Further Remarks on P Systems with Active Membranes, Separation, Merging, and Release Rules

Linqiang PAN[1,3], **Artiom ALHAZOV**[2,3], **Tseren-Onolt ISHDORJ**[3]

[1]Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, People's Republic of China
E-mail: lqpan@mail.hust.edu.cn

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

[3]Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: {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, Ts. 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, (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.disco.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 Ψ_V.

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

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

   (a_0) $[\ a \to v]_h$, for $h \in H, a \in O, v \in O^*$
   (object evolution rules, associated with membranes and depending on the label, 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) $a[\ \ ]_h \to [\ b]_h$, for $h \in H, a, b \in O$
   (communication rules; an object is introduced in the membrane during this process);

   (c_0) $[\ a\ ]_h \to [\ \ ]_h b$, for $h \in H, a, b \in O$
   (communication rules; an objects sent out of the membrane during this process);

(d_0) $[\ a\ ]_h \to b$, for $h \in H, a, b \in O$
(dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

(e_0) $[\ a]_h \to [\ b]_h[\ c]_h$, for $h \in H, a, b, c \in O$
(division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label; the object specified in the rule is replaced in the two new membranes by possibly new objects);

(g_0) $[\ ]_h[\ ]_h \to [\ ]_h$, 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) $[\ O]_h \to [\ U]_h[\ O - U]_h$, 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 $U - O$ are placed in the other membrane);

(i_0) $[\ [\ O]_h]_h \to [\ ]_h O$, 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 $(b_0), \cdots, (i_0)$ 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:

(g_0') $[\ ]_{h_1}[\ ]_{h_2} \to [\ ]_{h_3}$, for $h_1, h_2, h_3 \in H$.

(h_0') $[\ O]_{h_1} \to [\ U]_{h_2}[\ O - U]_{h_3}$, for $h_1, h_2, h_3 \in\ H, U \subset O$.

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 Π_n, $n \in \mathbb{N}$, associated with the sizes n of instances $X(n)$ of the problem, such that the system Π_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 3.1 *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 a confluent way.*

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

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

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

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

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

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

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:

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

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

Generation phase:

G1. $[\,O]_1 \to [\,U]_1[\,O - U]_1$,
$U = \{x'_{i,t,j}, \bar{x}'_{i,t,j} \mid 1 \le i \le m, 0 \le t \le j-1, 1 \le j \le n\}$
$\cup \{c'_{i,j} \mid 1 \le i \le m, 1 \le j \le n\} \cup \{d'_i \mid 1 \le i \le n+1\}$.

G2. $[\ x_{i,t,j} \to x_{i,t-1,j}x'_{i,t-1,j}]_1$,
$[\ \bar{x}_{i,t,j} \to \bar{x}_{i,t-1,j}\bar{x}'_{i,j-1,j}]_1$,
$[\ x'_{i,t,j} \to x_{i,t-1,j}x'_{i,t-1,j}]_1$,
$[\ \bar{x}'_{i,t,j} \to \bar{x}_{i,t-1,j}\bar{x}'_{i,t-1,j}]_1$, $1 \leq i \leq m$, $1 \leq t \leq j$, $1 \leq j \leq n$.

G3. $[\ x_{i,0,j} \to c_{i,j}c'_{i,j}]_1$,
$[\ \bar{x}_{i,0,j} \to \lambda]_1$,
$[\ \bar{x}'_{i,0,j} \to c_{i,j}c'_{i,j}]_1$,
$[\ x'_{i,0,j} \to \lambda]_1$, $1 \leq i \leq m$, $1 \leq j \leq n$.

G4. $[\ c_{i,j} \to c_{i,j+1}c'_{i,j+1}]_1$,
$[\ c'_{i,j} \to c_{i,j+1}c'_{i,j+1}]_1$, $1 \leq i \leq m$, $1 \leq j \leq n-1$.

G5. $[\ c_{i,n} \to c_{i,n+1}]_1$,
$[\ c'_{i,n} \to c_{i,n+1}]_1$, $1 \leq i \leq m$.

G6. $[\ d_i \to d_{i+1}d'_{i+1}]_1$, $0 \leq i \leq n$.
$[\ d'_i \to d_{i+1}d'_{i+1}]_1$, $1 \leq i \leq n$.
$[\ d_{n+1} \to d_{n+2}]_1$.
$[\ d'_{n+1} \to d_{n+2}]_1$.

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 $x_1, x_2, ..., x_n$. 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 $x_{i,j,j} = \mathit{true}$ 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 $x_j = \mathit{false}$ and is erased if $x_j = \mathit{true}$. Note that at step $n+2$, each membrane with label 1 has the object d_{n+2}.

Checking phase:

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

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. $[\ d_{n+2}]_{m+1} \to [\ ]_{m+1}d_{n+2}$.

C3. $[\ d_{n+2} \to tt]_s$.

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

C4. $t[\ ]_0 \to [\ t]_0$.

C5. $t[\ ]_{m+1} \to [\ t]_{m+1}$.

C6. $[\ O]_0 \to [\ U']_{m+1}[\ O - U']_{m+2}$,
$U' = \{t\}$.

