International Journal of Computers, Communications & Control
Vol. I (2006), No. 1, pp. 25-32

One More Universality Result for P Systems with Objects on Membranes

Gheorghe Păun

Abstract: We continue here the attempt to bridge brane calculi with membrane computing, following the investigation started in [2]. Specifically, we consider P systems with objects placed *on* membranes, and processed by membrane operations. The operations used in this paper are membrane *creation* (*cre*), and membrane *dissolution* (*dis*), defined in a way which reminds the operations *pino, exo* from a brane calculus from [1]. For P systems based on these operations we prove the universality, for one of the two possible variants of the operations; for the other variant the problem remains open.
Keywords: Membrane computing, Brane calculi, Matrix grammar, Universality

1 Introduction

This paper is a direct continuation of [2], where a first step was made to bridge membrane computing [4], [5], [6] and brane calculi [1]. The main point of this effort is to define P systems which work with multisets of objects placed *on* the membranes rather than inside the compartments defined by membranes, and to process these multisets by means of operations with membranes rather than by multiset rewriting rules acting only on objects. The operations *pino, exo, mate, drip* were formalized in [2] as membrane computing rules, and used in defining P systems based on them. The universality of *mate, drip* operations was proved in [2] (for systems using simultaneously at any step of a computation at most eleven membranes). We give here an universality result for other two operations, membrane creation (*cre*), and membrane dissolution (*dis*), which have the same syntax as *pino, exo* operations, but a different interpretation in what concerns the contents of the handled membranes – details can be found in Section 3 below. Actually, as it was the case in [2] with *pino, exo*, we have two variants of each of the operations *cre, dis*. For one of these variants, we prove the Turing completeness, while the case of the other variant remains open (we believe that a similar result holds true).

2 Prerequisites

All notions of formal language theory we use are elementary and standard, and can be found in any basic monograph of formal language theory. For the sake of completeness, we introduce below only the notion of matrix grammars with appearance checking – after specifying that by *RE* we denote the family of recursively enumerable languages, and by *PsRE* the family of Parikh images of languages from *RE* (the Parikh mapping associated with an alphabet V is denoted by Ψ_V).

A matrix grammars with appearance checking [3] is a construct $G = (N,T,S,M,F)$, where N,T are disjoint alphabets (of non-terminals and terminals, respectively), $S \in N$ (axiom), M is a finite set of *matrices*, that is sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$, and F is a set of occurrences of rules in the matrices of M.

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 (1) $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or (2) $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (If applicable, the rules from F should be applied, but if they cannot be applied, then we may skip them. That is why the rules from F are said to be applied in the *appearance checking* mode.) If $F = \emptyset$, then the grammar is said to be without appearance checking.

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$, where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$.

The family of languages of this form is denoted by MAT_{ac}; it is known that $MAT_{ac} = RE$.

We say that a matrix grammar with appearance checking $G = (N,T,S,M,F)$ is in the *Z-binary normal form* if $N = N_1 \cup N_2 \cup \{S,Z,\#\}$, 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. $(X \to Y, A \to w)$, with $X,Y \in N_1, A \in N_2, w \in (N_2 \cup T)^*, |w| \leq 2$,

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

4. $(Z \to \lambda)$.

Moreover, there is only one matrix of type 1, F consists exactly of all rules $A \to \#$ appearing in matrices of type 3, and, if a sentential form generated by G contains the symbol Z, then it is of the form Zw, for some $w \in (T \cup \{\#\})^*$ (that is, the appearance of Z makes sure that, except for Z, all symbols are either terminal or the trap-symbol #). The matrix of type 4 is used only once, in the last step of a derivation.

For each language $L \in RE$ there is a matrix grammar with appearance checking G in the Z-binary normal form such that $L = L(G)$.

As usual, we represent multisets over an alphabet V by strings over V, with the obvious observation that all permutations of a string represent the same multiset.

3 P Systems Using the Cre/Dis Operations

We start by recalling from [2] the formalization of the operations *pino, exo* in terms of membrane computing.

A membrane is represented, as usual, by a pair of square brackets, [ ], but we associate here with membranes *multisets of object* (corresponding to the proteins embedded in the real membranes). A membrane having associated a multiset u (represented by a string) is written in the form $[\ ]_u$; we also use to say that the membrane is *marked* with the multiset u.

