# P Systems with Active Membranes and Separation Rules [1]

**Linqiang Pan**
(Huazhong University of Science and Technology, Hubei, China, and
Rovira i Virgili University, Tarragona, Spain
lqpan@mail.hust.edu.cn)

**Tseren-Onolt Ishdorj**
(Rovira i Virgili University, Tarragona, Spain
tserenonolt.ishdorj@estudiants.urv.es)

**Abstract:** The P systems are a class of distributed parallel computing devices of a biochemical type. In this paper, a new definition of separation rules in P systems with active membranes is given. Under the new definition, the efficiency and universality of P systems with active membranes and separation rules instead of division rules are investigated.

**Key Words:** Natural computing, Membrane computing, Distributed computing

**Category:** C.1.4, F.1.1

## 1 Introduction

The P systems are a class of distributed parallel computing devices of a biochemical type, introduced in [Păun 2000], which can be seen as a general computing architecture where various types of objects can be processed by various operations. The area starts from the observation that certain processes which take place in the complex structure of living cells can be considered as computations. For details we refer to [Păun 2000], [Păun 2002].

Informally speaking, in P systems with active membranes one uses six types of rules: (*a*) multiset rewriting rules, (*b*) rules for introducing objects into membranes, (*c*) rules for sending objects out of membranes, (*d*) rules for dissolving membranes, (*e*) rules for dividing elementary membranes, and (*f*) rules for dividing non-elementary membranes.

Membrane division − inspired from cell division well-known in biology − is the most investigated way for obtaining an exponential working space in a linear time, and solving on this basis hard problems, typically **NP**-complete problems, in polynomial (often, linear) time. Details can be found in [Păun 2001, Păun 2002, Pérez-Jiménez et al. 2002]. Recently, also **PSPACE**-complete problems were attacked in this way (see [Sosík 2003, Alhazov et al. 2003]).

---

[1] Dedicated to Professor Zhang Kemin for his 70th birthday

Separation is also a well known phenomenon of cell biology. Many macro-molecules are too large to be transported through membranes by means of protein channels, and then they are transported through vesicle formation. This process can transport packages of chemicals out of the cell.

By division, the two new membranes have exactly the same objects except for at most a pair of different objects; it is also possible that the two new membranes have different charges. However, in the biological phenomenon of separation, the contents of the two new membranes evolved from a membrane can be significantly different. In [Alhazov and Ishdorj 2004], separation rules are introduced to P systems with active membranes, where for each separation rule, a different subset $U$ of objects is considered to activate the membrane to separate. In this paper, we give a new definition of separation rules. In the new definition, for each separation rule one object is used to activate separation; for all separation rules a uniform subset $O_1$ of objects is used to denote which membrane objects should go. From this point of view, the new definition is an improvement of the definition in [Alhazov and Ishdorj 2004]. Under the new definition, the efficiency and universality of P systems with active membranes and separation rules instead of division are investigated.

## 2   P Systems with Active Membranes

We assume the reader to be familiar with basic elements of complexity theory and formal language theory, for instance, from [Papadimitriou 1994, Salomaa 1973, Rozenberg and Salomaa 1997], as well as with the basic knowledge of membrane computing, for instance, from [Păun 2002] (details and recent results from membrane computing can be found at the web address `http://psystems.disco.uni mib.it`). We only mention that $RE$ denote the family of recursively enumerable languages, and that for a family of languages $FL$, by $PsFL$ we denote the family of Parikh sets of languages in $FL$; as usual, the Parikh mapping associated with an alphabet $V$ is denoted by $\Psi_V$.

A P system *with active membranes* (and electrical charges) is a construct

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

where:

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

2. $O$ is the alphabet of *objects*, where $O = O_1 \cup O_2$, $O_1, O_2 \neq \emptyset$, $O_1 \cap O_2 = \emptyset$;

3. $H$ is a finite set of *labels* for membranes;

4. $\mu$ is a *membrane structure*, consisting of $m$ membranes, labeled (not necessarily in a one-to-one manner) with elements of $H$;

5. $w_1, \ldots, w_m$ are strings over $O$, describing the *multisets of objects* placed in the $m$ regions of $\mu$;

6. $R$ is a finite set of *developmental rules*, of the following forms:

   (a) $[\, a \rightarrow v]^e_h$,
   for $h \in H, e \in \{+, -, 0\}, a \in O, v \in O^*$
   (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) $a[\ ]^{e_1}_h \rightarrow [\, b]^{e_2}_h$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (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);

   (c) $[\, a\,]^{e_1}_h \rightarrow [\ ]^{e_2}_h b$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (communication rules; an object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);

   (d) $[\, a\,]^e_h \rightarrow b$,
   for $h \in H, e \in \{+, -, 0\}, a, b \in O$
   (dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

   (e) $[\, a\,]^{e_1}_h \rightarrow [\, b\,]^{e_2}_h [\, c\,]^{e_3}_h$,
   for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$
   (division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects);

   (f) $[\, [\ ]^{\alpha_1}_{h_1} \cdots [\ ]^{\alpha_1}_{h_k} [\ ]^{\alpha_2}_{h_{k+1}} \cdots [\ ]^{\alpha_2}_{h_n}]^{\alpha_0}_{h_0}$
   $\rightarrow [\, [\ ]^{\alpha_3}_{h_1} \cdots [\ ]^{\alpha_3}_{h_k}]^{\alpha_5}_{h_0} [\, [\ ]^{\alpha_4}_{h_{k+1}} \cdots [\ ]^{\alpha_4}_{h_n}]^{\alpha_6}_{h_0}$,
   for $k \geq 1, n > k, h_i \in H, 0 \leq i \leq n$, and $\alpha_0, \ldots, \alpha_6 \in \{+, -, 0\}$ with $\{\alpha_1, \alpha_2\} = \{+, -\}$; if the membrane with the label $h_0$ contains other membranes than those with the labels $h_1, \ldots, h_n$ specified above, then they must have neutral charges in order to make this rule applicable; these membranes are duplicated and then are part of the contents of both new copies of the membrane $h_0$

(division of non-elementary membranes; this is possible only if a membrane contains two immediately lower membranes of opposite polarization, $+$ and $-$; the membranes of opposite polarizations are separated in the two new membranes, but their polarization can change; always, all membranes of opposite polarizations are separated by applying this rule).

