Minimal Parallelism and Number of Membrane Polarizations *

Artiom Alhazov[1,2]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinau, MD-2028, Moldova
E-mail: artiom@math.md

[2] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: artiome.alhazov@estudiants.urv.cat

Abstract. It is known that the satisfiability problem (SAT) can be efficiently solved by a uniform family of P systems with active membranes with two polarizations working in a maximally parallel way. We study P systems with active membranes without non-elementary membrane division, working in minimally parallel way. The main question we address is what number of polarizations is sufficient for an efficient computation depending on the types of rules used.
In particular, we show that it is enough to have four polarizations, sequential evolution rules changing polarizations, polarizationless non-elementary membrane division rules and polarizationless rules of sending an object out. The same problem is solved with the standard evolution rules, rules of sending an object out and polarizationless non-elementary membrane division rules, with six polarizations. It is an open question whether these numbers are optimal.

1 Introduction

Membrane computing with symbol-objects is a biologically inspired framework of distributed parallel multiset processing; see [9] for an overwiew and [12] for the comprehensive bibliography. The most addressed questions are completeness (solving every solvable problem) and efficiency (solving hard problems in feasible time). We focus on the latter one.

An interesting class of membrane systems are those with active membranes (see [8]), where membrane division can be used for solving computationally hard problems in polynomial time. Let us mention a few results:

– A *semi–uniform* **solution** to SAT using three polarizations and division for non-elementary membranes, [8].

* The work is partially supported by the project TIC2003-09319-C03-01 from Rovira i Virgili University

- A *polarizationless* solution, [2].
- Using only division for elementary membranes, with three polarizations, [10].
- A *uniform* solution, with elementary membrane division, [11].
- Using only *two polarizations*, in a uniform way, with elementary membrane division, [3].
- Computational **completeness** of P systems with three polarizations and three membranes, [9].
- Using only *two polarizations* and two membranes, [5].
- Using only *one membrane*, with two polarizations, [4].
- *Polarizationless* systems are complete, with no known bound on the number of membranes, [1].
- Solving `SAT` in a **minimally parallel** way, using non-elementary membrane division (replicating both objects and inner membranes), [6].
- Avoiding polarizations by using rules *changing membrane labels*. Using (up to the best author's knowledge) either cooperative rules or non-elementary division as above, [7].

Given a P system, a rule and an object, whether this rule is applicable to this object in some membrane might depend on both membrane label (that usually cannot be changed) and membrane polarization. Essentially, the number of polarizations is the number of states that can be encoded directly on the membrane.

Minimal parallelism provides less synchronization between the objects, so one might expect the need of a stronger control, i.e., more polarizations. It is not difficult to construct the system in such a way that the rules are global (i.e., the membrane labels are not distinguished), most likely without adding additional polarizations. In this way the results dealing with the number of polarizations can be reformulated in terms of number of membrane labels (in that case, the systems have no polarizations, but the rules are allowed to modify membrane labels).

2 Preliminaries

2.1 Solvability by P systems with input

Definition 1. *A* P system with input *is a tuple* (Π, Σ, i_Π)*, where (a)* Π *is a P system with working alphabet , with* m *membranes labelled with* $1, \cdots, m$*, and initial multisets* $w_1, \cdots, w_m$ *(over* $O - \Sigma$*) associated with them; (b)* $\Sigma \subseteq O$ *is an (input) alphabet, (c)* i_Π *is the label of a distinguished (input) membrane.*

The *initial configuration* of (Π, Σ, i_Π) with an input multiset w over Σ is

$$(\mu, w_1, \cdots, w_{i_\Pi} \cup w, \cdots, w_m).$$

We call (Π, Σ, i_Π) a *decisional* P system with input if there exists two distinguished objects $\texttt{yes}, \texttt{no} \in O$ and for any valid input (see *cod* function in the

definition below) all its computations send to the environment exactly one object, either `yes` (in this case the computation is called an *accepting* one) or `no`. Moreover, (Π, Σ, i_Π) is called *confluent* if for any valid input all its computations halt in the same configuration.

