

# Further Remarks on P Systems with Active Membranes, Separation, Merging, and Release Rules <sup>\*</sup>

Linqiang PAN<sup>1,3</sup>, Artiom ALHAZOV<sup>2,3</sup>, and Tseren-Onolt ISHDORJ<sup>3</sup>

<sup>1</sup> Department of Control Science and Engineering  
Huazhong University of Science and Technology  
Wuhan 430074, Hubei, People's Republic of China  
`lqpan@mail.hust.edu.cn`

<sup>2</sup> Institute of Mathematics and Computer Science  
Academy of Science of Moldova  
Str. Academiei 5, Chişinău, MD 2028, Moldova  
`artiom@math.md`

<sup>3</sup> Research Group on Mathematical Linguistics  
Rovira i Virgili University  
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain  
{`lp@fll`, `artiome.alhazov@estudiants`,  
`tserenonolt.ishdorj@estudiants`}.`urv.es`

**Abstract.** The P systems are a class of distributed parallel computing devices of a biochemical type. In this note, we show that by using membrane separation to obtain exponential workspace, SAT problem can be solved in linear time in a uniform and confluent way by active P systems without polarizations. This improves some results already obtained by A. Alhazov, T.-O. Ishdorj. A universality result related to membrane separation is also obtained.

## 1 Introduction

The P systems are a class of distributed parallel computing devices of a biochemical type, introduced in [4], which can be seen as a general computing architecture where various types of objects can be processed by various operations. The approach starts from the observation that certain processes which take place in the complex structure of living organisms can be considered as computations. For a motivation and detailed description of various P system models we refer to [4], [5].

Informally speaking, in P systems with active membranes without polarizations one uses six types of rules:  $(a_0)$  multiset rewriting rules,

---

<sup>\*</sup> Dedicated to Prof. Zhang Kemin for his 70th birthday

( $b_0$ ) rules for introducing objects into membranes, ( $c_0$ ) rules for sending objects out of membranes, ( $d_0$ ) rules for dissolving membranes, ( $e_0$ ) rules for dividing elementary membranes, and ( $f_0$ ) rules for dividing non-elementary membranes, see [2]. In these rules, a single object participates during the process. A few more types of rules are introduced in [1]: ( $g_0$ ) membrane merging rules, ( $h_0$ ) membrane separation rules, ( $i_0$ ) membrane release rules in P systems with active membranes without polarizations. A common feature of those rules is the use of multisets of objects during the computation process.

By using membrane separation to obtain an exponential working space in a linear time, it is shown in [1] that SAT problem can be solved in linear time in a semi-uniform and confluent way by some particular combinations of types of rules in P systems with active membranes. In this note, the results are improved in the uniform way. A result about universality is also obtained.

## 2 P Systems with Active Membranes and New Membrane Operations

We assume the reader to be familiar with basic elements of membrane computing, for instance, from [5] (details and recent results from membrane computing can be found at the web address <http://psystems.disc.o.unimib.it>). We only mention that  $RE$  denotes 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  $\Psi_V$ .

The P systems with active membranes with some new operations (without polarization) of the form  $\Pi = (O, H, \mu, w_1, \dots, w_m, R)$  were considered in [1], with the following components:

1.  $m \geq 1$  is the initial degree of the system;
2.  $O$  is the alphabet of *objects*;
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.  $w_1, \dots, w_m$  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:
  - ( $a_0$ )  $[a \rightarrow v]_h$ , for  $h \in H, a \in O, v \in O^*$   
 (object evolution rules, associated with membranes and depending on the label, 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);