The following four operations were defined in [2]:

$$\begin{aligned}
pino_i &: [\ ]_{uav} \to [[\ ]_{ux}]_v, && (1)\\
exo_i &: [[\ ]_{ua}]_v \to [\ ]_{uxv}, && (2)\\
pino_e &: [\ ]_{uav} \to [[\ ]_v]_{ux}, && (3)\\
exo_e &: [[\ ]_u]_{av} \to [\ ]_{uxv}. && (4)
\end{aligned}$$

in all cases with $a \in V$, $u, x \in V^*$, $v \in V^+$, with $ux \in V^+$ for *pino* rules, where V is a given alphabet of objects.

In each case, multisets of proteins are transferred from input membranes to output membranes as indicated in the rules, with protein a evolved into the multisets x (which can be empty). The subscripts i and e stand for "internal" and "external", respectively, pointing to the "main" membrane of the operation in each case.

It is important to note that the multisets u, v and the protein a marking the left hand membranes of these rules correspond to the multisets u, v, x from the right hand side of the rules; specifically, the multiset uxv resulting when applying the rule is precisely split into ux and v, with these two multisets assigned to the two new membranes.

The rules are applied as follows. Assume that we have a membrane $[\ ]_{zuav}$, for $a \in V, u, v, z \in V^*$. By a $pino_i$ rule as in (1), we obtain any one of the pairs of membranes $[\ [\ ]_{z_1 ux}\ ]_{z_2 v}$ such that $z = z_1 z_2,\ z_1, z_2 \in V^*$, and by a $pino_e$ rule as in (3), we obtain any one of the pairs of membranes $[\ [\ ]_{z_1 v}\ ]_{z_2 ux}$ such that $z = z_1 z_2,\ z_1, z_2 \in V^*$.

In the case of the two *exo* operations, the result is uniquely determined. From a pair of membranes $[\ [\ ]_{z_1 ua}]_{z_2 v}$, by an exo_i rule as in (2) we obtain the membrane $[\ ]_{z_1 z_2 uxv}$, and from $[\ [\ ]_{z_1 u}]_{z_2 av}$, by an exo_e rule as in (4) we obtain the same membrane $[\ ]_{z_1 z_2 uxv}$.

The contents of membranes involved in these operations is transferred from the input membranes to the output membranes in the same way as in brane calculi (**P,Q** represent here the possible contents of the respective membranes):

$$\begin{aligned}
pino_i &: [\ \mathbf{P}\ ]_{uav} \to [[\ ]_{ux}\ \mathbf{P}\ ]_v,\\
exo_i &: [[\ \mathbf{P}\ ]_{ua}\ \mathbf{Q}\ ]_v \to \mathbf{P}\ [\ \mathbf{Q}\ ]_{uxv},\\
pino_e &: [\ \mathbf{P}\ ]_{uav} \to [[\ ]_v\ \mathbf{P}\ ]_{ux},\\
exo_e &: [[\ \mathbf{P}\ ]_u\ \mathbf{Q}\ ]_{av} \to \mathbf{P}\ [\ \mathbf{Q}\ ]_{uxv}.
\end{aligned}$$

Here we change the interpretation of these rules, as suggested below (because the new semantics do not correspond to the operations *pino, exo*, we change the name of operations to *cre, dis*, for "membrane creation" and

"membrane dissolution"):

$$\begin{aligned} cre_i &: [\ \mathbf{P}\ ]_{uav} \to [\,[\ \mathbf{P}\ ]_{ux}\,]_v, \\ dis_i &: [\,[\ \mathbf{P}\ ]_{ua}\ \mathbf{Q}\,]_v \to [\ \mathbf{P}\ \mathbf{Q}\ ]_{uxv}, \\ cre_e &: [\ \mathbf{P}\ ]_{uav} \to [\,[\ \mathbf{P}\ ]_v\,]_{ux}, \\ dis_e &: [\,[\ \mathbf{P}\ ]_u\ \mathbf{Q}\,]_{av} \to [\ \mathbf{P}\ \mathbf{Q}\ ]_{uxv}. \end{aligned}$$

That is, when a membrane is created inside an existing membrane, the new membrane contains all previously existing membranes, and while dissolving a membrane, its contents remains inside the membrane where it was placed before the operation. The interpretation of the latter operation is rather similar to the usual dissolution operation in membrane computing, while the membrane creation is understood as doubling the existing membrane, with a distribution of the multiset marking the initial membrane to the two new membranes.