Definition 2. *Consider a decision problem $X = (I_X, \theta_X)$: I_X is the set of possible instances of X and θ_X is a boolean function over I_X. We say that X is solvable in polynomial time by a uniform family of P systems $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbb{N}}$ if the following conditions hold:*

- *The family $\mathbf{\Pi}$ is polynomially constructible, i.e., there exists a deterministic Turing machine constructing the system $\Pi(n)$ from n in polynomial time.*
- *There exists a pair (s, cod) of polynomial-time computable functions mapping every instance $u \in I_X$ of the problem X into a natural number and a multiset (over the alphabet of $\Pi(s(u))$), respectively. The instance u is to be solved by a system $\Pi(s(u))$ with the multiset $cod(u)$ placed in the input membrane, as described below.*
- *The family $\mathbf{\Pi}$ is polynomially bounded with respect to (X, cod, s), i.e., there exists a polynomial function $p(n)$ such that for each $u \in I_X$ every computation of the system $\Pi(s(u))$ with input $cod(u)$ is halting in at most $p(s(u))$ steps.*
- *The family $\mathbf{\Pi}$ is sound with respect to (X, cod, s), i.e., for each $u \in I_X$ if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$.*
- *The family $\mathbf{\Pi}$ is complete with respect to (X, cod, s), i.e., for each $u \in I_X$ if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.*

2.2 P systems with active membranes

Definition 3. *A P system* with active membranes *is a P system with the working alphabet O, with the set H of membrane labels, with the set E of polarizations, and with the rules of the following forms:*

(a) $[\ a \to u\ ]_h^e$ *for $a \in O$, $u \in O^*$, $h \in H$ and $e \in E$. These are object evolution rules. An object $a \in O$ in the region associated with a membrane with label h and polarization e evolves to a multiset $u \in O^*$.*

(b) $a[\ \ ]_h^e \to [\ b\ ]_h^{e'}$ *for $a, b \in O$, $h \in H$ and $e, e' \in E$. These are send–in communication rules. An object a from the region immediately outside a membrane with label h and polarization e is introduced in this membrane, transformed into b and changing the polarization of the membrane to e'.*

(c) $[\ a\ ]_h^e \to [\ \ ]_h^{e'} b$ *for $a, b \in O$, $h \in H$ and $e, e' \in E$. These are send–out communication rules. An object a is sent out from the region associated with membrane with label h and polarization e to the region immediately outside, transformed into b and changing the polarization of the membrane to e'.*

(*d*) $[\ a\ ]_h^e \to a$ *for* $a, b \in O$, $h \in H$ *and* $e \in E$. *These are dissolution rules. A membrane with label* h *and polarization* e *is dissolved in reaction with an object* a, *transformed into* b. *The skin is never dissolved.*

(*e*) $[\ a\ ]_h^e \to [\ b\ ]_h^{e'} [\ c\ ]_h^{e''}$ *for* $a, b, c \in O$, $h \in H$ *and* $e, e', e'' \in E$. *These are division rules for elementary membranes. An elementary membrane can be divided into two membranes with the same label, possibly with different polarizations, possibly transforming some objects.*

Generally, rules of type (a) are executed in parallel, while at most one rule out of all rules of types $(b), (c), (d), (e)$ can be applied to the same membrane in the same step. We will also speak about the sequential version

$$(a''_s)\ [\ a\ ]_h^e \to [\ u\ ]_h^{e'} \text{ for } a \in O,\ u \in O^*,\ h \in H \text{ and } e, e' \in E.$$

of rules (a) (let us use $''$ to indicate that the rule is allowed to change the polarization of the membrane) and their modifications $(b_0), (c_0), (d_0), (e_0), (a'_{0s}), (b'_0), (c'_0), (e'_0)$ (here, 0 represents that the rules neither distinguish polarization nor change it, while $'$ means that the rule is allowed to change membrane label).

2.3 Minimal parallelism

These rules are applied according to the following principles:

- The rules of type (a) may be applied in parallel. At one step, a membrane can be the subject of *only one* rule of types $(a'_{0s}), (a''_s)$ and $(b), (c), (d), (e)$ with their modifications.
- In one step, one object of a membrane can be used by only one rule (non-deterministically chosen), but **for every membrane at least one object** that can evolve by one rule of any form, must evolve (no rules associated to a membrane are applied only if none are applicable for the objects that do not evolve).
- If at the same time a membrane is divided by a rule of type (e) and there are objects in this membrane which evolve by means of rules of type (a), then we suppose that first the evolution rules of type (a) are used, and then the division is produced. Of course, this process takes only one step.

3 Using Rules (a''_s)

The three size parameters of the `SAT` problem are the number m of clauses, the number n of variables and the total number l of occurrences of variables in clauses (clearly, $l \leq mn$: without restricting generality, we could assume that no variable appears in the same clause more than once, with or without negation).

Theorem 1. *A uniform family of confluent P systems with rules* $(a''_s), (c_0), (e_0)$ *working in minimally parallel way can solve* `SAT` *with four polarizations in* $O(l(m+n))$ *number of steps.*

Proof. The main idea of the construction is to implement a maximally parallel step sequentially. For this, a "control" object will be changing the polarization, and then an input object or a clause object will be restoring it. Since the input is encoded in l objects, changing and restoring polarization will happen for l times, the counting is done by the "control" object.