(Note that, in order to simplify the writing, in contrast to the style customary in the literature, we have omitted the label of the left parenthesis from a pair of parentheses which identifies a membrane.) The rules of type $(a)$ are applied in the parallel way (all objects which can evolve by such a rule should do it), while the rules of types $(b), (c), (d), (e), (f)$ are used sequentially, in the sense that one membrane can be used by at most one rule of these types at a time. In total, the rules are used in the non-deterministic maximally parallel manner: all objects and all membranes which can evolve, should evolve. Only halting computations give a result, non-halting computations give no output.

The separation rules introduced in [Alhazov and Ishdorj 2004] are of the following form (without polarizations):

$(h_0)$ $[\,O\,]_h \to [\,U\,]_h[\,O - U\,]_h$, for $h \in\ H, U \subset O$.

If polarization is considered, they will be of the form:

$(h)$ $[\,O\,]_h^{e_1} \to [\,U\,]_h^{e_2}[\,O - U\,]_h^{e_3}$, for $h \in\ H, U \subset O$.

Here, we introduce a new definition of separation rules.

$(g)$ $[\,a\,]_h^{e_1} \to [\,O_1\,]_h^{e_2}[\,O_2\,]_h^{e_3}$,
for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a \in O$
(separation rules for elementary membranes; in reaction with an object, the membrane is separated into two membranes with the same label, possibly different polarization; at the same time, the object $a$ can evolve; the objects from $O_1$ are placed in the first membrane, those from $O_2$ are placed in the second membrane).

As the rules of types $(b), (c), (d), (e)$, and $(f)$, the rules of type $(g)$ are used sequentially, in the sense that one membrane can be used by at most one rule of these types at a time. If at the same time a membrane $h$ is separated, and there are objects in this membrane which evolve by means of rules of type $(a)$, then in the two new membranes we introduce the result of evolution; that is, we suppose that first the evolution rules of type $(a)$ are used, changing the objects, then separation is produced. Of course, this process takes one step.

Note the difference between division rules and separation rules. In a division rule, besides the fact that the object specified in the rule is replaced by two possibly different new objects, the two new membranes have the same copies of

objects. In contrast, in a separation rule, the new objects are placed into the two new membranes according to $O_1$ and $O_2$; these two new membranes can be much different, for instance, in the number of objects.

For the difference between the definitions of $(g)$ and $(h)$, see the introduction.

To understand what it means solving a problem in a uniform and confluent way, here we briefly recall some related notions. Given a decision problem $X$, we say that it can be solved in polynomial (linear) time by recognizing P systems in a uniform way, if, informally speaking, we can construct in polynomial time a family of recognizing P systems $\Pi_n$, $n \in \mathbb{N}$, associated with the sizes $n$ of instances $X(n)$ of the problem, such that the system will always stop in a polynomial (linear, respectively) number of steps, sending out the object yes if the instance $X(n)$ has a positive answer and the object no if the instance $X(n)$ has a negative answer. If the computation of a P system is nondeterministic, but all branches of a computation eventually reach a unique configuration, then we say that the system is confluent.

In the next section, we will show that P systems with separation rules instead of division rules can solve the SAT problem in linear time in a uniform and deterministic way.

## 3    Solving SAT by P Systems with Separation Rules

**Theorem 1.** *P systems with rules of types* $(a), (b), (c),$ *and* $(g)$ *can solve* SAT *in linear time in a uniform and deterministic way.*

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

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

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

$$V(\langle n, m \rangle) = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\}.$$

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

$$w = \{x_{i,j} \mid x_j \in \{y_{i,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq m, 1 \leq j \leq n\}$$
$$\cup \{\bar{x}_{i,j} \mid \neg x_j \in \{y_{i,k} \mid 1 \leq k \leq l_i\}, 1 \leq i \leq m, 1 \leq j \leq n\}.$$

For given $(n, m) \in \mathbb{N}^2$, we construct a recognizing P system $(\Pi(\langle n, m \rangle), V(\langle n, m \rangle), 2)$ with:

$$\Pi(\langle n, m \rangle) = (O(\langle n, m \rangle), H, \mu, w_1, w_2, R),$$

$$O(\langle n, m \rangle) = O_1 \cup O_2,$$
$$O_1 = \{x_{i,j}, \bar{x}_{i,j}, \mid 0 \le i \le m, 1 \le j \le n\} \cup \{c_i \mid 1 \le i \le m+2\}$$
$$\cup \{d_i \mid 1 \le i \le 2n+2m+2\} \cup \{r_{i,j} \mid 0 \le i \le m, 1 \le j \le n\}$$
$$\cup \{e, t, \lambda, \texttt{yes}, \texttt{no}\},$$
$$O_2 = \{x'_{i,j}, \bar{x}'_{i,j}, \mid 0 \le i \le m, 1 \le j \le n\} \cup \{d'_i \mid 1 \le i \le n\}$$
$$\cup \{r_{i,j} \mid 0 \le i \le m, 1 \le j \le n\} \cup \{e'\},$$
$$\mu = [\, [\, ]_2 ]_1,$$
$$w_1 = \lambda, \ w_2 = d_1,$$
$$H = \{1, 2\},$$

and the following rules (we also give explanations about the use of these rules):

1. $[\, d_i ]_2^0 \to [\, O_1 ]_2^+ [\, O_2 ]_2^-$, $1 \le i \le n$.
   In membrane with label 2, when it is "electrically neutral", object $d_i$ causes the membrane to separate and to choose for a variable $x_i$, $1 \le i \le n$, both values *true* and *false*, in form of charges $+$ and $-$ of the two created membranes with the label 2. These rules allow us to have $2^n$ internal membranes, in $n$ steps.

2. $[\, x_{i,j} \to x_{i,j} x'_{i,j} ]_2^0$, $1 \le i \le m$, $1 \le j \le n$.
   $[\, \bar{x}_{i,j} \to \bar{x}_{i,j} \bar{x}'_{i,j} ]_2^0$, $1 \le i \le m$, $1 \le j \le n$.
   $[\, d_i \to d_{i+1} e d'_{i+1} e' ]_2^0$, $1 \le i \le n-1$.
   At the same time with using the rule of type (1), the objects evolve by rules of type (2).