Using rules as defined above, we can define a P system as

$$\Pi = (A, \mu, u_1, \ldots, u_m, R),$$

where:

1. A is an alphabet (finite, non-empty) of objects;
2. μ is a membrane structure with $m \geq 2$ membranes;
3. $u_1, \ldots, u_m$ are multisets of objects (represented by strings over A) bound to the m membranes of μ at the beginning of the computation; the skin membrane is marked with $u_1 = \lambda$;
4. R is a finite set of *cre, dis* rules, of the forms specified above, with the objects from the set A.

For a rule of any type, with u, a, v as above, $|uav|$ is called the *weight* of the rule.

In what follows, the skin membrane plays no role in the computation, *no rule can be applied to it*. Also, we stress the fact that there is no object in the compartments of μ; a membrane can contain other membranes inside, but in-between membranes there is nothing.

When using any rule of any type, we say that the membranes from its left hand side are *involved* in the rule; they all are "consumed", and the membranes from the right hand side of the rule are produced instead. Similarly, the object a specified in the left hand side of rules is "consumed", and it is replaced by the multiset x.

The evolution of the system is defined in the standard way used in membrane computing, with the rules applied in the non-deterministic maximally parallel manner, with each membrane involved in at most one rule. Thus, the parallelism is maximal at the level of membranes – each membrane which can evolve has to do it – but each multiset of objects evolves in a sequential manner, as only one rule can act on any multiset in a transition step. More precise details can be found in [2]. A computation which starts from the initial configuration is *successful* if (i) it halts, that is, it reaches a configuration where no rule can be applied, and (ii) in the halting configuration there are only two membranes, the skin (marked with λ) and an inner one. The *result* of a successful computation is the vector of multiplicities of objects which mark the inner membrane in the halting configuration. The set of all vectors computed in this way by Π is denoted by $Ps(\Pi)$.

The family of all sets of vectors $Ps(\Pi)$ computed by P systems Π using at any moment during a computation at most m membranes, and cre_i, dis_i rules of weight at most p, q, respectively, is denoted by $PsOP_m(cre_p, dis_q)$. When one of the parameters m, p, q is not bounded we replace it with $*$.

We end this section by pointing out some relations which follow directly from the definitions (and from Turing-Church thesis).

Lemma 1. (i) $PsOP_m(cre_p, dis_q) \subseteq PsOP_{m'}(cre_{p'}, dis_{q'})$, *for all* $m \leq m', p \leq p', q \leq q'$.
(ii) $PsOP_*(cre_*, dis_*) \subseteq PsRE$.

We also recall the main result from [2]: $PsOP_{11}(mate_5, drip_5) = PsRE$ (the notation is self-explanatory).

4 Universality for the Cre/Dis Operations

In the case of *cre, dis* operations as defined above, we cannot generate vectors of norm 0 or 1: in each rule $[\ ]_{uav} \to [[\ ]_{ux}]_v$, $[[\ ]_{ua}]_v \to [\ ]_{uxv}$ (necessary in the last step of any computation in order to get only one internal membrane) we have imposed to have $|uxv| \geq 2$. That is why the universality below is obtained modulo vectors of the form $(0,\ldots,0)$ and $(0,\ldots,0,1,0,\ldots,0)$. We denote by $Ps'RE$ and $Ps'OP_m(cre_p, dis_q)$ the sets of vectors from $PsRE$ and $PsOP_m(cre_p, dis_q)$ having the sum of elements greater than or equal to 2.

Theorem 2. *$Ps'RE = Ps'OP_m(cre_p, dis_q)$ for all $m \geq 7, p \geq 4$, and $q \geq 4$.*

Proof. Let us consider a language $L \in RE = MAT_{ac}$, $L \subseteq V^2V^*$, for an alphabet V with n symbols. We write this language in the form

$$L = \bigcup_{a,b \in V} \{ab\}\partial^l_{ab}(L).$$

Let $G_{ab} = (N_{ab}, V, S_{ab}, M_{ab}, F_{ab})$ be a matrix grammar with appearance checking such that $L(G_{ab}) = \partial^l_{ab}(L)$, for $a, b \in V$. We consider these grammars G_{ab} in the Z-normal form, with the notations from Section 2 (hence $N_{ab} = N_{ab,1} \cup N_{ab,2} \cup \{S_{ab}, Z_{ab}, \#\}$), and we construct the matrix grammar $G = (N, V, S, M, F)$ with

$$\begin{aligned}
N &= N_1 \cup N_2 \cup \{Z_{ab} \mid a,b \in V\} \cup \{S, \#\},\\
N_1 &= \bigcup_{a,b \in V} N_{ab,1},\\
N_2 &= \bigcup_{a,b \in V} N_{ab,2},\\
M &= \{(S \to XA) \mid \text{for } (S_{ab} \to XA) \in M_{ab},\ a,b \in V\}\\
&\cup \{(X \to Y, A \to w) \mid \text{for } (X \to Y, A \to w) \in M_{ab}, a,b \in V\}\\
&\cup \{(Z_{ab} \to ab) \mid \text{for } (Z \to \lambda) \in M_{ab}, a,b \in V\}.
\end{aligned}$$

Obviously, $L(G) = L$.