Let us consider a propositional formula in the conjunctive normal form:

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

Let us encode the instance of β in the alphabet $\Sigma(\langle n, m, l\rangle)$ by multisets X, X' of the clause-variable pairs such that the variable appears in the clause without negation, with negation or neither:

$$\begin{aligned}
\Sigma(\langle n, m, l\rangle) &= \{v_{j,i,1,s} \mid 1 \le j \le m,\ 1 \le i \le n,\ 1 \le s \le 2\},\\
X &= \{(v_{j,i,1,1}, 1) \mid x_i \in \{y_{j,k} \mid 1 \le k \le l_j\},\\
&\qquad 1 \le j \le m,\ 1 \le i \le n\},\\
X' &= \{(v_{j,i,1,2}, 1) \mid \neg x_i \in \{y_{j,k} \mid 1 \le k \le l_j\},\\
&\qquad 1 \le j \le m,\ 1 \le i \le n\}.
\end{aligned}$$

We construct the following P system:

$$\begin{aligned}
\Pi(\langle n, m, l\rangle) &= (O, H, E, [\,[\ ]_2^0 [\ ]_3^0\,]_1^0, w_1, w_2, w_3, R), \text{ with}\\
O &= \{v_{j,i,k,s} \mid 1 \le j \le m,\ 1 \le i \le n,\\
&\qquad 1 \le k \le m+n+1,\ 1 \le s \le 4\}\\
&\cup \{d_{i,k} \mid 1 \le i \le m+n+1,\ 1 \le k \le 2l\}\\
&\cup \{t_{i,k}, f_{i,k} \mid 1 \le i \le n,\ 1 \le k \le l\}\\
&\cup \{d_i \mid 1 \le i \le m+n+1\} \cup \{S, Z, \texttt{yes}, \texttt{no}\}\\
&\cup \{z_k \mid 1 \le k \le (4l+3)n + m(4l+1) + 2\}\\
w_1 &= \lambda,\ w_2 = d_1,\ w_3 = z_0,\ H = \{1,2,3\},\ E = \{0,1,2,3\},
\end{aligned}$$

and the rules are listed below. The computation consists of three stages.

1. Producing 2^n membranes with label 2, corresponding to the possible assignments of variables $x_1, \cdots, x_n$ and selecting clauses that are satisfied for every assignment (groups A and C of rules).
2. Checking for all assignments whether all clauses are satisfied (groups B and D of rules).
3. Generating `yes` from the positive answer, and sending it to the environment. Generating `no` from the timeout (during the first two stages the number of steps is counted in the object in membrane with label 3) and sending it to the environment if there was no positive answer (groups E and F of rules).

Stage 1 consists of n cycles and stage 2 consists of m cycles. Each cycle's aim is to process all l objects, i.e., each object counts the number of cycles completed, and in the first stage the clauses are evaluated while in the second stage the presence of each clause is checked.

In the case of maximal parallelism, a cycle could be performed in a constant number of (actually, one or two) steps, while the minimal parallelism cannot guarantee that all objects are processed. The solution used here is the following. A cycle consists of marking (setting the last index to 3 or 4) all l objects one by one while performing the necessary operation, and then unmarking (setting the last index to 1 or 2) all of them. Marking or unmarking an object happens in two steps: the control object changes the polarization from 0 to 1, 2 (to mark) or to 3 (to unmark), and then one of the objects that has not yet been (un)marked is processed, resetting the polarization to 0.

Control objects in membrane 2: select clauses

A1 (for variable i: divide)
$[\ d_i\ ] \to [\ t_{i,0}\ ][\ f_{i,0}\ ]$, $1 \le i \le n$

A2 (process and mark all l objects)
$[\ t_{i,k-1}\ ]^0 \to [\ t_{i,k}\ ]^1$, $1 \le i \le n$, $1 \le k \le l$
$[\ f_{i,k-1}\ ]^0 \to [\ f_{i,k}\ ]^2$, $1 \le i \le n$, $1 \le k \le l$

A3 (prepare to unmark objects)
$[\ t_{i,l}\ ]^0 \to [\ d_{i,0}\ ]^0$, $1 \le i \le n$
$[\ f_{i,l}\ ]^0 \to [\ d_{i,0}\ ]^0$, $1 \le i \le n$

A4 (unmark all l objects)
$[\ d_{i,k-1}\ ]^0 \to [\ d_{i,k}\ ]^3$, $1 \le i \le n$, $1 \le k \le l$

A5 (switch to the next variable)
$[\ d_{i,l}\ ]^0 \to [\ d_{i+1}\ ]^0$, $1 \le i \le n$