C7. $[\ d_i \to d_{i+1}]_0$, $0 \le i \le n+m+5$.

C8. $[\ d_{n+m+6} \to d_{n+m+7}]_{m+2}$.

At step $n+m+5$, one copy of t enters the membrane with label 0, and (suppose β has s solutions, $1 \le s \le 2^n$) 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 β has no solution, then no object enters membrane labeled 0 and rule C6 is not applied.

Output phase:

O1. $[\ t \to \lambda]_{m+1}$.

O2. $[\ d_{n+m+7}]_0 \to [\ ]_0\texttt{no}$.

O3. $[\ d_{n+m+7}]_{m+2} \to [\ ]_{m+2}\texttt{yes}$.

O4. $[\ \texttt{no}]_s \to [\ ]_s\texttt{no}$.

O5. $[\ \texttt{yes}]_s \to [\ ]_s\texttt{yes}$.

If β 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 β 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. If β 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). □

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 3.1 and the proofs in Theorem 2 and 3 in [1]. So, here we omit the proofs.

Theorem 3.2 *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 3.3 *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 $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, 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 $(A_1 \to x_1, \ldots, A_n \to x_n)$ 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 $A_i \to x_i$ 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 $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. 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. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
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 $x \in T^*, |x| \leq 2$.

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 [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 Π; 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 4.1 $PsOP(a_0, c_0, h'_0) = PsRE$.

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 $X \to \lambda$ from matrices of type 4 by $X \to f$, where f is a new symbol.

We construct the P system of degree 2

$$\begin{aligned}
\Pi &= (O, H, [\ [\ ]_{X_{init}}]_1, w_1 = \lambda, w_{X_{init}} = c_0 A_{init}, R),\\
O &= T \cup N_2 \cup \{A_m \mid A \in N_2, m \in B_1\}\\
&\cup \{d_m, d'_m \mid m \in B\} \cup \{c_m \mid m \in B_2\} \cup \{c_0, \#, \lambda\},\\
H &= N_1 \cup \{X_m \mid X \in N_1, m \in B\} \cup \{0, 1, f\},
\end{aligned}$$

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 Π works.

The simulation of a matrix $m : (X \to Y, A \to x)$, 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 two steps, using the next rules:

1. $[\ A \to A_m d_m]_X$,
   $[\ O]_X \to [\ U]_0 [\ O - U]_{Y_m}$, $U = \{d_m\}$,
   $[\ c_0 \to c_n d_n]_X$, $n \in B_2$,

2. $[\ A_m \to x d'_m]_{Y_m}$,
   $[\ O]_{Y_m} \to [\ U]_0 [\ O - U]_Y$, $U = \{d'_m\}$,
   $[\ c_n \to c_0]_{Y_m}$, $n \in B_2$,
   $[\ d_n \to \lambda]_{Y_m}$, $n \in B_2$.

The first rule of the matrix is simulated by the change of the label of the inner membrane by separation rule (the "dummy" objects d_m, d'_m, and membrane 0 play no further role), and the correctness of this operation is obvious (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 two steps, using the next rules:

3. $[\ c_0 \to c_m d_m]_X$,
   $[\ O]_X \to [\ U]_0 [\ O - U]_{Y_m}$, $U = \{d_m\}$,
   $[\ A \to A_n d_n]_X$, $n \in B_1$,

4. $[\ c_m \to c_0 d'_m]_{Y_m}$,
   $[\ O]_{Y_m} \to [\ U]_0 [\ O - U]_Y$, $U = \{d'_m\}$,
   $[\ A_n \to \#]_{Y_m}$, $n \in B_1$,
   $[\ A \to \#]_{Y_m}$,
   $[\ d_n \to \lambda]_{Y_m}$, $n \in B_1$.

While the membrane with label X is separated, if any copy of A is present, then it can remain the same at this step (i.e., not evolving) and in the next step evolve to the trap-object # (this is the case if A does not appear in the matrices of type 2 and 4), or it evolves to $A_n d_n$, in the next step the object A_n introduces the trap-object # and the computation never stops. If no A is present, then the objects c_m 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:

5. $[\ A \to \#]_f$, for all $A \in N_2$,

6. $[\ \# \to \#]_h$, for all $h \in H$,

7. $[\ a]_f \to [\ ]_f a$, for all $a \in T$,

8. $[\ a]_1 \to [\ ]_1 a$, for all $a \in T$.

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

Remark 4.1 *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* (a_0) *and* (h'_0) *are sufficient to reach universality for membrane systems with internal output.*

5 Final Remark

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 (he is also the corresponding 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] A. Alhazov, Ts. Ishdorj, Membrane Operations in P Systems with Active Membranes, in the present volume.

[2] A. Alhazov, L. Pan, Gh. Păun, Trading Polarizations for Labels in P Systems with Active Membranes, submitted 2003.

[3] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

[4] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and TUCS Research Report 208, 1998 (`http://www.tucs.fi`).

[5] Gh. Păun, *Membrane Computing: An Introduction*, Springer, Heidelberg, 2002.

[6] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity Classes in Models of Cellular Computing with Membranes, *Natural Computing*, 2(3) (2003), 265–285.