3. $[\, x_{i,1} \to r_{i,1} ]_2^+$, $1 \le i \le m$.
   $[\, \bar{x}_{i,1} \to \lambda ]_2^+$, $1 \le i \le m$.
   $[\, x'_{i,1} \to \lambda ]_2^-$, $1 \le i \le m$.
   $[\, \bar{x}'_{i,1} \to r'_{i,1} ]_2^-$, $1 \le i \le m$.
   The rules of type (3) try to implement a process allowing the internal membranes to encode the assignment of a variable and, simultaneously, to check the value of all clauses by this assignment, in such a way that, if the clause is true, then an object $r_{i,1}$ or $r'_{i,1}$ will appear in the membrane. In the other case, the object encoding the variable will disappear.

4. $[\, x_{i,j} \to x_{i,j-1} ]_2^+$, $1 \le i \le m$, $2 \le j \le n$.
   $[\, \bar{x}_{i,j} \to \bar{x}_{i,j-1} ]_2^+$, $1 \le i \le m$, $2 \le j \le n$.
   $[\, x'_{i,j} \to x_{i,j-1} ]_2^-$, $1 \le i \le m$, $2 \le j \le n$.
   $[\, \bar{x}'_{i,j} \to \bar{x}_{i,j-1} ]_2^-$, $1 \le i \le m$, $2 \le j \le n$.
   The check process described in the rules of type (3) is always made with respect to the first variable appearing in the internal membrane. Hence, the

rules of type (4) take charge of making a cyclic path through all the variables to get that, initially, the first variable is $x_1$, then $x_2$, and so on.

5. $[\, e\,]_1^+ \to [\ ]_1^0 e.$
   $[\, e'\,]_1^- \to [\ ]_1^0 e.$
   $[\, d_i' \to d_i\,]_2^-,\, 1 \le i \le n.$
   The auxiliary objects $e$ and $e'$ exit the membrane changing the polarizations to neutral and the object $d_i'$ in the membrane with negative charge evolves to the object $d_i$ (for the use of the rules of type (1) and the third rules of type (2)), so that the above described generating process of the assignments and the encoding of the satisfied clauses can cycle.

6. $[\, r_{i,k} \to r_{i,k+1} r_{i,k+1}'\,]_2^0$, for $1 \le i \le m,\, 1 \le k \le n-1.$
   $[\, r_{i,k}' \to r_{i,k+1} r_{i,k+1}'\,]_2^0$, for $1 \le i \le m,\, 1 \le k \le n-1.$
   These rules are designed to denote the fact that if clause $C_i$ is satisfied by the assignment encoded by a membrane, then the new membranes obtained from it by separation also satisfy the clause $C_i$. The second subscript of $r_{i,k}$ or $r_{i,k}'$ is used for synchronization, which is necessary, because of the use of objects $r_{i,n}$ and $r_{i,n}'$ in the rules of types (11), (12), and (13).

7. $[\, d_i \to d_{i+1}\,]_2^0$, for $n \le i \le 2n-2.$
   $[\, d_{2n-2} \to d_{2n-1} c_1\,]_2^0.$
   Through the counter objects $d_i$, the rules of type (7) control the process of synchronization of the objects $r_{i,k}$ and $r_{i,k}'$ in the internal membranes.

8. $[\, d_{2n-1}\,]_2^0 \to [\ ]_2^+ d_{2n-1}.$
   The application of the rules of type (8) will show that the system is ready to check which clauses are true by the assignment encoded by an internal membrane.

9. $[\, d_i \to d_{i+1}\,]_1^0,\, 2n-1 \le i \le 2n+2m+1.$
   The rules of type (9) supply counter objects $d_i$ in the skin, in such a way that, if objects $d_{2n+2m-1}$ appear, then they show the end of the checking of the clauses. The objects $d_i$, with $2n+2m \le i \le 2n+2m+2$, will control the final stage of the computation.

10. $[\, r_{1,n}\,]_2^+ \to [\ ]_2^- r_{1,n}.$
    $[\, r_{1,n}'\,]_2^+ \to [\ ]_2^- r_{1,n}.$
    For all $2^n$ internal membranes, we check whether $r_{1,n}$ or $r_{1,n}'$ is present in each membrane. If this is the case, then $r_{1,n}$ or $r_{1,n}'$ is sent out of the membrane where it is present (one copy of $r_{1,n}$ or $r_{1,n}'$ exits the membrane where it is present, the other copies will evolve to $r_{0,n}$ or $r_{0,n}'$ by the rules of type (10), which will never evolve again), changing in this way the polarization of that membrane, to negative. The membranes which do not contain the object $r_{1,n}$

or $r'_{1,n}$ remain positive and they will no longer evolve, as no further rule can be applied to them.

11. $[\, r_{i,n} \to r_{i-1,n}]_2^-$, for $1 \leq i \leq n$.
    $[\, r'_{i,n} \to r'_{i-1,n}]_2^-$, for $1 \leq i \leq n$.
    The rules of type (10) always work with respect to the objects $r_{1,n}$ or $r'_{1,n}$,
    The relabel in the rules of type (11) takes charge of making a cyclic path through all $r_{i,n}$ and $r'_{i,n}$, and evolves the objects $r_{1,n}$ or $r'_{1,n}$ to $r_{0,n}$ or $r'_{0,n}$ (which will never evolve again).

12. $r_{1,n}[\,]_2^- \to [\, r_{0,n}]_2^+$.
    The objects $r_{1,n}$ from the skin membrane return to internal membranes with negative charge, changing to $r_{0,n}$, and returning the polarization of the membrane to positive. This makes possible the use of rules of type (10).

    Note that in the skin membrane the number of copies of $r_{1,n}$ is equal to the number of membranes with negative charge; thus, because of parallelism, each membrane which previously contained objects $r_{1,n}$ or $r'_{1,n}$ will now contain an object $r_{0,n}$.

13. $[\, c_i \to c_{i+1}]_2^-$, $1 \leq i \leq m$.
    The presence of object $c_i$ (with $2 \leq i \leq m + 1$) in the internal membrane shows that the assignment makes the first $i - 1$ clauses true.

14. $[\, c_{m+1}]_2^+ \to [\,]_2^+ c_{m+1}$.
    The presence of the object $c_{m+1}$ shows that all clauses are satisfied by the assignment encoded by an internal membrane. The rule of type (14) sends to the skin the objects $c_{m+1}$ appearing in the internal membranes.

15. $[\, c_{m+1} \to c_{m+2} t]_1^0$.
    The objects $c_{m+1}$ in the skin evolve to objects $c_{m+2} t$. The objects $t$ in the skin are produced simultaneously with the appearance of the objects $d_{2n+2m+1}$ in the skin, and they will be used to output the computing result.