We assume that all two-rules matrices from M are injectively labeled, in the form $m_l : (X \to Y, A \to x)$, $l \in Lab$, for a set of labels Lab.

Starting from the grammar G we now construct a P system

$$\Pi = (A, [\ [\ ]\ ], \lambda, S_1S_2, R),$$

with the alphabet

$$\begin{aligned}
A &= \{Y, Y', Y'', Y''', Y^{iv}, Y^v, Y^{vi}, Y^{vii}, Y^{viii}, Y^{ix}, Y^x \mid Y \in N_1\}\\
&\cup \{\alpha, \alpha', \alpha'' \mid \alpha \in N_2 \cup V\}\\
&\cup \{\bar{A} \mid A \in N_2\}\\
&\cup \{Z_{ab}, Z'_{ab}, Z''_{ab}, Z'''_{ab} \mid a,b \in V\}\\
&\cup \{E, H, H', S_1, S_2, S_3, c_1, \ldots, c_{11}, c_0, c'_0, c''_0, c'_3, c''_3, d_1, d_2, d'_1, d'_2, f', f'', \#\},
\end{aligned}$$

and the rules from the set R as constructed below.

Any computation starts from the configuration $[\ [\ ]_{S_1S_2}]_\lambda$, by using the following rules:

$$\begin{aligned}
&\text{Step 1:} && [\ ]_{S_1S_2} \to [\ [\ ]_X\ ]_{S_2},\\
&\text{Step 2:} && [\ [\ ]_X\ ]_{S_2} \to [\ ]_{Xc_0d_1S_2},\\
&\text{Step 3:} && [\ ]_{XS_2c_0d_1} \to [\ [\ ]_{XS_3}\ ]_{c_0d_1},\\
&\text{Step 4:} && [\ ]_{S_3X} \to [\ [\ ]_{E\bar{A}}\ ]_X,\ [\ ]_{c_0d_1} \to [\ [\ ]_{c'_0}\ ]_{d_1},\\
&\text{Step 5:} && [\ [\ ]_{E\bar{A}}\ ]_X \to [\ ]_{E\bar{A}X},\ [\ [\ ]_{c'_0}\ ]_{d_1} \to [\ ]_{c''_0d_1},\\
&\text{Step 6:} && [\ ]_{X\bar{A}E} \to [\ [\ ]_{XA}\ ]_E,\ [\ ]_{c''_0d_1} \to [\ [\ ]_{c_1}\ ]_{d_1},
\end{aligned}$$

for each matrix $(S_{ab} \to XA) \in M_{ab}$, for $a, b \in V$.

The rules are used as indicated in the table above, with two rules simultaneously applied in steps 4, 5, 6. The only possible branching is in step 3, when instead of the rule $[\,]_{XS_2c_0d_1} \to [\,[\,]_{XS_3}\,]_{c_0d_1}$, we can also use the rule $[\,]_{c_0d_1} \to [\,[\,]_{c_0'}\,]_{d_1}$. In this way we obtain the membranes $[\,[\,]_{c_0'}\,]_{d_1}$, with XS_2 distributed among them. Because S_3 will be never introduced, we continue only with rules which process membranes marked with c_i and d_1, namely, the rules from the third column of Table 1; in this way, the computation will never stop, both because we can return again and again to a pair of membranes of the form $[\,[\,]_{c_1}\,]_{d_1}$, and because pairs of membranes marked with c_3' will appear and introduce trap objects/membranes – see also below.

The evolution of the membrane structure is indicated in Figure 1.

