(c)* $[d_i \to d_{i+1}]_2$*, for every* $1 \le i \le n+2$*;*
  - *(d)* $[c]_{n+3} \to \varepsilon$*, for every* $c \in cod(\varphi) \cup cod'(\varphi)$*;*
  - *(e)* $d_{n+2}[\ ]_{n+3} \to [yes]_{n+3}$,
    $[yes]_{n+3} \to [\ ]_{n+3} yes$,
    $[yes]_2 \to [\ ]_2 yes$,
    $[yes]_1 \to [\ ]_1 yes$;
  - *(f)* $[d_{n+2}]_2 \to [d_{n+3}]_2[no]_2$,
    $[no]_2 \to [\ ]_2 no$,
    $[no]_1 \to [\ ]_1 no$.

Now we give an example to make easier to follow the computations of the P systems defined above.

Example 2. Let us consider again Example 1 and the formula $\varphi = \{\{x, y, z\}, \{\bar{x}\}, \{\bar{y}\}, \{\bar{z}\}\}$ in it. Let $\Pi(\varphi)$ be the P system constructed in Definition 2 from φ. The initial configuration of $\Pi(\varphi)$ looks as follows:

$$[d_1\ [[xyz, \bar{x}, \bar{y}, \bar{z}]_3]_2]_1.$$

In the first step xyz remains unchanged, $\bar{x}$ is marked with a prime, and from $\bar{y}$ and $\bar{z}$ the symbols, $\bar{y}$, $\bar{y}'$ and $\bar{z}$, $\bar{z}'$ are created, respectively. Then, in the same step, $\Pi(\varphi)$ separates the symbols in membrane 3 into the two new membranes according to that whether they are marked with a prime or not. Thus, after the first step the configuration of the system looks as follows:

$$[d_2\, [[xyz, \bar{y}, \bar{z}]_4, [\bar{x}', \bar{y}', \bar{z}']_4]_2]_1.$$

In the second step, xyz and $\bar{y}'$ remain unchanged, $\bar{y}$ is marked with a prime, and $\bar{x}'$, $\bar{z}$ and $\bar{z}'$ are each rewritten to $\bar{x}\bar{x}'$, $\bar{z}\bar{z}'$, and $\bar{z}\bar{z}'$, respectively, by the corresponding rules in (a3). Then the symbols are separated into the new membranes as follows:

$$[d_3\, [[xyz, \bar{z}]_5, [\bar{y}', \bar{z}']_5, [\bar{x}, \bar{z}]_5, [\bar{x}', \bar{y}', \bar{z}']_5]_2]_1.$$

Finally, after the third step of $\Pi(\varphi)$, its configuration looks as follows:

$$[d_4\, [[xyz]_6, [\bar{z}']_6, [\bar{y}]_6, [\bar{y}', \bar{z}']_6, [\bar{x}]_6, [\bar{x}', \bar{z}']_6, [\bar{x}, \bar{y}]_6, [\bar{x}', \bar{y}', \bar{z}']_6]_2]_1.$$

Now, since every membrane with label 6 contains at least one object, $\Pi(\varphi)$ can continue the computation and send out to the environment the symbol *no* in the same way as $\Pi(3)$ does it.

The correctness of the P system $\Pi(\varphi)$ constructed in Definition 2 from a formula $\varphi \in Form_n$ ($n \in \mathbb{N}$) is based on the following lemma.

Lemma 1. *Let $\varphi \in Form_n$ ($n \in \mathbb{N}$) and $\Pi(n)$, $\Pi(\varphi)$ be the P systems constructed in Definition 1 and Definition 2, respectively. Consider the configurations of $\Pi(\varphi)$ and $\Pi(n)$ started with φ after n steps. Then $\Pi(\varphi)$ has an empty membrane with label $n+3$ if and only if there is an empty membrane of $\Pi(n)$ with the same label.*

Proof. Clearly these P systems have the same membrane structure after every step of the systems. Consider the configurations of them after the ith step for some ($0 \le i \le n$) and let m_1 and m_2 be two corresponding membranes with label $i+3$ in $\Pi(n)$ and $\Pi(\varphi)$, respectively. We show that m_1 and m_2 have the same cardinality which clearly implies the statement of the lemma. It can be seen that, for every object c in m_1, the following holds. There is an object d in the membrane with label 3 of $\Pi(n)$ and there are distinct literals $l_{i_1}, \dots, l_{i_k}$ ($k \le i$) over the variables in X_n not occurring in $\hat{d}$ such that c represents the clause that is yielded by adding the above literals to $\hat{d}$. But then d or d' is in m_2 which means that $|m_1| \le |m_2|$. Using similar arguments, one can show that $|m_2| \le |m_1|$ also holds which concludes the proof of the lemma.

Since we know that the configuration of $\Pi(n)$ (started with φ) after n steps has an empty membrane with label $n+3$ if and only if φ is satisfiable (cf. the discussion after Example 1), we have the following theorem.

Theorem 2. *The SAT can be solved in linear time by a polynomially semi-uniform family $\mathbf{\Pi} : (\Pi(\varphi))_{\varphi \in Form}$ of polarizationless recognizing P systems with the following properties: the elements of $\mathbf{\Pi}$ are deterministic, do not use non-elementary membrane division, and the size of a formula $\varphi \in Form$ is described by the number of variables occurring in the formula.*

4 Conclusions

We proposed a new approach for solving SAT by P systems with active membranes. This approach is based on a method that creates complete clauses from the clauses of a formula in CNF.