16. $[\, t]_1^0 \to [\,]_1^+ t$.
    The rule of type (16) sends out of the system an object $t$ changing the polarization of the skin to positive, then objects $t$ remaining in the skin are not able to evolve. Then by rule of type (17), the object $c_{m+2}$ can exit the skin producing an object yes, telling us that the formula is satisfiable, and the computation halts.

17. $[\, c_{m+2}]_1^+ \to [\,]_1^-$ yes.
    The application of the rule of type (17) changes the polarization in the skin membrane to negative in order that the objects $c_{m+2}$ remaining in it are not able to continue evolving.

18. $[\ d_{2n+2m+2}\ ]_1^0 \rightarrow [\ ]_1^+\ \texttt{no}$.

By the rule (18) the object $d_{2n+2m+2}$ only evolves when the skin has neutral charge (this is the case when the formula is not satisfiable). Then the system will evolve sending out to the environment an object $\texttt{no}$ and changing the polarization of the skin to positive, in order that objects $d_{2n+2m+2}$ remaining in the skin, do not evolve.

From the previous explanation of the use of rules, one can easily see how this P system works. It is clear that the object $\texttt{yes}$ is sent to the environment if and only if the formula $\beta$ is satisfiable. This is achieved in $3n + 2m + 4$ steps: in $2n$ steps we create $2^n$ internal membranes (as well as the $2^n$ different truth-assignments), then $n$ steps for synchronization; it takes $2m$ steps to check whether all clauses are satisfied by an assignment; further 4 steps are necessary to output the computing result $\texttt{yes}$. If formula $\beta$ is not satisfiable, then at step $3n + 2m + 4$ the system sends the object $\texttt{no}$ 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, 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 [Păun 2002] (although P systems in [Păun 2002] are semi-uniform). So $\texttt{SAT}$ problem was decided in linear time $(3n + 2m + 2)$ by recognizing active P systems with separation rule in a uniform way, and this concludes the proof.   $\square$

## 4    Removing Polarizations

Following the idea in [Alhazov and Pan 2004, Alhazov et al. 2003], let us consider now rules of types $(a) - (e)$ and $(g)$ without polarizations. They are of the following forms (because "no polarization" means "neutral polarization", we add the subscript 0 to the previous letters identifying the six types of rules; as above, $O = O_1 \cup O_2$ is the alphabet of objects and $H$ is the set of labels of membranes):

$(a_0)$  $[\ a \rightarrow v]_h$, where $a \in O, v \in O^*$, and $h \in H$,

$(b_0)$  $a[\ ]_h \rightarrow [\ b]_h$, where $a, b \in O$ and $h \in H$,

$(c_0)$  $[\ a]_h \rightarrow [\ ]_h b$, where $a, b \in O$ and $h \in H$,

$(d_0)$  $[\ a]_h \rightarrow b$, where $a, b \in O$ and $h \in H$,

$(e_0)$  $[\ a]_h \rightarrow [\ b]_h[\ c]_h$, where $a, b, c \in O$ and $h \in H$,

$(g_0)$  $[\ a]_h \rightarrow [\ O_1]_h[\ O_2]_h$, for $h \in H, a \in O$.

Rules of types $(b), (c), (e), (g)$ in Section 2 were introduced without the capability of changing the label of membranes they involve (this makes no sense for dissolving rules), but in [Păun 2002] one already considers rules of type $(e)$ which can change both the label and the polarization of membranes. Such rules are of the form

$$[\, a\,]_{h_1}^{e_1} \rightarrow [\, b\,]_{h_2}^{e_2} [\, c\,]_{h_3}^{e_3}, \text{ with } a, b, c \in O, \ e_1, e_2, e_3 \in \{+, -, 0\}, \text{ and } h_1, h_2, h_3 \in H,$$

and they have been called of type $(e')$. We extend this idea and this notation to rules of types $(b_0), (c_0), (e_0), (g_0)$: their primed versions indicate the fact that the labels can be changed. Specifically, these rules are of the following forms:

$(b'_0) \quad a[\ ]_{h_1} \rightarrow [\, b\,]_{h_2}$, where $a, b \in O$ and $h_1, h_2 \in H$,

$(c'_0) \quad [\, a\,]_{h_1} \rightarrow [\ ]_{h_2} b$, where $a, b \in O$ and $h_1, h_2 \in H$,

$(e'_0) \quad [\, a\,]_{h_1} \rightarrow [\, b\,]_{h_2} [\, c\,]_{h_3}$, where $a, b, c \in O$ and $h_1, h_2, h_3 \in H$,

$(g'_0) \quad [\, a\,]_{h_1} \rightarrow [\, O_1\,]_{h_2} [\, O_2\,]_{h_3}$, for $h_1, h_2, h_3 \in H, a \in O$.

In the following, we consider the efficiency and universality of active P systems without polarization with separation rules instead of division.

## 4.1 Efficiency

**Theorem 2.** *P systems with rules of types $(a_0), (b_0), (c_0), (g'_0)$ can solve* SAT *in linear time in a uniform and confluent way.*

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

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

As above, the instance $\beta$ (to which the size $(m, n)$ is associated) is encoded as a multiset over

$$V(\langle n, m \rangle) = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \le i \le m, 1 \le j \le n\}.$$

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

$$w = \{x_{i,j} \mid x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m, 1 \le j \le n\}$$
$$\cup \{\bar{x}_{i,j} \mid \neg x_j \in \{y_{i,k} \mid 1 \le k \le l_i\}, 1 \le i \le m, 1 \le j \le n\}.$$

For given $(n, m) \in \mathbb{N}^2$, we construct a recognizing P system $(\Pi(\langle n, m \rangle), V(\langle n, m \rangle), 2)$ with:

$$\Pi(\langle n, m \rangle) = (O(\langle n, m \rangle), H, \mu, w_1, w_2, w_7, R),$$
$$O(\langle n, m \rangle) = O_1 \cup O_2,$$
$$O_1 = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \le i \le m, 0 \le j \le n\} \cup \{d_i \mid 0 \le i \le 2n + 2m + 6\}$$
$$\cup \{c_i \mid 1 \le i \le m\} \cup \{\lambda, e, f_0, f_1, \texttt{yes}, \texttt{no}\},$$
$$O_2 = \{x'_{i,j}, \bar{x}'_{i,j} \mid 1 \le i \le m, 0 \le j \le n\} \cup \{d'_i \mid 0 \le i \le n - 1\}$$
$$\cup \{c'_i \mid 1 \le i \le m\},$$
$$\mu = [\,[\,]_2[\,]_7\,]_1,$$
$$w_1 = \lambda, \;\; w_2 = w_7 = d_0,$$
$$H = \{0, 1, 2, 3, 4, 5, 6, 7\},$$

and the following rules (we also give explanations about the use of these rules):

### Generation phase

G1 $[\,d_i\,]_2 \to [\,O_1\,]_3[\,O_2\,]_4$, $0 \le i < n$.
  $[\,d_i \to d_i d'_i\,]_2$, $0 \le i < n$.