- ( $b_0$ )  $a[ ]_h \rightarrow [ b ]_h$ , for  $h \in H, a, b \in O$   
(communication rules; an object is introduced in the membrane during this process);
- ( $c_0$ )  $[ a ]_h \rightarrow [ ]_h b$ , for  $h \in H, a, b \in O$   
(communication rules; an objects sent out of the membrane during this process);
- ( $d_0$ )  $[ a ]_h \rightarrow b$ , 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);
- ( $e_0$ )  $[ a ]_h \rightarrow [ b ]_h [ c ]_h$ , 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 specified in the rule is replaced in the two new membranes by possibly new objects);
- ( $g_0$ )  $[ ]_h [ ]_h \rightarrow [ ]_h$ , for  $h \in H$   
(merging rules for elementary membranes; in reaction of two membranes, they are merged into a single membrane; the objects of the former membranes are put together in the new membrane);
- ( $h_0$ )  $[ O ]_h \rightarrow [ U ]_h [ O - U ]_h$ , for  $h \in H, U \subset O$   
(separation rules for elementary membranes; the membrane is separated into two membranes with the same labels; the objects from  $U$  are placed in the first membrane, those from  $O - U$  are placed in the other membrane);
- ( $i_0$ )  $[ [ O ]_h ]_h \rightarrow [ ]_h O$ , for  $h \in H$   
(release rule; the objects in a membrane are released out of a membrane, surrounding it, while the first membrane disappears).

The rules are applied non-deterministically, in the maximally parallel manner; among the rules of types ( $b_0$ ),  $\dots$ , ( $i_0$ ) at most one can be applied to each membrane at each step. In this way, we get transition from a configuration of the system to the next configuration. A sequence of transitions is a computation. A computation is halting if no other rules can be applied in its last configuration.

Rules of types ( $a_0$ ), ( $b_0$ ), ( $c_0$ ), ( $d_0$ ), and ( $e_0$ ) were introduced in [2], and ( $g_0$ ), ( $h_0$ ), and ( $i_0$ ) introduced in [1], without polarizations of membranes, and without the capability of changing the label of membranes they involve. Following the notation in [2] and [5], we use the primed versions to indicate the fact that the labels of membranes can be changed.

For example, the primed versions of merging and separation rules are of the following forms:

$$\begin{aligned} (g'_0) \quad & [ ]_{h_1} [ ]_{h_2} \rightarrow [ ]_{h_3}, \text{ for } h_1, h_2, h_3 \in H. \\ (h'_0) \quad & [ O ]_{h_1} \rightarrow [ U ]_{h_2} [ O - U ]_{h_3}, \\ & \text{for } h_1, h_2, h_3 \in H, U \subset O. \end{aligned}$$

To understand the difference of uniform construction and semi-uniform construction of P systems, we recall some notions about solving decidability problems in the membrane computing framework. Given a decision question  $X$ , 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  $\Pi_n$  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. In [5], the complexity classes related to P systems are defined in the semi-uniform way: P systems are constructed starting not from the size  $n$ , but from an instance  $X(n)$ . For a clearer description of the difference between uniform and semi-uniform constructions, please refer to [6].

### 3 Efficiency

From [1] we know that P systems with rules of types  $(a_0)$ ,  $(b_0)$ ,  $(c_0)$ , and  $(h'_0)$  can solve **SAT** in linear time in a semi-uniform and confluent way. The following theorem improves this result in a uniform way.

**Theorem 1.** *A uniform family of P systems with rules of types  $(a_0)$ ,  $(b_0)$ ,  $(c_0)$ ,  $(h'_0)$  can solve **SAT** in linear time in confluent way.*

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

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

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

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

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

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

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

$$\begin{aligned} \Pi(\langle n, m \rangle) = & (O(\langle n, m \rangle), H, \mu, w_0, w_1, w_s, R), \\ O(\langle n, m \rangle) = & \{x_{i,t,j}, \bar{x}_{i,t,j} \mid 1 \leq i \leq m, 1 \leq j \leq n, \\ & 0 \leq t \leq j\} \\ & \cup \{x'_{i,t,j}, \bar{x}'_{i,t,j} \mid 1 \leq i \leq m, 1 \leq j \leq n, \\ & 0 \leq t \leq j-1\} \\ & \cup \{c_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n+1\} \\ & \cup \{c'_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\} \\ & \cup \{d_i \mid 0 \leq i \leq n+m+7\} \\ & \cup \{d'_i \mid 1 \leq i \leq n+1\} \\ & \cup \{t, \lambda, \text{yes}, \text{no}\}, \\ \mu = & [ [ ]_0 [ ]_1 ]_s, \\ w_s = & \lambda, \\ w_0 = & d_0, \\ w_1 = & d_0, \\ H = & \{s, 0, 1, \dots, m+2\}, \end{aligned}$$