Initial	$[\,[\,]_{S_1S_2}\,]_\lambda$
Step 1	$[\,[\,[\,]_X\,]_{S_2}\,]_\lambda$
Step 2	$[\,[\,]_{Xc_0d_1S_2}\,]_\lambda$
Step 3	$[\,[\,[\,]_{XS_3}\,]_{c_0d_1}\,]_\lambda$
Step 4	$[\,[\,[\,[\,[\,]_{E\bar{A}}\,]_X\,]_{c_0'}\,]_{d_1}\,]_\lambda$
Step 5	$[\,[\,[\,]_{E\bar{A}X}\,]_{c_0''d_1}\,]_\lambda$
Step 6	$[\,[\,[\,[\,[\,]_{XA}\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$

Figure 1: The evolution of membranes at the beginning of computations.

Thus, we end with a configuration of the form $[\,[\,[\,[\,[\,]_{XA}\,]_E\,]_{c_1}\,]_{d_1}]_\lambda$.

The rules for simulating the two-rules matrices from M are indicated in Table 1; by w' we denote here the string obtained from w by priming one symbol; if $w = \lambda$, then $w' = f'$, hence $\alpha' = f', \alpha'' = f''$ and, in row 6, $\alpha = \lambda$.

Step	$m_l : (X \to Y, A \to w)$	$m_l : (X \to Y, B \to \#)$	
1	$[\,[\,]_X\,]_E \to [\,]_{X_lE}$	$[\,[\,]_X\,]_E \to [\,]_{X_lE}$	$[\,[\,]_{c_1}\,]_{d_1} \to [\,]_{c_2c_3d_1}$
2	$[\,]_{AEX_l} \to [\,[\,]_{w'}\,]_{EX_l}$	$[\,]_{X_lBE} \to [[\,]_{X_l\#\#}\,]_E$	$[\,]_{c_3c_2d_1} \to [[\,]_{c_3'}]_{c_2d_1}$
3	$[\,[\,]_{EX_l}\,]_{c_3'} \to [\,]_{EY'c_3'}$	$[\,[\,]_{X_lE}\,]_{c_3'} \to [\,]_{Y^{vi}HEc_3'}$	$[\,]_{c_2d_1} \to [\,[\,]_{c_4}\,]_{d_1}$
4	$[\,]_{c_3'Y'E} \to [\,[\,]_{c_3'Y''}\,]_E$	$[\,]_{Y^{vi}c_3'EH} \to [[\,]_{c_3'Y^{vii}}\,]_{EH}$	$[\,[\,]_{c_4}\,]_{d_1} \to [\,]_{c_5d_1}$
5	$[[\,]_{\alpha'}\,]_{c_3'Y''} \to [\,]_{\alpha''c_3'Y''}$	$[\,]_{Y^{vii}c_3'} \to [\,[\,]_{Y^{viii}}\,]_{c_3'}$ $[\,]_{HE} \to [\,[\,]_{H'}\,]_E$	$[\,]_{c_5d_1} \to [\,[\,]_{c_6}\,]_{d_1}$
6	$[[\,]_{\alpha''c_3'Y''}]_E \to [\,]_{\alpha c_3'Y''E}$	$[[\,]_{c_3'}\,]_{H'} \to [\,]_{c_3''H'}$	$[\,[\,]_{c_6}\,]_{d_1} \to [\,]_{c_7d_1}$
7	$[\,]_{c_3'Y''E} \to [\,[\,]_{c_3'Y'''}\,]_E$	$[[\,]_{Y^{viii}}\,]_{c_3''H'} \to [\,]_{Y^{ix}c_3''H'}$	$[\,]_{c_7d_1} \to [\,[\,]_{c_8}\,]_{d_1}$
8	$[\,[\,]_{c_3'Y'''}\,]_E \to [\,]_{Y'''E}$	$[[\,]_{Y^{ix}c_3''H'}]_E \to [\,]_{Y^{ix}H'E}$	$[\,[\,]_{c_8}\,]_{d_1} \to [\,]_{c_9d_1}$
9	$[\,]_{Y'''E} \to [\,[\,]_{Y^{iv}}\,]_E$	$[\,]_{Y^{ix}H'E} \to [\,]_{Y^x}\,]_E$	$[\,]_{c_9d_1} \to [\,[\,]_{c_{10}}\,]_{d_1}$
10	$[\,[\,]_{Y^{iv}}\,]_E \to [\,]_{Y^vE}$	$[\,[\,]_{Y^x}\,]_E \to [\,]_{Y^xE}$	$[\,[\,]_{c_{10}}\,]_{d_1} \to [\,]_{c_{11}d_1}$
11	$[\,]_{Y^vE} \to [\,[\,]_Y\,]_E$	$[\,]_{Y^xE} \to [\,[\,]_Y\,]_E$	$[\,]_{c_{11}d_1} \to [\,[\,]_{c_1}\,]_{d_1}$

Table 1: Rules for simulating two-rules matrices.