Control objects in membrane 2: check clauses

B1 (test if clause i is satisfied)
$[\ d_{n+i}\ ]^0 \to [\ d_{n+i,1}\ ]^2$, $1 \le i \le m$

B2 (process and mark the other $l-1$ objects)
$[\ d_{n+i,k-1}\ ]^0 \to [\ d_{n+i,k}\ ]^1$, $1 \le i \le m$, $1 \le k \le l$

B3 (unmark all l objects)
$[\ d_{n+i,l+k-1}\ ]^0 \to [\ d_{n+i,l+k}\ ]^3$, $1 \le i \le m$, $1 \le k \le l$

B4 (switch to the next clause)
$[\ d_{n+i,2l}\ ]^0 \to [\ d_{n+i+1}\ ]^0$, $1 \le i \le m$

B5 (send a positive answer)
$[\ d_{m+n+1}\ ] \to [\ ]S$

Input objects in membrane 2: select clauses

C1 (mark an object)
$[\ v_{j,i,k,s}\ ]^p \to [\ v_{j,i,k+1,s+2}\ ]^0$,
$1 \le i \le m$, $1 \le j \le n$, $1 \le k \le m$, $k \ne m$, $1 \le s \le 2$, $1 \le p \le 2$

C2 (a true variable present without negation or a false variable present with negation satisfies the clause)
$[\ v_{j,i,i,s}\ ]^s \to [\ v_{j,i,i+1,3}\ ]^0$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$

C3 (a true variable present with negation or a false variable present without negation does not satisfy the clause)
$[\ v_{j,i,i,3-s}\ ]^s \to [\ v_{j,i,i+1,4}\ ]^0$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$

C4 (unmark an object)
$[\ v_{j,i,k,s+2}\ ]^3 \to [\ v_{j,i,k,s}\ ]^0$,
$1 \le i \le m$, $1 \le j \le n$, $2 \le k \le m+1$, $1 \le s \le 2$

Input objects in membrane 2: check clauses

D1 (check if the clause is satisfied at least by one variable)
$[\ v_{j,i,m+j,1}\ ]^2 \to [\ v_{j,i,k+1,3}\ ]^0$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$

D2 (mark an object)
$[\ v_{j,i,m+k,s}\ ]^1 \to [\ v_{j,i,k+1,s+2}\ ]^0$,
$1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n$, $1 \le s \le 2$

D3 (unmark an object)
$[\ v_{j,i,m+k,s+2}\ ]^3 \to [\ v_{j,i,k,s}\ ]^0$,
$1 \le i \le m$, $1 \le j \le n$, $2 \le k \le n+1$, $1 \le s \le 2$

Control objects in membrane 3

E1 (count)
$[\ z_{k-1}\ ]^0 \to [\ z_k\ ]^0$, $1 \le k \le N = (4l+3)n + m(4l+1) + 2$

E2 (send time-out object)
$[\ z_N\ ] \to [\ ]Z$

Control objects in the skin membrane

F1 (a positive result generates the answer)
$[\ S\ ]^0 \to [\ \texttt{yes}\ ]^1$

F2 (without the positive answer, the time-out generates the negative answer)
$[\ Z\ ]^0 \to [\ \texttt{no}\ ]^0$

F3 (send the answer)
$[\ \texttt{yes}\ ] \to [\ ]\texttt{yes}$
$[\ \texttt{no}\ ] \to [\ ]\texttt{no}$

Let us now explain how the system works in more details.

Like the input objects, the control objects keep track of the number of cycles completed. The control object also remembers whether marking or unmarking takes place, as well as the number of objects already (un)marked. Moreover, the control object is responsible to pass the "right" information to the objects via polarization: in stage 1, 1 if the variable is true, and 2 if the variable is false; in stage 2, 1 if the clause is already found, and 2 if the clause is being checked for.

During the first stage, an object $v_{j,i,1,s}$ is transformed into $v_{j,i,n+1,t}$, where $t = 1$ if variable x_j satisfies clause C_i, or $t = 2$ if not. The change of the last index from s to t happens when the third index is equal to i. Notice that although only information about what clauses are satisfied seems to be necessary for checking

if β is true for the given assignment of the variables, the information such as the number of cycles completed is kept for synchronization purposes, and the other objects are kept so that their total number remains l. The control object d_1 is transformed into d_{n+1}. Stage 1 takes $(4l+3)n$ steps.

If some clause is not satisfied, then the computation in the corresponding membrane is "stuck" with polarization 2. Otherwise, during the second stage an object $v_{j,i,n+1,t}$ is transformed into $v_{j,i,n+m+1,t}$, while the control object d_{n+1} becomes d_{m+n+1}. Stage 2 takes $m(4l+1)$ steps, plus one extra step to send objects S to skin, if any.

