ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII "AL.I. CUZA" DIN IAŞI (S.N.)
MATEMATICĂ, Tomul LVII, 2011, Supliment

COMPUTING BY ENDOCYTOSIS AND EXOCYTOSIS*

BY

BOGDAN AMAN and GABRIEL CIOBANU

Abstract. Mobile membranes represent a rule-based, parallel and nondeterministic computing model inspired by cells and their movements in which mobility is given by specific endocytosis and exocytosis rules. We present several results regarding the computability power of different systems of mobile membranes.

Mathematics Subject Classification 2000: 68Q05, 68Q17, 68Q45.

Key words: membrane systems, mobility, computability power.

1. Introduction

To cope with complex systems of biology, new approaches using mathematics and software tools are required. The biologists increasingly recognize that computational methods are powerful enough to model the biological processes. In this sense, "mathematics is biology's next microscope, only better"[10]. In biology, new ensemble behaviours emerge from the interactions of biological elements, and new formalisms are required to cope with these properties. In this sense, "biology is mathematics' next physics, only better"[10]. Systems biology is an emerging field which arises from the interaction of biology, mathematics and computer science.

Membrane systems (also called P systems) were introduced in [15] as a class of computing devices inspired by biology [16]. They have several applications described in [9]. Links between membrane systems and process calculi are presented in [7].

*The work of Bogdan Aman is supported by POSDRU/89/1.5/S/49944.

2. Mobile membranes

Inspired from biology, we introduced in [2] a new model of computation characterized by two features:

1. A spatial structure consisting of a hierarchy of membranes (which do not intersect) with objects associated to them. A membrane without any other membranes inside is called elementary.

2. Mobility rules describing the evolution of the structure: endocytosis (moving an elementary membrane inside a neighbouring membrane) and exocytosis (moving an elementary membrane outside the membrane where it is placed). More specific rules are given by pinocytosis (engulfing zero external membranes) and phagocytosis (engulfing one external elementary membrane).

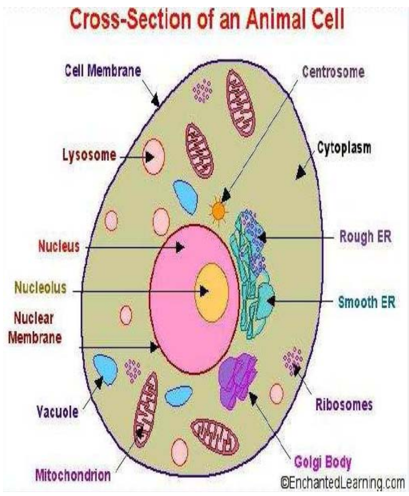

Figure 1: A biological cell is a hierarchical structure, with a flow of materials which underlies their functioning.

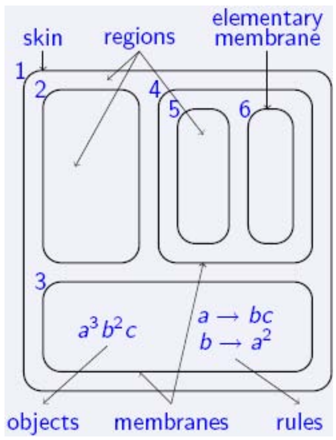

Figure 2: A membrane system is characterized by parallel application of rules, depending on the available "resources".

Definition 1. A system of mobile membranes is formally defined by $\Pi = (V, H, \mu, w_1/v_1, \ldots, w_n/v_n, R)$, where

- $n \geq 1$ represents the initial *number* of membranes;

- V is an alphabet (its elements are called *objects*);
- H is a finite set of *labels* for membranes;
- $\mu \subseteq H \times H$ describes the *membrane structure*, such that $(i, j) \in \mu$ denotes that membrane j is contained in membrane i;
- $w_1, \ldots, w_n \in V^*$ are *multisets of objects* placed inside the n regions;
- $v_1, \ldots, v_n \in V^*$ are *multisets of objects* placed on the n membranes;
- R is a finite set of *mobility rules.*

The computations are performed in the following way: starting from an initial configuration, the system evolves by applying the rules in a non-deterministic and maximally parallel manner. A rule is applicable when all the involved objects and membranes appearing in its left hand side are available. The maximally parallel way of using the rules means that in each step we apply a maximal multiset of rules, namely a multiset of rules such that no further rule can be added to the multiset. A halting configuration is reached when no rule is applicable.

3. Computability power

We define four classes of mobile membranes, and follow the research line initiated in membrane computing, namely defining membrane systems with a minimal set of ingredients that have the same power as Turing machines.

3.1. Simple mobile membranes

The evolution of systems of *simple mobile membranes* is made using the set R of rules:

$[[a]_m]_h \to [[v]_m]_h$, for $h, m \in H$, $a \in V$, $v \in V^*$; local object evolution (levol)
$[a]_m \to [v]_m$, for $m \in H$, $a \in V$, $v \in V^*$; global object evolution (gevol)
$[a]_h[\ ]_m \to [[b]_h]_m$, for $h, m \in H$, $a, b \in V$; endocytosis (exo)
$[[a]_h]_m \to [b]_h[\ ]_m$, for $h, m \in H$, $a, b \in V$; exocytosis (endo)

Turing completeness can be obtained by using systems of four mobile membranes together with global evolution, endocytosis and exocytosis rules [12]. In [2] we improve this result by decreasing the number of membranes to three and using (levol) rules instead of (gevol) rules.