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

**Generation phase:**

$$\begin{aligned} \text{G1. } & [O]_1 \rightarrow [U]_1 [O-U]_1, \\ & U = \{x'_{i,t,j}, \bar{x}'_{i,t,j} \mid 1 \leq i \leq m, 1 \leq j \leq n, \\ & \quad 0 \leq t \leq j-1\} \\ & \cup \{c'_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\} \\ & \cup \{d'_i \mid 1 \leq i \leq n+1\}. \end{aligned}$$

- G2.  $[x_{i,t,j} \rightarrow x_{i,t-1,j}x'_{i,t-1,j}]_1,$   
 $[\bar{x}_{i,t,j} \rightarrow \bar{x}_{i,t-1,j}\bar{x}'_{i,t-1,j}]_1,$   
 $[x'_{i,t,j} \rightarrow x_{i,t-1,j}x'_{i,t-1,j}]_1,$   
 $[\bar{x}'_{i,t,j} \rightarrow \bar{x}_{i,t-1,j}\bar{x}'_{i,t-1,j}]_1,$   
 $1 \leq i \leq m, 1 \leq t \leq j, 1 \leq j \leq n.$
- G3.  $[x_{i,0,j} \rightarrow c_{i,j}c'_{i,j}]_1,$   
 $[\bar{x}_{i,0,j} \rightarrow \lambda]_1,$   
 $[\bar{x}'_{i,0,j} \rightarrow c_{i,j}c'_{i,j}]_1,$   
 $[x'_{i,0,j} \rightarrow \lambda]_1, 1 \leq i \leq m, 1 \leq j \leq n.$
- G4.  $[c_{i,j} \rightarrow c_{i,j+1}c'_{i,j+1}]_1,$   
 $[c'_{i,j} \rightarrow c_{i,j+1}c'_{i,j+1}]_1,$   
 $1 \leq i \leq m, 1 \leq j \leq n-1.$
- G5.  $[c_{i,n} \rightarrow c_{i,n+1}]_1,$   
 $[c'_{i,n} \rightarrow c_{i,n+1}]_1, 1 \leq i \leq m.$
- G6.  $[d_i \rightarrow d_{i+1}d'_{i+1}]_1, 0 \leq i \leq n.$   
 $[d'_i \rightarrow d_{i+1}d'_{i+1}]_1, 1 \leq i \leq n.$   
 $[d_{n+1} \rightarrow d_{n+2}]_1.$   
 $[d'_{n+1} \rightarrow d_{n+2}]_1.$

In  $n+1$  steps, by rules of types G1–G4,  $2^n$  membranes with label 1 are created, corresponding to the all possible  $2^n$  truth assignments of the variables  $x_1, x_2, \dots, x_n$ . During this process, every object  $x_{i,j,j}$  of the input evolves to  $x_{i,0,j}$  and  $x'_{i,0,j}$  in  $j$  steps. Then, they evolve to  $c_{i,j}$  in membranes where *true* value was chosen for  $x_j$  (recall that  $x_{i,j,j} = \text{true}$  satisfies clause  $C_i$ ) and are erased in membranes where *false* value was chosen for  $x_j$ . In the next  $n-j$  steps,  $c_{i,j}$  evolves to  $c_{i,n}$  or  $c'_{i,n}$ . It takes one more step  $c_{i,n}$  and  $c'_{i,n}$  evolve to  $c_{i,n+1}$  by rules of type G5, which means the system is ready for checking phase. Similarly,  $\bar{x}_{i,j,j}$  changes to  $c_{i,n+1}$  if  $x_j = \text{false}$  and is erased if  $x_j = \text{true}$ . Note that at step  $n+2$ , each membrane with label 1 has the object  $d_{n+2}$ .

**Checking phase:**

- C1.  $[O]_i \rightarrow [U]_i [O-U]_{i+1},$   
 $U = \{c_{i,n+1}\}, 1 \leq i \leq m.$

Starting with  $i = 1$ , in membranes with label  $i$ , objects  $c_{i,n+1}$  will be separated from the other objects, and the label of the membrane with objects from  $O - \{c_{i,n+1}\}$  will become  $i+1$ . The membranes which do not contain the objects  $c_{i+1,n+1}$  will never evolve anymore. If all objects  $c_{i,n+1}, 1 \leq i \leq m$ , are present in some membrane, then after  $m$  steps this membrane will evolve into a membrane with label  $m+1$ , containing object  $d_{n+2}$ , by the rules of type C1.