After stage 2 is completed, one copy of S, if any, is transformed into `yes`, changing the polarization of the skin membrane. In the same time `yes`, if it has been produced, is sent out, object Z comes to the skin from region 3. If the polarization of the skin remained 0, Z changes to `no`, which is then sent out. Depending on the answer, stage 3 takes 2 or 4 steps. In either case, the result is sent out in the last step of the computation. □

Notice that membrane labels are not indicated in the rules. This means that the system is organized in such a way that the rules are *global*, i.e., the system would work equally well starting with the configuration $\mu = [\, w_1 [\, w_2 \,]_1^0 [\, w_3 \,]_1^0 \,]_1^0$, the labels were only given for the simplicity of explanation.

Using the remark in the end of the Introduction, we can obtain the following

Corollary 1. *A uniform family of confluent polarizationless P systems with rules* $(a'_{0s}), (c_0), (e_0)$ *working in minimally parallel way can solve* `SAT` *with membrane labels of four kinds.*

The statement follows directly from the possibility of rewriting a global rule $[\, a \,]^e \to [\, u \,]^{e'}$ of type (a''_s) in a rule $[\, a \,]_e \to [\, u \,]_{e'}$ of type (a'_{0s}) (which is polarizationless but is able to change the membrane label).

4 Using Rules (a)

An informal idea of this section is to replace rules of type (a''_s) with rules (a) producing additional objects, and rules (c), sending an additional object out to change the polarization.

Theorem 2. *A uniform family of confluent P systems with rules* $(a), (c), (e_0)$ *working in minimally parallel way can solve* `SAT` *with six polarizations in* $O(l(m+n))$ *number of steps.*

Proof. The strategy used in the construction below is similar to that of the previous theorem. However, since the application of the evolution rules no longer changes the polarization of the membrane, the control symbols $d_{i,k}$, $t_{i,k}$, $f_{i,k}$ no longer "operate" in polarization 0, but rather in polarization that toggles between 0 (for even k) and 5 (for odd k), to prevent multiple applications of evolution rules in a row in the same membrane. Moreover, the input objects are

actually allowed to evolve in parallel (and the degree of parallelism is chosen non-deterministically), but in the end of both halves of a cycle it is possible to count the number of extra objects produced, to make sure that all l objects have been processed.

For the same propositional formula

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

and the same encoding of the instance of β in the alphabet $\Sigma(\langle n, m, l\rangle)$ by multisets X, X',

$$\begin{aligned}
\Sigma(\langle n, m, l\rangle) &= \{v_{j,i,1,s} \mid 1 \leq j \leq m,\ 1 \leq i \leq n,\ 1 \leq s \leq 2\}, \\
X &= \{(v_{j,i,1,1}, 1) \mid x_i \in \{y_{j,k} \mid 1 \leq k \leq l_j\}, \\
&\qquad 1 \leq j \leq m,\ 1 \leq i \leq n\}, \\
X' &= \{(v_{j,i,1,2}, 1) \mid \neg x_i \in \{y_{j,k} \mid 1 \leq k \leq l_j\}, \\
&\qquad 1 \leq j \leq m,\ 1 \leq i \leq n\}.
\end{aligned}$$

we construct the following P system:

$$\begin{aligned}
\Pi(\langle n, m, l\rangle) &= (O, H, E, [\,[\ ]_2^0[\ ]_3^0\,]_1^0, w_1, w_2, w_3, R), \text{ with} \\
O &= \{v_{j,i,k,s} \mid 1 \leq j \leq m,\ 1 \leq i \leq n, \\
&\qquad 1 \leq k \leq m+n+1,\ 1 \leq s \leq 4\} \\
&\cup \{d_{i,k} \mid 1 \leq i \leq m+n+1,\ 1 \leq k \leq 2l\} \\
&\cup \{t_{i,k}, f_{i,k} \mid 1 \leq i \leq n,\ 1 \leq k \leq l\} \\
&\cup \{d_i \mid 1 \leq i \leq m+n+1\} \cup \{S, Z, \texttt{yes}, \texttt{no}\} \\
&\cup \{z_k \mid 1 \leq k \leq (4l+3)n + m(4l+1) + 2\} \\
&\cup \{o_{i,j} \mid 0 \leq i \leq 5,\ 0 \leq j \leq 5\} \\
w_1 &= \lambda,\ w_2 = d_1,\ w_3 = z_0,\ H = \{1,2,3\},\ E = \{0,1,2,3,4,5\},
\end{aligned}$$

and the rules are listed below. The computation stages are the same as in the previous proof.