G2 $[\,d_i\,]_3 \to [\,O_1\,]_2[\,O_2\,]_0$, $0 \le i < n$.
  $[\,d_i \to d_{i+1} d'_0\,]_3$, $0 \le i < n$.

G3 $[\,d'_i\,]_4 \to [\,O_1\,]_2[\,O_2\,]_0$, $0 \le i < n$.
  $[\,d'_i \to d_{i+1} d'_0\,]_4$, $0 \le i < n$.

G4 $[\,d_n\,]_2 \to [\,O_1\,]_5[\,O_2\,]_0$.
  $[\,d_n \to d_0 d'_0\,]_2$.

The objects $d_i$ and $d'_i$ are used to control the generation process. In $2n + 1$ steps, $2^n$ membranes with label 5 are created, corresponding to the truth assignments of the variables $x_1, \cdots, x_n$. During this process, the object $d_i$ inside the membrane with label 3 corresponds to the *true* value of variable $x_{i+1}$, and the object $d'_i$ inside the membrane with label 4 corresponds to the *false* value of variable $x_{i+1}$. The created membranes with label 0 are dummy membranes: no rules can be applied to them.

G5 $[\,x_{i,j} \to x_{i,j-1} x'_{i,j-1}\,]_2$, $1 \le i \le m$, $1 \le j \le n$.
  $[\,\bar{x}_{i,j} \to \bar{x}_{i,j-1} \bar{x}'_{i,j-1}\,]_2$, $1 \le i \le m$, $1 \le j \le n$,

G6 $[\,x_{i,0} \to c_i\,]_3$, $1 \le i \le m$.
  $[\,\bar{x}_{i,0} \to \lambda\,]_3$, $1 \le i \le m$.

G7 $[\,\bar{x}'_{i,0} \to c_i\,]_4$, $1 \le i \le m$.
  $[\,x'_{i,0} \to \lambda\,]_4$, $1 \le i \le m$.

G8 $[\ x'_{i,j} \rightarrow x_{i,j}]_4$, $1 \le i \le m$, $1 \le i \le n-1$.

G9 $[\ c_i \rightarrow c_i c'_i]_2$, $1 \le i \le m$.

G10 $[\ c'_i \rightarrow c_i]_4$, $1 \le i \le m$.

The labels of the created membranes toggles between 2 at even steps and 3 or 4 at odd steps. Every object $x_{i,j}$ of the input evolves to $x_{i,0}$ or $x'_{i,0}$ in $2j-1$ steps. Then, it evolves to $c_i$ in membranes where *true* value was chosen for $x_j$ (recall that $x_{i,j} = true$ satisfies clause $C_i$) and is erased in membranes where *false* value was chosen for $x_j$. Similarly, $\bar{x}_{i,j}$ changes to $c_i$ if $x_j = false$ and is erased if $x_j = true$. After $2n+1$ steps, the membranes with label 5 will represent all possible truth assignments of the variables in $\beta$. Every such membrane will contain $d_0$ and the objects representing the clauses satisfied by the present truth assignment.

### Checking phase

C1 $[\ c_1]_5 \rightarrow [\ O_1]_6 [\ O_2]_0$.
$\quad [\ c_1 \rightarrow c_0 d'_0]_5$.
$\quad [\ c_0 \rightarrow \lambda]_5$.

C2 $[\ c_i \rightarrow c_{i-1}]_6$, $1 \le i \le m$,

C3 $[\ d_i]_6 \rightarrow [\ O_1]_5 [\ O_2]_0$.
$\quad [\ d_i \rightarrow d_{i+1} d'_0]_6$.

C4 $[\ d_m \rightarrow e f_0]_5$.

A membrane with label 5 where object $c_1$ appears will change the label to 6 (recall that no rule is ever applied in membranes with label 0 created by separation). In a membrane with label 6, the subscripts of all objects $c_j$ are decremented by one, and at the same time the subscript of $d_i$ is incremented by one and the label of membrane changes back to 5.

If in the beginning of the checking phase $c_1, \cdots, c_i$ are present ($1 \le i \le m-1$), but $c_{i+1}$ is absent, then the evolution of the membrane finishes after $2i$ steps with label 5, with $d_i$ and without $c_1$. If all objects $c_i, 1 \le i \le m$, are present in the beginning of the checking phase, then after $2m$ steps they will all be rewritten into $c_0$, then removed, and $d_0$ will evolve into $d_m$ (and into $e f_0$ in one more step).

C5 $[\ e]_5 \rightarrow [\ ]_5 e$.
$\quad [\ f_0 \rightarrow f_1]_5$.

C6 $e[\ ]_7 \rightarrow [\ e]_7$.
$\quad [\ f_1]_5 \rightarrow [\ O_1]_6 [\ O_2]_0$.
$\quad [\ f_1 \rightarrow d_{2m+2n+4} d'_0]_5$.

C7 $e[\ ]_6 \to [\ e]_6$.

If $\beta$ has solutions (suppose $\beta$ has $s$ solutions, $1 \le s \le 2^n$), then at step $2n + 2m + 3$, every membrane corresponding to a solution of $\beta$ ejects $e$ in the skin region, and at the same time $f_0$ changes to $f_1$. At step $2n + 2m + 4$, one copy of $e$ enters the membrane with label 7, and $s$ membranes change label from 5 to 6 by the separation rule $[\ f_1]_5 \to [\ O_1]_6[\ O_2]_0$. At step $2n + 2m + 5$, $s - 1$ copies of $e$ enter in $s - 1$ membranes of the $s + 1$ membranes with labels 6 and 7. If $\beta$ has no solution, then no object $e$ enters the membrane labeled 7.

### Output phase

O1 $[\ d_i \to d_{i+1}]_7$, $0 \le i \le 2m + 2n + 5$.

O2 $[\ e]_7 \to [\ O_1]_6[\ O_2]_0$.
$[\ e \to \mathtt{yes}d_0']_7$.