We also consider the rules

$$[\,]_{X_lE} \to [\,[\,]_{\#\#}\,]_E, \text{ for each matrix } m_l : (X \to Y, A \to w),$$
$$[\,[\,]_{H'}\,]_E \to [\,]_{\#\#E},$$
$$[\,]_{\#\#} \to [\,[\,]_\#\,]_\#,$$
$$[\,[\,]_\#\,]_\# \to [\,]_{\#\#}.$$

The simulation of matrices in G is performed by modifying the marking of the central membranes, those emerging from the initial membranes with markings XA and E, with these operations being assisted by the two membranes with markings c_1 and d_1 and their successors, which are external to the central membranes where the sentential form of G is produced. Always during the computation, the membranes remain embedded one in another, in a linear manner, never having two membranes on the same level (here stands the essential difference between the interpretation of the *cre*, *dis* operations and the interpretation of the *pino*, *exo* operations from [1], [2]).

The evolution of the membranes and of their relevant markings can be followed in Figure 2. If in the second step the rule $[\,]_{AEX_l} \to [\,[\,]_{w'}\,]_{EX_l}$ is not applicable (hence the matrix m_l cannot be applied), then the rule $[\,]_{X_lE} \to [\,[\,]_{\#\#}\,]_E$ will be applied, introducing the trap-object #, and the computation will never halt.

Starting	$[\,[\,[\,[\,[\,]_X\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$
Step 1	$[\,[\,[\,]_{X_lEA}\,]_{c_2c_3d_1}\,]_\lambda$
Step 2	$[\,[\,[\,[\,[\,]_{w'}\,]_{EX_l}\,]_{c'_3}\,]_{c_2d_1}\,]_\lambda$
Step 3	$[\,[\,[\,[\,[\,]_{\alpha'}\,]_{EY'c'_3}\,]_{c_4}\,]_{d_1}\,]_\lambda$
Step 4	$[\,[\,[\,[\,[\,]_{\alpha'}\,]_{c'_3Y''}\,]_E\,]_{c_5d_1}\,]_\lambda$
Step 5	$[\,[\,[\,[\,[\,]_{\alpha''c'_3Y''}\,]_E\,]_{c_6}\,]_{d_1}\,]_\lambda$
Step 6	$[\,[\,[\,]_{\alpha c'_3Y''E}\,]_{c_7d_1}\,]_\lambda$
Step 7	$[\,[\,[\,[\,[\,]_{c'_3Y'''}\,]_E\,]_{c_8}\,]_{d_1}\,]_\lambda$
Step 8	$[\,[\,[\,]_{Y'''E}\,]_{c_9d_1}\,]_\lambda$
Step 9	$[\,[\,[\,[\,[\,]_{Y^{iv}}\,]_E\,]_{c_{10}}\,]_{d_1}\,]_\lambda$
Step 10	$[\,[\,[\,]_{Y^vE}\,]_{c_{11}d_1}\,]_\lambda$
Step 11	$[\,[\,[\,[\,[\,]_Y\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$

Figure 2: The evolution of membranes when simulating $m_l : (X \to Y, A \to w)$.

The evolution of membranes in the case of the simulation of a matrix $m_l : (X \to Y, B \to \#)$ can be followed in Figure 3. This time, if B is present, in step 2 we have to use the rule $[\,]_{X_lBE} \to [\,[\,]_{X_i\#\#}\,]_E$, and the computation will never halt. If no copy of B is present, then the central membrane does not evolve, waiting for the membrane marked with c'_3 to be produced; this membrane can be used in the next step for evolving the central membrane.