- C2.  $[d_{n+2}]_{m+1} \rightarrow [ ]_{m+1} d_{n+2}$ .  
 C3.  $[d_{n+2} \rightarrow tt]_s$ .

If  $\beta$  has solutions, then at step  $n + m + 3$ , every membrane corresponding to a solution of  $\beta$  ejects  $d_{n+2}$  in the skin region, then they will all be rewritten into  $tt$ .

- C4.  $t[ ]_0 \rightarrow [t]_0$ .  
 C5.  $t[ ]_{m+1} \rightarrow [t]_{m+1}$ .  
 C6.  $[O]_0 \rightarrow [U']_{m+1}[O - U']_{m+2}$ ,  
 $U' = \{t\}$ .  
 C7.  $[d_i \rightarrow d_{i+1}]_0, 0 \leq i \leq n + m + 5$ .  
 C8.  $[d_{n+m+6} \rightarrow d_{n+m+7}]_{m+2}$ .

At step  $n + m + 5$ , one copy of  $t$  enters the membrane with label 0, and (suppose  $\beta$  has  $s$  solutions,  $1 \leq s \leq 2^n$ )  $s$  copies of  $t$  enter the  $s$  membranes with label  $m + 1$ . At step  $n + m + 6$ ,  $s - 1$  copies of  $t$  enter the membranes with label  $m + 1$ , or  $s - 2$  copies of  $t$  enter the  $s - 2$  membranes with label  $m + 1$ , and 1 copy of  $t$  enters the membrane with label 0. Using rule C6, membrane with label 0 is separated into two membranes, which contain object  $t$  and object  $d_{n+m+6}$  or  $d_{n+m+7}$  respectively. If  $\beta$  has no solution, then no object enters membrane labeled 0 and rule C6 is not applied.

**Output phase:**

- O1.  $[t \rightarrow \lambda]_{m+1}$ .  
 O2.  $[d_{n+m+7}]_0 \rightarrow [ ]_0 \mathbf{no}$ .  
 O3.  $[d_{n+m+7}]_{m+2} \rightarrow [ ]_{m+2} \mathbf{yes}$ .  
 O4.  $[\mathbf{no}]_s \rightarrow [ ]_s \mathbf{no}$ .  
 O5.  $[\mathbf{yes}]_s \rightarrow [ ]_s \mathbf{yes}$ .

If  $\beta$  has solutions, then at step  $n + m + 8$ , object  $d_{n+m+7}$  in the membrane with label  $m + 2$  ejects **yes** into skin and then into the environment. If  $\beta$  has no solution, then after  $n + m + 8$  steps object  $d_{n+m+7}$  ejects object **no** into skin and then into the environment.

From the previous explanation of the use of rules, one can easily see how the P system designed in the above proof works, and it halts at step  $n + m + 9$ . It is easy to prove that the designed P system is uniform. But If  $\beta$  has at least two solutions, then the behavior of this system is not deterministic: at step  $n + m + 6$  either one of the rules of types C4 and C6 can be applied to the membrane with label 0 (applying C4 in step  $2n + 2m + 6$  results in one extra copy of  $t$  in membrane with

label 0 and one copy of  $t$  missing in some membrane with label  $m + 1$ ). However, the system is confluent: in either case mentioned above, after three further steps, the system produces the output **yes** and halts in the same configuration (in the membranes with label  $m + 1$ , the objects  $t$  are erased).  $\square$

The following two theorems improve the corresponding theorems in [1], which are not difficult to be proved by using the generation phase in the proof of Theorem 1 and the proofs in Theorem 2 and 3 in [1]. So, here we omit the proofs.