1. Producing 2^n membranes corresponding to the possible variables assignments; selecting satisfied clauses (groups A and C).
2. Checking whether all clauses are satisfied (groups B and D).
3. Generating the answer and sending it to the environment. (groups E and F).

Stage 1 consists of n cycles and stage 2 consists of m cycles. Each cycle's aim is to process all l objects, i.e., each object counts the number of cycles completed,

and in the first stage the clauses are evaluated while in the second stage the presence of each clause is checked.

A cycle consists of marking (setting the last index to 3 or 4) all l objects one by one while performing the necessary operation, and then unmarking (setting the last index to 1 or 2) all of them. Marking or unmarking an object generally happens in five steps:

1. the control object produces two "polarization changers",
2. one of them changes the polarization from 0 or 5 to 1, 2 (to mark) or to 3 (to unmark),
3. one of the objects that has not yet been (un)marked is processed, producing a "witness" — yet another "polarization changer",
4. the "witness" switches the polarization to 4,
5. the second "changer" produced in step 1 of this routine changes the polarization to 5 or 0.

Notice, however, that "step" 3 might actually take more than one step (more objects can be (un)marked in parallel, or even in a row, creating a supply of "witnesses"). Step 4 might actually be executed in parallel with the last step of "step" 3 (sending out a previous "witness" while producing more). Finally, "step" 3 might even be skipped if a previous "witness" is already there. What matters is that the whole (un)marking routine takes at most $5l$ steps.

Changing polarization of membrane 2

O1 (change from i to j)
$[\ o_{i,j}\ ]^{i} \to [\ \ ]^{j}\ o_{4,5}$, $0 \le i \le 5$, $0 \le j \le 5$

O2 ("witnesses" of D2 are "compatible" with "witnesses" of D1; this does not interfere with the rest of the computation)
$[\ o_{1,4}\ ]^{2} \to [\ \ ]^{4}\ o_{4,5}$

Control objects in membrane 2: select clauses

A1 (for variable i: divide)
$[\ d_i\ ] \to [\ t_{i,0}\ ][\ f_{i,0}\ ]$, $1 \le i \le n$

A2 (process and mark all l objects)
$[\ t_{i,k-1} \to t_{i,k} o_{0,1} o_{4,5}\ ]^{0}$, $1 \le i \le n$, $1 \le k \le l$, k is odd
$[\ f_{i,k-1} \to f_{i,k} o_{0,2} o_{4,5}\ ]^{0}$, $1 \le i \le n$, $1 \le k \le l$, k is odd
$[\ t_{i,k-1} \to t_{i,k} o_{5,1} o_{4,0}\ ]^{5}$, $1 \le i \le n$, $1 \le k \le l$, k is even
$[\ f_{i,k-1} \to f_{i,k} o_{5,2} o_{4,0}\ ]^{5}$, $1 \le i \le n$, $1 \le k \le l$, k is even

A3 (prepare to unmark objects)
$[\ t_{i,l} \to d_{i,0}\ ]^{0}$, $1 \le i \le n$, if l is even
$[\ f_{i,l} \to d_{i,0}\ ]^{0}$, $1 \le i \le n$, if l is even
$[\ t_{i,l} \to d_{i,0} o_{5,0}\ ]^{5}$, $1 \le i \le n$, if l is odd
$[\ f_{i,l} \to d_{i,0} o_{5,0}\ ]^{5}$, $1 \le i \le n$, if l is odd

A4 (unmark all l objects)
$[\ d_{i,k-1} \to d_{i,k} o_{0,3} o_{4,5}\ ]^{0}$, $1 \le i \le n$, $1 \le k \le l$, k is odd
$[\ d_{i,k-1} \to d_{i,k} o_{5,3} o_{4,0}\ ]^{0}$, $1 \le i \le n$, $1 \le k \le l$, k is even

A5 (switch to the next variable)
$[\ d_{i,l} \to d_{i+1}\ ]^0$, $1 \le i \le n$, if l is even
$[\ d_{i,l} \to d_{i+1}o_{5,0}\ ]^5$, $1 \le i \le n$, if l is odd