Starting	$[\,[\,[\,[\,[\,]_X\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$
Step 1	$[\,[\,[\,]_{X_lE}\,]_{c_2c_3d_1}\,]_\lambda$
Step 2	$[\,[\,[\,[\,]_{X_lE}\,]_{c'_3}\,]_{c_2d_1}\,]_\lambda$
Step 3	$[\,[\,[\,[\,]_{Y^{vi}HEc'_3}\,]_{c_4}\,]_{d_1}\,]_\lambda$
Step 4	$[\,[\,[\,[\,]_{c'_3Y^{vii}}\,]_{EH}\,]_{c_5d_1}\,]_\lambda$
Step 5	$[\,[\,[\,[\,[\,[\,]_{Y^{viii}}\,]_{c'_3}\,]_{H'}\,]_E\,]_{c_6}\,]_{d_1}\,]_\lambda$
Step 6	$[\,[\,[\,[\,[\,]_{Y^{viii}}\,]_{c''_3H'}\,]_E\,]_{c_7d_1}\,]_\lambda$
Step 7	$[\,[\,[\,[\,[\,]_{Y^{ix}c''_3H'}\,]_E\,]_{c_8}\,]_{d_1}\,]_\lambda$
Step 8	$[\,[\,[\,]_{Y^{ix}H'E}\,]_{c_9d_1}\,]_\lambda$
Step 9	$[\,[\,[\,[\,[\,]_{Y^x}\,]_E\,]_{c_{10}}\,]_{d_1}\,]_\lambda$
Step 10	$[\,[\,[\,]_{Y^xE}\,]_{c_{11}d_1}\,]_\lambda$
Step 11	$[\,[\,[\,[\,[\,]_Y\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$

Figure 3: The evolution of membranes when simulating $m_l : (X \to Y, B \to \#)$.

Another step when we can apply a rule different from that indicated in Table 1 is step 4, when we can also use the rule $[\,]_{HE} \to [\,[\,]_{H'}\,]_E$. In this way, we pass to the configuration of membranes $[\,[\,[\,[\,]_{H'w_1}\,]_{Ew_2}\,]_{c_5d_1}\,]_\lambda$, where $w_1w_2 = Y^{vi}c'_3$. No rule can be applied to the two inner membranes other than $[\,[\,]_{H'}\,]_E \to [\,]_{\#\#E}$, and again the computation will never stop.

Therefore, the simulation of matrices in G should be done as above, and in this way we return to a configuration as that we have started with, with four membranes marked with X, E, c_1, d_1, respectively (the central membranes also having on them the symbols of the current sentential form of G which is simulated in Π).

Note that the rules used for simulating a matrix $m_l : (X \to Y, A \to w)$ cannot be mixed with the rules used for simulating a matrix $m_{l'} : (X' \to Y', A' \to \#)$, because of the injective labeling of matrices from M and because of the priming of symbols from N_1.

The process can be iterated, hence at some moment we introduce the symbol Z_{ab} identified by the symbols from N_1 used. The respective configuration is of the form: $[\,[\,[\,[\,[\,]_{Z_{ab}}\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$. The central membrane will

"swallow" all other membranes, also removing all auxiliary objects. To this aim, we use the following rules:

$$\begin{aligned}
&\text{Step 1} && [\,[\,]_{Z_{ab}}\,]_E \to [\,]_{Z'_{ab}E},\\
&\text{Step 2} && [\,[\,]_{Z'_{ab}E}\,]_{c_2c_3} \to [\,]_{Z'_{ab}bc_2c_3},\\
&\text{Step 3} && [\,]_{Z'_{ab}c_2c_3d_1} \to [\,[\,]_{Z'_{ab}}\,]_{c_3d_1},\\
&\text{Step 4} && [\,[\,]_{Z'_{ab}}\,]_{c_3d_1} \to [\,]_{Z''_{ab}c_3d_1},\\
&\text{Step 5} && [\,]_{Z''_{ab}c_3d_1} \to [\,[\,]_{Z''_{ab}}\,]_{d_1},\\
&\text{Step 6} && [\,[\,]_{Z''_{ab}}\,]_{d_1} \to [\,]_{Z'''_{ab}d_1},\\
&\text{Step 7} && [\,]_{Z'''_{ab}d_1b} \to [\,[\,]_{Z'''_{ab}}\,]_b,\\
&\text{Step 8} && [\,[\,]_{Z'''_{ab}}\,]_b \to [\,]_{ab},
\end{aligned}$$

for all $a,b \in V$. Furthermore, we consider the rules

$$\begin{aligned}
&[\,]_{Z'_{ab}E} \to [\,[\,]_{\#\#}\,]_E,\\
&[\,[\,]_{c'_3}\,]_{c'_3} \to [\,]_{\#\#c'_3},\\
&[\,]_{\#a} \to [\,[\,]_{\#\#}\,]_a, \text{ for all } a \in V.
\end{aligned}$$