**Theorem 2.** *A uniform family of P systems with rules of types  $(a_0)$ ,  $(c_0)$ ,  $(g_0)$ ,  $(h'_0)$  can solve SAT in linear time in a confluent way.*

**Theorem 3.** *A uniform family of P systems with rules of types  $(a_0)$ ,  $(g_0)$ ,  $(h'_0)$ ,  $(i_0)$  can solve SAT in linear time in a confluent way.*

## 4 Universality

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

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  $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n)$ ,  $n \geq 1$ , 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  $w \Longrightarrow z$  if there is a matrix  $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n)$  in  $M$  and the strings  $w_i \in (N \cup T)^*$ ,  $1 \leq i \leq n+1$ , such that  $w = w_1, z = w_{n+1}$ , and, for all  $1 \leq i \leq n$ , 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  $A_i \rightarrow x_i$  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  $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$ . 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.  $(S \rightarrow XA)$ , with  $X \in N_1, A \in N_2$ ,
2.  $(X \rightarrow Y, A \rightarrow x)$ , with  $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$ ,
3.  $(X \rightarrow Y, A \rightarrow \#)$ , with  $X, Y \in N_1, A \in N_2$ ,
4.  $(X \rightarrow \lambda, A \rightarrow x)$ , with  $X \in N_1, A \in N_2$ , and  $x \in T^*, |x| \leq 2$ .

Moreover, there is only one matrix of type 1 (that is why one uses to write it in the form  $(S \rightarrow X_{init}A_{init})$ , in order to fix the symbols  $X, A$  present in it), and  $F$  consists exactly of all rules  $A \rightarrow \#$  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 [3].

The following theorem shows that by using membrane separation rule 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  $PsOP(a_0, c_0, h'_0)$  we denote the family of sets  $Ps(\Pi)$  computed by P systems using the types of rules  $a_0, c_0$ , and  $h'_0$ .

**Theorem 4.**  $PsOP(cat_1, c_0, h'_0) = PsRE$ .

*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 of forms 2, 3, and 4 are injectively labeled with elements of a set  $B$  (without loss of generality, suppose  $0 \notin B$ ),  $B = B_1 \cup B_2$ ,  $B_1$  for the matrices of forms 2 and 4, and  $B_2$  for the matrices of form 3. Replace the rule  $X \rightarrow \lambda$  from matrices of type 4 by  $X \rightarrow f$ , where  $f$  is a new symbol.

We construct the P system of degree 2

$$\begin{aligned} \Pi &= (O, C, H, [ [ ]_{X_{init}} ]_1, w_1 = \lambda, w_{X_{init}} = cc_0dA_{init}, R), \\ O &= T \cup N_2 \cup \{c_0, c_1, c_2, c_3, c_4, \#\}, \\ C &= \{c\}, \\ H &= N_1 \cup \{X_m, X'_m \mid X \in N_1, m \in B\} \cup \{0, 1, f\}, \end{aligned}$$

and the set  $R$  containing the following rules. We present them in blocks as used for simulating matrices of  $G$ , thus also having clear the way the system  $\Pi$  works.

The simulation of a matrix  $m : (X \rightarrow Y, A \rightarrow x)$ , with  $X \in N_1, Y \in N_1 \cup \{f\}, A \in N_2$ , and  $x \in (N_2 \cup T)^*, |x| \leq 2$ , is done in four steps, using the next rules:

1.  $[c_0 \rightarrow c_1 c_2 c_3]_X$ ,
2.  $[O]_X \rightarrow [\{c_1\}]_0 [O - \{c_1\}]_{X_m}$ ,
3.  $[cA \rightarrow cx]_{X_m}$ ,  
 $[c_3 \rightarrow c_0 c_4]_{X_m}$ ,  
 $[cd \rightarrow c\#]_{X_m}$ ,  
 $[O]_{X_m} \rightarrow [\{c_2\}]_0 [O - \{c_2\}]_{X'_m}$ ,
4.  $[O]_{X'_m} \rightarrow [\{c_4\}]_0 [O - \{c_4\}]_Y$ .