O3 $[\ \mathtt{yes}]_6 \to [\ ]_6\mathtt{yes}$.

O4 $[\ \mathtt{yes}]_1 \to [\ ]_1\mathtt{yes}$.

O5 $[\ d_i \to \lambda]_6$, $i \in \{2n + 2m + 5, 2n + 2m + 6\}$.

O6 $[\ d_{2n+2m+6}]_7 \to [\ ]_7\mathtt{no}$.

O7 $[\ \mathtt{no}]_1 \to [\ ]_1\mathtt{no}$.

If $\beta$ has solutions, then at step $2n + 2m + 4$ the membrane with label 7 receives a copy of $e$ by the rule C6. In this case, the rule O2 will be applied either at step $2n + 2m + 5$ or at step $2n + 2m + 6$ (this can happen if $s > 1$ and the rule C6 is applied at step $2n + 2m + 5$), changing the label of the membrane from 7 to 6. It will take two more steps to eject object $\mathtt{yes}$ in the skin and then into the environment. If $\beta$ has no solutions, then after step $2n + 2m + 6$ the membrane with label 7 remains with label 7 and then the rule O6 is applied, ejecting object $\mathtt{no}$ into the skin and then into the environment.

If $\beta$ has at least two solutions, then the behavior of this system is not deterministic: at step $2n + 2m + 5$ either one of the rules C6 and O2 can be applied to the membrane with label 7 (applying C6 at step $2n + 2m + 5$ results in one extra copy of $e$ in membrane with label 7 and one copy of $e$ missing in some membrane with label 6). However, the system is confluent: in either case mentioned above, after at most three further steps, the system produces the output $\mathtt{yes}$ and halts in the same configuration (the membrane with label 7 changes its label to 6 and the counter $d_{2n+2m+5}$ or $d_{2n+2m+6}$ is erased). $\qquad \square$

From the confluence point of view, using rules of type $(c_0')$ allows to obtain a stronger result:

**Theorem 3.** *P systems with rules of types* $(a_0), (c'_0), (g_0)$ *can solve* SAT *in linear time in a uniform and deterministic way.*

*Proof.* We consider a propositional formula $\beta$ in the conjunctive normal form and in the previous proofs, encoded as a multiset $w$ over $\Sigma(\langle n, m \rangle)$ in the same way as above. For given $(n, m) \in \mathbb{N}^2$, we construct a recognizing P system $(\Pi(\langle n, m \rangle), V(\langle n, m \rangle), 2)$, with

$$\Pi(\langle n, m \rangle) = (O(\langle n, m \rangle), H, \mu, w_1, w_2, R),$$
$$O(\langle n, m \rangle) = O_1 \cup O_2,$$
$$O_1 = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \le i \le m, 1 \le j \le n\} \cup \{d_i \mid 0 \le i \le 4n + 2m + 2\}$$
$$\cup \{e_i, \bar{e}_i \mid 0 \le i \le n - 1\} \cup \{c_{i,0}, c_{i,1}, c_{i,2} \mid 0 \le i \le m\}$$
$$\cup \{a, u, \bar{u}, v, \bar{v}, t, \lambda, \texttt{yes}, \texttt{no}\},$$
$$O_2 = \{x'_{i,j}, \bar{x}'_{i,j} \mid 1 \le i \le m, 1 \le j \le n\} \cup \{\bar{e}'_i \mid 0 \le i \le n - 1\}$$
$$\cup \{f, \bar{u}', \bar{v}'\},$$
$$\mu = [\,[\,]_2\,]_1,$$
$$w_1 = w_2 = d_0,$$
$$H = \{1, 2, 3, 4, 5, 6, 7\},$$

and the following rules:

### Generation phase

G1 $[\, d_i \to e_i a u \,]_2$, $0 \le i \le n - 2$.
$[\, d_{n-1} \to e_{n-1} a v \,]_2$.

G2 $[\, a \,]_2 \to [\, O_1 \,]_2 [\, O_2 \,]_2$.
$[\, a \to t f \,]_2$.
$[\, e_i \to \bar{e}_i \bar{e}'_i \,]_2$, $0 \le i \le n - 1$.
$[\, u \to \bar{u} \bar{u}' \,]_2$.
$[\, v \to \bar{v} \bar{v}' \,]_2$.

G3 $[\, t \,]_2 \to [\,]_3 \lambda$,
$[\, f \,]_2 \to [\,]_4 \lambda$,

G4 $[\, \bar{e}_i \to d_{i+1} \,]_3$, $0 \le i \le n - 2$.
$[\, \bar{e}'_i \to d_{i+1} \,]_4$, $0 \le i \le n - 2$.
$[\, \bar{e}_{n-1} \to d_0 u \,]_3$.
$[\, \bar{e}'_{n-1} \to d_0 u \,]_4$.

G5 $[\, \bar{u} \,]_3 \to [\,]_2 \lambda$.
$[\, \bar{u}' \,]_4 \to [\,]_2 \lambda$.
$[\, \bar{v} \,]_3 \to [\,]_5 \lambda$.
$[\, \bar{v}' \,]_4 \to [\,]_5 \lambda$.

In $4n$ steps, $2^n$ membranes are created, corresponding to the truth assignments of the variables $x_1, \cdots, x_n$. During this process, object $d_i$ inside the membrane with label 3 corresponds to the *true* value of variable $x_{i+1}$, and object $d_i$ inside the membrane with label 4 corresponds to the *false* value of variable $x_{i+1}$. Object $a$ is used to choose the truth assignment of variables, and objects $u$ and $u'$ are used to change the membrane label back to 2. At step $4n$, the labels 3 and 4 of internal membranes are changed to 5 by the objects $v$ and $v'$. The membranes with label 5 represent all possible truth assignments of the variables in $\beta$. Every such membrane will contain $d_0$, $u$, and the objects representing the clauses satisfied.

G6 $\left[\, x_{i,j} \to x_{i,j} x'_{i,j} \right]_2$, $1 \le i \le m$, $1 \le j \le n$.
    $\left[\, \bar{x}_{i,j} \to \bar{x}_{i,j} \bar{x}'_{i,j} \right]_2$, $1 \le i \le m$, $1 \le j \le n$.

G7 $\left[\, x_{i,1} \to c_{i,0} \right]_3$, $1 \le i \le m$.
    $\left[\, \bar{x}_{i,1} \to \lambda \right]_3$, $1 \le i \le m$.