The first of these rules is used in step 2 if the rule $[\,[\,]_{Z'_{ab}E}\,]_{c_2c_3} \to [\,]_{Z'_{ab}bc_2c_3}$ is not used – the objects $c_2c_3d_1$ might be used at that time by the rule $[\,]_{c_3c_2d_1} \to [\,[\,]_{c'_3}\,]_{c_2d_1}$ from Table 1. Similarly, if this last rule is used in step 3 instead of the rule $[\,]_{Z'_{ab}c_2c_3d_1} \to [\,[\,]_{Z'_{ab}}\,]_{c_3d_1}$, then a membrane marked with c'_3 is introduced, which will never be removed. In particular, after 11 steps, we introduce another membrane marked with c'_3, and then the rule $[\,[\,]_{c'_3}\,]_{c'_3} \to [\,]_{\#\#c'_3}$ is used, preventing the termination of the computation. In conclusion, the evolution of the membranes in the final stage of the computation is as indicated in Figure 4.

Starting	$[\,[\,[\,[\,[\,]_{Z_{ab}}\,]_E\,]_{c_1}\,]_{d_1}\,]_\lambda$
Step 1	$[\,[\,[\,]_{Z'_{ab}E}\,]_{c_2c_3d_1}\,]_\lambda$
Step 2	$[\,[\,]_{Z'_{ab}bc_2c_3d_1}\,]_\lambda$
Step 3	$[\,[\,[\,]_{Z'_{ab}}\,]_{c_3d_1}\,]_\lambda$
Step 4	$[\,[\,]_{Z''_{ab}c_3d_1}\,]_\lambda$
Step 5	$[\,[\,[\,]_{Z''_{ab}}\,]_{d_1}\,]_\lambda$
Step 6	$[\,[\,]_{Z'''_{ab}d_1}\,]_\lambda$
Step 7	$[\,[\,[\,]_{Z'''_{ab}}\,]_b\,]_\lambda$
Step 8	$[\,[\,]_{ab}\,]_\lambda$

Figure 4: The evolution of membranes in the end of computations.

The equality $\Psi_V(L(G)) = Ps(\Pi)$ follows from the previous explanations.

With the observation that the maximal number of membranes present in the system is seven, in step 5 from Figure 3 (during the simulation of matrices with a rule to be used in the appearance checking mode), and that the rules have the weight as specified in the theorem, we conclude the proof. □

5 Final Remarks

The case of using the operations cre_e, dis_e remains as a task for the reader, and the same with other operations from brane calculus – see also [2] for related problems. Improvements of the result in Theorem 2 are also plausible in what concerns the degree of context-sensitivity of the rules (and maybe also in what concerns the number of membranes). The same problems can be formulated for the result from [2].

As a general research topic, it remains to systematically investigate P systems with multisets of objects placed on membranes (maybe also in the compartments), processed by membrane handling operations like in brane calculi (maybe also by local multiset rewriting rules).

References

[1] L. Cardelli, Brane calculi. Interactions of biological membranes, *Proc. Computational Methods in Systems Biology, 2004*, Springer-Verlag, Berlin, to appear.

[2] L. Cardelli, Gh. Păun, An universality result for a (mem)brane calculus based on mate/drip operations, *Intern. J. Foundations of Computer Sci.*, 17, 1 (2006), 49–68.

[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 Turku Center for Computer Science–TUCS Report 208, November 1998, `www.tucs.fi`).

[5] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[6] The membrane computing web page: `http://psystems.disco.unimib.it`.

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
and
Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: george.paun@imar.ro, gpaun@us.es

Editor's note about the author:

Gheorghe PĂUN (born on December 6, 1950) graduated the Faculty of Mathematics of the Bucharest University in 1974 and got his PhD at the same faculty in 1977. He has won many scholarships, in Germany, Finland, The Netherlands, Spain, etc. Presently he is a senior researcher at the Institute of Mathematics of the Romanian Academy, Bucharest, and a Ramon y Cajal research professor at Sevilla University, Spain. Since 1997 he is a Corresponding Member of the Romanian Academy, and since 2006 a member of Academia Europaea. His main research fields are formal language theory (regulated rewriting, contextual grammars, grammar systems), automata theory, combinatorics on words, computational linguistics, DNA computing, membrane computing (this last area was initiated by him in 1998). He has (co)authored and (co)edited more than fifty books in these areas, and he has (co)authored more than 400 research papers. In the last two decades he has visited many universities from Europe, USA, Canada, Japan, also participating to many international conferences, several times as an invited speaker. He is a member of the editorial board of numerous computer science journals and professional associations.