We defined a uniform and a semi-uniform family of P systems with active membranes where we implemented the above method. Both systems can decide the satisfiability of a formula in CNF in linear time in the number of variables occurring in the formula. To achieve this efficiency we did not use non-elementary membrane division or polarizations. On the other hand we used separation rules with membrane label changing. The number of computation steps in existing solutions without non-elementary membrane division depends also on the number of clauses in the input formula. However, we cannot say that our results are improvements of the existing ones because of the following reasons. Our uniform solution is not polynomially uniform, while our other solution is not uniform. To improve our results, we are planning to create a polynomially uniform solution based on our method using a formula encoding technique similar to the commonly used one in many existing solutions.

Concerning our existing solutions, it should be mentioned that in Definition 1, the rules in (a) and (c)-(f) have constant size, i.e., they involve a constant number of objects. Moreover, it is not difficult to see that during the evolution of $\Pi(n)$, membranes with label i $(3 \leq i \leq n+3)$ have no more objects than the number m of the clauses in the input formula. Thus the separation rules in (b) always should act on membranes with no more than m objects (similar properties also hold in the case of the P systems defined in Definition 2).

It seems that our solutions may be improved by elaborating and implementing the following observations. First, consider again Example 1 and the P system $\Pi(3)$ with input φ in this example. One can observe that since $\bar{x}$ occurs in a membrane with label 4, every membrane with label 6 that is created from this membrane contains a complete clause. Thus, the system could have dissolved this membrane with label 4, without creating those four membranes with label 6 and changing the output of the system. In general this means that if the P system $\Pi(n)$ created in Definition 1 for some $n \in \mathbb{N}$ has a membrane with label $i+3$ $(1 \leq i \leq n-1)$ containing a clause that do not contain variables $x_{i+1}, \ldots, x_n$, then $\Pi(n)$ could dissolve this membrane without changing the output of the system and saving the creation of $\mathcal{O}(2^{n-i})$ membranes.

It is also clear that if $\Pi(n)$ has an empty membrane with label $i+3$ for some $(1 \leq i \leq n-1)$, then it has an empty membrane with label $n+3$ as well. Thus, the satisfiability of the input formula can turn out earlier than the nth step of the system and this also could save some superfluous membrane creations.

Implementing the above observations we could reduce the number of membranes created during the computation of the system. However, we should note that in general our P systems with the above improvements still would use exponential workspace (in the number of the variables of the input formula).

Since our P system $\Pi(n)$ given in Definition 1 has exponential size in n, it is a reasonable question whether a constant time solution of SAT exists based on $\Pi(n)$. One can see that slightly modifying $\Pi(n)$, a P system $\Pi'(n)$ could be given such that, for a formula $\varphi \in Form_n$, $\Pi'(n)$ can create the complete clauses of φ' only in one step (although in this case some of the rules of $\Pi'(n)$ should introduce an exponential number of objects). On the other hand, it is not clear how could we ensure $\Pi'(n)$ to send out to the environment the correct symbol *yes* or *no* using only constant number of steps.

We are planning to implement our P systems on certain systems using parallel hardware since we would like to see whether our new approach can be utilized in practice as well.

Acknowledgements. The authors are grateful to the reviewer for the many valuable comments and suggestions that improved the manuscript.

References

1. Alhazov, A.: Minimal parallelism and number of membrane polarizations. The Computer Science Journal of Moldova **18(2)**, (2010) 149–170
2. Alhazov, A., Pan, L., Paun, G.: Trading polarizations for labels in P systems with active membranes. Acta Inf. **41(2-3)**, (2004) 111–144
3. Ciobanu, G., Pan, L., Paun, G., Pérez-Jiménez, M.J.: P systems with minimal parallelism. Theor. Comput. Sci. **378(1)**, (2007) 117–130
4. Freund, R., Paun, G., Pérez-Jiménez, M.J.: Polarizationless P Systems with Active Membranes Working in the Minimally Parallel Mode. In: UC. (2007) 62–76
5. Gazdag, Z., Kolonits, G.: A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Vaszil, G. (eds.) Proceedings of the 13th International Conference on Membrane Computing (CMC13), (2012) 211–220
6. Kusper, G.: Solving and Simplifying the Propositional Satisfiability Problem by Sub-Model Propagation. Ph.D. thesis, RISC, Johannes Kepler University, Linz, Austria (2005)
7. Pan, L., Alhazov, A.: Solving HPP and SAT by P Systems with Active Membranes and Separation Rules. Acta Inf. **43(2)**, (2006) 131–145
8. Paun, G.: Computing with membranes. J. Comput. Syst. Sci. **61(1)**, (2000) 108–143
9. Paun, G.: P Systems with Active Membranes: Attacking NP-Complete Problems. Journal of Automata, Languages and Combinatorics **6(1)**, (2001) 75–90
10. Paun, G.: Introduction to membrane computing. In: Applications of Membrane Computing, (2006) 1–42
11. Paun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010), `http://portal.acm.org/citation.cfm?id=1738939`
12. Pérez-Jiménez, M.J., Jiménez, Á.R., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing **2(3)**, (2003) 265–285