G8 $\left[\, \bar{x}'_{i,1} \to c_{i,0} \right]_4$, $1 \le i \le m$.
    $\left[\, x'_{i,1} \to \lambda \right]_4$, $1 \le i \le m$.

G9 $\left[\, x_{i,j} \to x_{i,j-1} \right]_3$, $1 \le i \le m$, $2 \le i \le n$.
    $\left[\, \bar{x}_{i,j} \to \bar{x}_{i,j-1} \right]_3$, $1 \le i \le m$, $2 \le i \le n$.
    $\left[\, x'_{i,j} \to x_{i,j-1} \right]_4$, $1 \le i \le m$, $2 \le i \le n$.
    $\left[\, \bar{x}'_{i,j} \to \bar{x}_{i,j-1} \right]_4$, $1 \le i \le m$, $2 \le i \le n$.

The label of the created membranes is 2 and then changes to 3 or 4 at steps $4i + 3$, $0 \le i < n$. Every object $x_{i,j}$ of the input evolves to $x_{i,1}$ or $x'_{i,1}$ in $4(i - 1)$ steps. Then, it evolves to $c_{i,0}$ in membranes where *true* value was chosen for $x_j$ (recall that $x_{i,j} = true$ satisfies clause $C_i$) and is erased in membranes where *false* value was chosen for $x_j$. Similarly, $\bar{x}_{i,j}$ changes to $c_{i,0}$ if $x_j = false$, and is erased if $x_j = true$.

G10 $\left[\, c_{i,0} \to c_{i,1} \right]_2$, $1 \le i \le m$.

G11 $\left[\, c_{i,1} \to c_{i,2} c'_{i,2} \right]_2$, $1 \le i \le m$.

G12 $\left[\, c_{i,2} \to c_{i,0} \right]_3$, $1 \le i \le m$.
     $\left[\, c'_{i,2} \to c_{i,0} \right]_4$, $1 \le i \le m$.

The rules of types (G10), (G11), and (G12) are representing the fact that if the clause $C_i$ is satisfied in an internal membrane, then in the new created membranes from it $C_i$ is also satisfied.

### Checking phase

C1 $\left[\, c_{i,0} \to c_{i-1,0} \right]_5$, $1 \le i \le m$.

C2 $[\,u\,]_5 \to [\,\,]_6\lambda.$

C3 $[\,c_{0,0}\,]_6 \to [\,\,]_5\lambda.$

C4 $[\,d_i \to d_{i+1}u\,]_6,\ 0 \le i < m-1.$

C5 $[\,d_{m-1} \to d_m\,]_6.$

By expelling object $u$, membrane changes label from 5 to 6. At the same time the subscripts of all objects $c_{j,0}$ are decremented by one. A membrane with label 6 where object $c_{0,0}$ appears will change the label back to 5. At the same time the subscript of $d_i$ is incremented by one and $u$ is reproduced (except for $i = m-1$).

If in the beginning of the checking phase $c_{1,0}, \cdots, c_{i,0}$ are present $(1 \le i < m)$, but $c_{i+1,0}$ is absent, then after $2i + 1$ steps rule C3 will no longer be applicable and the membrane will have label 6, no object $c_{0,0}$, and will never change the label again. After $m + i + 1$ steps from the beginning of the checking phase, the membrane will stop evolving. If all objects $c_{i,0}$, $1 \le i \le m$, are present in the beginning of the checking phase, then after $2m$ steps they will all be erased, $d_0$ will evolve into $d_m$ and the membrane label will be 5.

### Output phase

O1 $[\,d_m\,]_5 \to [\,\,]_5\mathtt{yes}.$

O2 $[\,\mathtt{yes}\,]_1 \to [\,\,]_7\mathtt{yes}.$

O3 $[\,d_i \to d_{i+1}\,]_1,\ 0 \le i \le 4n + 2m + 1.$

O4 $[\,d_{4n+2m+2}\,]_1 \to [\,\,]_1\mathtt{no}.$

At step $4n + 2m + 1$, every membrane corresponding to a solution of $\beta$ expels **yes** in the skin region, and in the next step one copy of **yes** (if any) is ejected into the environment, changing the label of the skin from 1 to 7. If $\beta$ has no solutions, then after step $4n + 2m + 2$ the skin membrane remains with label 1 and then rule O4 is applied, ejecting the object **no** into the environment.     $\square$

### 4.2   Universality

Because the notion of a matrix grammar will be used below, here we introduce it.

A *matrix grammar with appearance checking* is a construct $G = (N, T, S, M, F)$, where $N, T$ are disjoint alphabets, $S \in N$, $M$ is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \ge 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and $F$ is a set of occurrences of rules in $M$ ($N$ is the nonterminal alphabet, $T$ is the terminal alphabet, $S$ is the axiom, while the elements of $M$ are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \implies z$ if there is a matrix $(A_1 \to x_1,$ $\ldots, A_n \to x_n)$ in $M$ and the strings $w_i \in (N \cup T)^*, 1 \le i \le n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \le i \le n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or $w_i = w_{i+1}$, $A_i$ does not appear in $w_i$, and the rule $A_i \to x_i$ appears in $F$. (The rules of a matrix are applied in order, possibly skipping the rules in $F$ if they cannot be applied − therefore we say that these rules are applied in the *appearance checking* mode.)

The language generated by $G$ is defined by $L(G) = \{w \in T^* \mid S \implies^* w\}$. The family of languages of this form is denoted by $MAT_{ac}$. It is known that $MAT_{ac} = RE$.

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in $M$ are in one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,

2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \le 2$,

3. $(X \to Y, A \to \#)$, with $X, Y \in N_1, A \in N_2$,

4. $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*, |x| \le 2$.

Moreover, there is only one matrix of type 1 (that is why one uses to write it in the form $(S \to X_{init}A_{init})$, in order to fix the symbols $X, A$ present in it), and $F$ consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

For each matrix grammar there is an equivalent matrix grammar in the binary normal form. Details can be found in [Dassow and Păun 1989].

The following theorem shows that by using membrane separation rules to change the labels of the membranes, the universality can be reached. Here $Ps(\Pi)$ denotes the set of vectors of natural numbers describing the multiplicity of objects expelled into the environment by the various halting computations in system $\Pi$; by $PsOP(a_0, c_0, g_0')$ we denote the family of sets $Ps(\Pi)$ computed by P systems using the types of rules $a_0$, $c_0$, and $g_0'$.