Control objects in membrane 2: check clauses

B1 (test if clause i is satisfied)
$[\ d_{n+i} \to d_{n+i,1}o_{0,2}o_{4,5}\ ]^0$, $1 \le i \le m$
B2 (process and mark the other $l-1$ objects)
$[\ d_{n+i,k-1} \to d_{n+i,k}o_{0,1}o_{4,5}\ ]^0$, $1 \le i \le m$, $1 \le k \le l$, k is odd
$[\ d_{n+i,k-1} \to d_{n+i,k}o_{5,1}o_{4,0}\ ]^0$, $1 \le i \le m$, $1 \le k \le l$, k is even
B3 (unmark all l objects)
$[\ d_{n+i,l+k-1} \to d_{n+i,l+k}o_{0,3}o_{4,5}\ ]^0$, $1 \le i \le m$, $1 \le k \le l$, $l+k$ is odd
$[\ d_{n+i,l+k-1} \to d_{n+i,l+k}o_{5,3}o_{4,0}\ ]^0$, $1 \le i \le m$, $1 \le k \le l$, $l+k$ is odd
B4 (switch to the next clause)
$[\ d_{n+i,2l} \to d_{n+i+1}\ ]^0$, $1 \le i \le m$
B5 (send a positive answer)
$[\ d_{m+n+1}\ ]^0 \to [\ \ ]^0 S$

Input objects in membrane 2: select clauses

C1 (mark an object)
$[\ v_{j,i,k,s} \to v_{j,i,k+1,s+2}o_{p,4}\ ]^p$,
$1 \le i \le m$, $1 \le j \le n$, $1 \le k \le m$, $k \ne m$, $1 \le s \le 2$, $1 \le p \le 2$
C2 (a true variable present without negation or a false variable present with negation satisfies the clause)
$[\ v_{j,i,i,s} \to v_{j,i,i+1,3}o_{s,4}\ ]^s$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$
C3 (a true variable present with negation or a false variable present without negation does not satisfy the clause)
$[\ v_{j,i,i,3-s} \to v_{j,i,i+1,4}o_{s,4}\ ]^s$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$
C4 (unmark an object)
$[\ v_{j,i,k,s+2} \to v_{j,i,k,s}o_{3,4}\ ]^3$,
$1 \le i \le m$, $1 \le j \le n$, $2 \le k \le m+1$, $1 \le s \le 2$

Input objects in membrane 2: check clauses

D1 (check if the clause is satisfied at least by one variable)
$[\ v_{j,i,m+j,1} \to v_{j,i,k+1,3}o_{1,4}\ ]^2$, $1 \le i \le m$, $1 \le j \le n$, $1 \le s \le 2$
D2 (mark an object)
$[\ v_{j,i,m+k,s} \to v_{j,i,k+1,s+2}o_{1,4}\ ]^1$,
$1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n$, $1 \le s \le 2$
D3 (unmark an object)
$[\ v_{j,i,m+k,s+2} \to v_{j,i,k,s}o_{3,4}\ ]^3$,
$1 \le i \le m$, $1 \le j \le n$, $2 \le k \le n+1$, $1 \le s \le 2$

Control objects in membrane 3

E1 (count)
$[\ z_{k-1} \to z_k\ ]^0$, $1 \le k \le N = (10l+5)n + m(10l+1) + 2$

E2 (send time-out object)
$[\ z_N\ ]^0 \to [\ \ ]^0 Z$

Control objects in the skin membrane

F1 (the first positive result sends the answer)
$[\ S\ ]^0 \to [\ \ ]^1 \texttt{yes}$

F2 (without the positive result, the time-out sends the negative answer)
$[\ Z\ ]^0 \to [\ \ ]^0 \texttt{no}$

Let us now explain how the system works in more details. The control objects keep track of the number of cycles completed, whether marking or unmarking takes place, as well as the number of objects already (un)marked. Moreover, the control object is responsible to pass the "right" information to the objects via polarization: in stage 1, by generating $o_{0,1}$ or $o_{5,1}$ if the variable is true, and $o_{0,2}$ or $o_{5,2}$ if the variable is false; in stage 2, $o_{0,1}$ or $o_{5,1}$ if the clause is already found, and $o_{0,2}$ or $o_{5,2}$ if the clause is being checked for.

During the first stage, an object $v_{j,i,1,s}$ is transformed into $v_{j,i,n+1,t}$, where $t = 1$ if variable x_j satisfies clause C_i, or $t = 2$ if not. The change of the last index from s to t happens when the third index is equal to i. The control object d_1 is transformed into d_{n+1}. Stage 1 takes at most $(10l + 5)n$ steps (at most $(10l + 3)n$ in the case when l is even).

If some clause is not satisfied, then the computation in the corresponding membrane is "stuck" with polarization 2. Otherwise, during the second stage an object $v_{j,i,n+1,t}$ is transformed into $v_{j,i,n+m+1,t}$, while the control object d_{n+1} becomes d_{m+n+1}. Stage 2 takes at most $m(10l + 1)$ steps, plus one extra step to send objects S to skin, if any.