In the membrane labeled with the nonterminal  $X$  we evolve  $c_0$  into the auxiliary objects  $c_1, c_2, c_3$ , then, using  $c_1$  we separate the objects, also guessing the matrix  $m$  to simulate. All relevant objects are placed in the membrane with label  $X_m$ . In step 3, in this membrane we both simulate the second rule of the matrix (if no copy of  $A$  is present, hence the rule  $[cA \rightarrow cx]_{X_m}$  cannot be used, then the trap object is introduced by the rule  $[cd \rightarrow c\#]_{X_m}$ , hence the simulation of the matrix should be complete) and evolve  $c_3$  to  $c_0$  and  $c_4$ . At the same time, the membrane is separated again, with all relevant objects placed in a membrane with label  $X'_m$ . In the fourth step, this last membrane is separated again, by using the “separator”  $c_4$ , and thus the membrane with label  $Y$  is introduced; it contains again the object  $c_0$ , hence we return to a configuration as that we have started with.

The simulation of a matrix  $m : (X \rightarrow Y, A \rightarrow \#)$ , with  $X, Y \in N_1$  and  $A \in N_2$ , is done in a very similar way, using again four steps and the same rules as above (of course, with  $X_m, X'_m$  associated with the new matrix), the only differences being that (i) the rule  $[cA \rightarrow cx]_{X_m}$  used in step 3 is now replaced with the rule  $[A \rightarrow \#]_{X_m}$ , and (ii) the rule  $[cd \rightarrow c\#]_{X_m}$  is no longer provided. In this way, if any copy of  $A$  appears in membrane  $X_m$ , then the trap object is introduced and the computation never halts. If  $A$  is not present, then both  $c$  and  $d$  waits unused, then the simulation of the rule  $X \rightarrow Y$  of the matrix is completed.

These procedures can be iterated.

We also consider the following rules:

5.  $[A \rightarrow \#]_f$ , for all  $A \in N_2$ ,
6.  $[\# \rightarrow \#]_h$ , for all  $h \in H$ ,
7.  $[a]_f \rightarrow [ ]_f a$ , for all  $a \in T$ ,
8.  $[a]_1 \rightarrow [ ]_1 a$ , for all  $a \in T$ .

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

*Remark 1.* 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  $(cat_1)$  and  $(h'_0)$  are sufficient to reach universality for membrane systems with internal output.

## 5 Final Remarks

This note shows that using membrane separation to obtain exponential workspace in a linear time, SAT problem can be solved in linear time in a uniform and confluent way by active P systems with membrane separation rules. In general, it is still open whether any semi-uniform solution to a problem can be modified to a uniform solution. By a direct proof, the universality related to separation rule is reached. In [2] by using division to change the labels of the membranes, we also have  $PsOP(a_0, c_0, e'_0) = PsRE$ . But it remains open using P systems with separation rules to simulate P systems with division rules.

**Acknowledgements.** The authors acknowledge IST-2001-32008 project "MolCoNet". The first author is also supported by grant DGU-SB2001-0092 from Spanish Ministry of Education, Culture, and Sport, National Natural Science Foundation of China (Grant No. 60373089), and Huazhong University of Science and Technology Foundation. The second author acknowledges also the Moldovan Research and Development Association (MRDA) and the U.S. Civilian Research and Development Foundation (CRDF), Award No. MM2-3034 for providing a challenging and fruitful framework for cooperation. The third author acknowledges The State Training Fund of the Ministry of Science, Technology, Education, and Culture of Mongolia.

## References

1. Alhazov A, Ishdorj T.-O (2004) Membrane Operations in P Systems with Active Membranes. In: Păun Gh, Riscos-Núñez A, Romero-Jiménez A, Sancho-Capparini F (eds.) *Second Brainstorming Week on Membrane Computing*, Sevilla, 2-7 February 2004, Research Group in Natural Computing **TR 01/2004**, University of Seville, 37-44
2. Alhazov A, Pan L, Păun Gh (in press) Trading Polarizations for Labels in P Systems with Active Membranes. *Acta Informaticae*
3. Dassow J, Păun Gh (1989) *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin
4. Păun Gh (2000,1998), Computing with Membranes. *Journal of Computer and System Sciences*, **61**, 1: 108–143, and TUCS Research Report **208**, (<http://www.tucs.fi>)
5. Păun Gh (2002) *Membrane Computing: An Introduction*, Springer, Heidelberg
6. Pérez Jiménez MJ, Romero Jiménez A, Sancho Caparrini F (2003) Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2** (3), 265-285