**Theorem 4.** $PsOP(a_0, c_0, g_0') = PsRE$.

*Proof.* Consider a matrix grammar $G = (N, T, S, M, F)$ with appearance checking, in the binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$ and with the matrices of the four forms introduced above. Assume that all matrices are injectively labeled with elements of a set $B$. Replace the rule $X \to \lambda$ from matrices of type 4 by $X \to f$, where $f$ is a new symbol.

We construct the P system of degree 2

$$\Pi = (O, H, [\ [\ ]_{X_{init}}]_1, w_1 = \lambda, w_{X_{init}} = c_0 A_{init}, R),$$
$$O = O_1 \cup O_2,$$
$$O_1 = T \cup N_2 \cup \{A_m \mid A \in N_2, m \in B\} \cup \{c, c_0, c_1, c_2, \#\},$$
$$O_2 = \{d\},$$
$$H = N_1 \cup \{X_m \mid X \in N_1, m \in B\} \cup \{0, 1, f\},$$

and the set $R$ containing the following rules.

The simulation of a matrix $m : (X \to Y, A \to x)$, with $X \in N_1, Y \in N_1 \cup \{f\}$, is done in three steps, using the next rules:

1. $[\ A]_X \to [\ O_1]_{Y_m}[\ O_2]_0$.
   $[\ A \to A_m d]_X$.

2. $[\ A_m \to xc]_{Y_m}$.

3. $[\ c]_{Y_m} \to [\ O_1]_Y [\ O_2]_0$.
   $[\ c \to d]_{Y_m}$.

4. $[\ c_1 \to c_2]_{Y_m}$.
   $[\ c_2 \to c_0]_{Y_m}$.

The first rule of the matrix is simulated by the change of the label of the inner membrane (the "dummy" object $d$ and membrane 0 play no further role). Note that if $X \in N_1$ also appears in a matrix of type 3, then when $A$ activates the separation, $c_0$ can evolve to $c_1 d$ by the second rule of type (5), then to $c_2$ and $c_0$ by the rules of type (4). The correctness of the simulation is obvious, because one cannot simulate one rule of the matrix without simulating the other rule.

The simulation of a matrix $m : (X \to Y, A \to \#)$, with $X, Y \in N_1$ and $A \in N_2$, is done also in three steps, using the next rules:

5. $[\ c_0]_X \to [\ O_1]_{Y_m}[\ O_2]_0$.
   $[\ c_0 \to c_1 d]_X$.

6. $[\ c_1 \to c_2]_{Y_m}$.
   $[\ A \to \#]_{Y_m}$.

7. $[\ c_2]_{Y_m} \to [\ O_1]_Y [\ O_2]_0$.
   $[\ c_2 \to c_0 d]_{Y_m}$.

While the membrane with label $X$ is used by object $c_0$, no other rule can be used. In the next step, if any copy of $A$ is present, then it introduces the trap-object $\#$ and the computation never stops. If no $A$ is present, then the objects $c_j$ evolve, returning the label of the membrane to $Y$ and introducing the auxiliary object $c_0$, for iterating the procedure.

We also consider the following rules:

8. $[\, A \rightarrow \#\,]_f$, for all $A \in N_2$.

9. $[\, \# \rightarrow \#\,]_h$, for all $h \in H$.

10. $[\, a]_f \rightarrow [\ ]_f a$.

11. $[\, a]_1 \rightarrow [\ ]_1 a$, for all $a \in T$.

The equality $\Psi_T(L(G)) = Ps(\Pi)$ easily follows from the above explanations. $\quad\square$

*Remark.* In the above proof, the rules of type $(c_0)$ are only used for sending the result of a computation out of the system. Therefore, rules of types $(a_0)$ and $(g'_0)$ are sufficient to reach universality for membrane systems with internal output.

## 5    Final Remark

In this paper, separation is introduced into active P systems, and the efficiency and universality of active P systems with separation rules instead of division are investigated. The universality result is obtained by a direct proof. It still remains open using P systems with separation rules to simulate P systems with division rules.

### Acknowledgements

### References

[Alhazov and Ishdorj 2004] Alhazov A., Ishdorj T.-O.: "Membrane Operations in P Systems with Active Membranes"; "Second Brainstorming Week on Membrane Computing" (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Capparini, eds.), Sevilla, 2-7 February 2004, Research Group in Natural Computing TR 01/2004, University of Seville, 37–44

[Alhazov et al. 2003] Alhazov A., Martín-Vide C., Pan L.: "Solving a **PSPACE**-Complete Problem by P Systems with Restricted Active Membranes"; Fundamenta Informaticae, 58, 2 (2003), 66–77

[Alhazov and Pan 2004] Alhazov A., Pan L.: "Polarizationless P Systems with Active Membranes"; to appear in Grammars 7 (2004)

[Alhazov et al. 2003] Alhazov A., Pan L., Păun Gh.: "Trading Polarizations for Labels in P Systems with Active Membranes"; submitted (2003)

[Dassow and Păun 1989] Dassow J., Păun Gh.: "Regulated Rewriting in Formal Language Theory"; Springer, Berlin (1989)

[Papadimitriou 1994]  Papadimitriou Ch. P.: " Computational Complexity"; Addison-Wesley, Reading, MA (1994)

[Păun 2000]  Păun Gh.: "Computing with Membranes"; Journal of Computer and System Sciences, 61, 1 (2000), 108–143

[Păun 2001]  Păun Gh.: "P Systems with Active Membranes: Attacking **NP**-Complete Problems"; Journal of Automata, Languages and Combinatorics, 6, 1 (2001), 75–90

[Păun 2002]  Păun Gh.: "Computing with Membranes: An Introduction"; Springer-Verlag, Berlin (2002)

[Pérez-Jiménez et al. 2002]  Pérez-Jiménez  M.J.,  Romero-Jiménez  A.,  Sancho-Caparrini F.: "Teoría de la Complejidad en Modelos de Computatión Celular con Membranas", Editorial Kronos, Sevilla (2002)

[Rozenberg and Salomaa 1997]  Rozenberg G., Salomaa A. (eds.): "Handbook of Formal Languages (3 volumes)"; Springer, Berlin (1997)

[Salomaa 1973]  Salomaa A.: "Formal Languages"; Academic Press, New York (1973)

[Sosík 2003]  Sosík P.: "Solving a **PSPACE**-Complete Problem by P Systems with Active Membranes"; "Proceedings of the Brainstorming Week on Membrane Computing" (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Report GRLMC 26/03, 2003, 305–312