After stage 2 is completed, one copy of S, if any, is sent out as `yes`, changing the polarization of the skin membrane. After this time has passed, object Z comes to the skin from region 3. If the polarization of the skin remained 0, Z is sent out as `no`. □

The rules of the system in the proof above are also *global*, so we can again obtain the following

Corollary 2. *A uniform family of confluent polarizationless P systems with rules* $(a), (c'_0), (e_0)$ *working in minimally parallel way can solve* `SAT` *with membrane labels of six kinds.*

5 Conclusions

Since changing membrane polarization controls what rules can be applied, the number of polarizations corresponds to the number of states of this control. Moreover, almost the only way the objects of the system may interact is via changing membrane polarization. Therefore, the number of polarizations is a complexity measure that deserves our attention.

For maximal parallelism it has been proved that two polarizations are sufficient for both universality (with one membrane) and efficiency, while one-polarization systems are still universal (with elementary membrane division and membrane dissolution), but are conjectured not to be efficient.

We proved that efficient solutions of computationally hard problems by P systems with active membranes working in minimally parallel way can be constructed avoiding both cooperative rules and non-elementary membrane division, thus improving results from [6],[7]. For this task, it is enough to have four polarizations, sequential evolution rules changing polarizations, polarizationless elementary membrane division rules and polarizationless rules of sending an object out. One can use the standard evolution and send-out rules, as well as polarizationless elementary membrane division rules; in this case, six polarizations suffice.

The first construction is "almost" deterministic: the only choices the system can make in each cycle is the order in which the input systems are processed. The second construction exhibits a more asynchronous behaviour of the input objects, which, depending on the chosen degree of parallelism, might speed up obtaining the positive answer, but less than by 20% [1]. In this case, controlling polarizations by evolution is still faster than controlling polarizations by communication.

A number of interesting problems related to minimal parallelism remain open. For instance, is it possible to decrease the number of polarizations/labels? Moreover, it presents an interest to study other computational problems in the minimally-parallel setting, for instance, the computational power of P systems with one active membrane working in the minimally parallel way.

References

1. A. Alhazov, P Systems without Multiplicities of Symbol-Objects, *Information Processing Letters*, accepted, 2005.
2. A. Alhazov, L. Pan, Gh. Păun, Trading Polarizations for Labels in P Systems with Active Membranes, *Acta Informaticae* **41**, 2-3, 2004, 111–144.
3. A. Alhazov, R. Freund, On the Efficiency of P Systems with Active Membranes and Two Polarizations, *Membrane Computing*, International Workshop, WMC 2004, Milan, 2004, Revised Selected and Invited Papers (G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, Eds.), *Lecture Notes in Computer Science* **3365**, Springer, 2005, 146–160, and *Fifth Workshop on Membrane Computing (WMC5)* (G. Mauri, Gh. Păun, C. Zandron, Eds.), University of Milano-Bicocca, Milan, 2004, 81–94.
4. A. Alhazov, R. Freund, A. Riscos-Núñez, One and Two Polarizations, Membrane Creation and Objects Complexity in P Systems, *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (SYNASC'05),

[1] The maximal total number of steps needed is slightly over $10l(m+n)$; the fastest computation happens if rules C2 are executed in parallel for all input objects, as well as rules C4, D2, D3, saving $lm-1$, $lm-1$, $ln-1$, $ln-1$ steps, respectively. Their total is $2l(m+n)-4$, which is less than (but assymptotically equal to) 1/5 of the worst time.

IEEE Computer Society, 2005, 385–394, and First International Workshop on *Theory and Application of P Systems*, Timişoara, Romania, 2005 (G. Ciobanu, Gh. Păun, Eds.), 2005, 9–18.

5. A. Alhazov, R. Freund, Gh. Păun, Computational Completeness of P Systems with Active Membranes and Two Polarizations, *Machines, Computations, and Universality*, International Conference, MCU 2004, Saint Petersburg, 2004, Revised Selected Papers (M. Margenstern, Ed.), *Lecture Notes in Computer Science* **3354**, Springer, 2005, 82–92.
6. G. Ciobanu, Gh. Păun, L. Pan, M.J. Pérez-Jiménez, P Systems with Minimal Parallelism, submitted, 2005.
7. T.O. Ishdorj, Power and Efficiency of Minimal Parallelism in Polarizationless P Systems, *J. Automata, Languages, and Combinatorics*, to appear.
8. Gh. Păun, P Systems with Active Membranes: Attacking **NP**–Complete Problems, *Journal of Automata, Languages and Combinatorics* **6**, 1, 2001, 75–90.
9. Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin (Natural Computing Series), 2002.
10. Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the Power of Membrane Division in P systems, *Theoretical Computer Science* **324**, 1, 2004, 61–85.
11. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity Classes in Cellular Computing with Membranes, *Natural Computing* **2**, 3, 2003, 265–285.
12. P Systems Webpage, `http://psystems.disco.unimib.it`