Theorem 1 ([2]). *Systems of three simple mobile membranes using rules of types* (levol), (endo), (exo) *have the same computational power as a Turing machine.*

3.2. Enhanced mobile membranes

Inspired by some biological mechanisms of the immune system, we introduced in [1] the systems of *enhanced mobile membranes* that evolve using rules from the set R of rules:

$[a]_h[\ ]_m \to [[b]_h]_m$ for $h, m \in H; a, b \in V$; endocytosis (exo)

$[[a]_h]_m \to [b]_h[\ ]_m$, for $h, m \in H,\ a, b \in V$; exocytosis (endo)

$[\ ]_h[c]_m \to [[\ ]_h d]_m$ for $h, m \in H,\ c, d \in V$; enhanced endocytosis (fendo)

$[[\ ]_h c]_m \to [\ ]_h[d]_m$ for $h, m \in H,\ c, d \in V$; enhanced exocytosis (fexo)

The computational power of enhanced mobile membranes using these mobility rules was studied in [8], where it is proved that twelve membranes can provide the computational universality. In [2] we improve this result by reducing the number of membranes to nine.

Theorem 2 ([2]). *Systems of nine enhanced mobile membranes using rules of types* (endo), (exo), (fendo), (fexo) *have the same computational power as a Turing machine.*

3.3. Mutual mobile membranes

Systems of *mutual mobile membranes* are defined in [3]. In these systems the endocytosis and the exocytosis work whenever the involved membranes "agree" on the movement; this agreement between membranes is expressed by appropriate objects and co-objects. The evolution of mutual mobile membranes is given by the following rules:

$[uv]_h[\overline{u}v']_m \to [\ [w]_h w']_m$ for $h, m \in H, u, \overline{u} \in V^+, v, v', w, w' \in V^*$;
mutual endocytosis (mendo)

$[[uv]_h \overline{u}v']_m \to [w]_h[w']_m$ for $h, m \in H, u, \overline{u} \in V^+, v, v', w, w' \in V^*$;
mutual exocytosis (mexo)

We have proved that it is enough to consider three mutual mobile membranes and the mobility rules (mendo) and (mexo) to get the full computational power of a Turing machine [3]. The number of three membranes represents the minimum number with respect to the movement provided by endocytosis and exocytosis.

Theorem 3 ([3]). *Systems of three mutual mobile membranes using rules of types* (mendo)*,* (mexo) *have the same power as a Turing machine.*

We notice that in each mobility rule from systems of simple and enhanced mobile membranes, in the left hand side of the rules only one object appears. By using multisets instead of objects and synchronization by objects and co-objects, we prove that it is enough to consider three mutual mobile membranes using (mendo) and (mexo) rules to obtain the power of Turing machines.

3.4. Mutual mobile membranes with objects on surface

We define systems of mutual membranes with objects on surface, following the idea of adding objects on membranes and using the biologically inspired rules pino/exo/phago as in [5, 6, 13, 14]. Inspired from brane calculus [4], we use objects and co-objects in phago and exo rules in order to illustrate the fact that both involved membranes agree on the movement. The evolution from a configuration to another is given by the following rules:

$[\ ]_{au\overline{a}v} \Rightarrow [[\ ]_{ux}]_{vy}$, for $a, \overline{a} \in V, u, v, x, y \in V^*, ux, vy \in V^+$ pino

$[[\ ]_{au}]_{\overline{a}v} \Rightarrow [\ ]_{uvx}$, for $a, \overline{a} \in V, u, v, x \in V^*, uvx \in V^+$ exo

$[\ ]_{au}[\ ]_{\overline{a}bv} \Rightarrow [[[\ ]_{ux}]_b]_{vy}$, for $a, \overline{a}, b \in V, u, v, x, y \in V^*, ux, vy \in V^+$ phago

We investigate the computational power of systems of mutual mobile membranes with objects on surface controlled by pairs of rules: pino/exo or phago/exo, proving that they are universal using a small number of membranes. Table 1 contains a summary of results (existing and new ones).

The multiplicity vector of the multiset from all membranes is considered as a result of the computation. Thus, the result of a halting computation consists of all the vectors describing the multiplicity of objects from all the membranes; a non-halting computation provides no output. The number of objects from the right-hand side of a rule is called its *weight*. In Table 1 systems of n mutual mobile membranes with objects on surface using *rules* (op_1, op_2) of weights (w_1, w_2) have the same computational power as a Turing machine.

Since each recursively enumerable language can be generated by a matrix grammar in strong binary normal form [11], the proofs of these results are given by constructing systems of mutual mobile membranes with objects on surface able to generate the same language as a matrix grammar in strong binary normal form. A matrix grammar $G = (N, T, S, Z, F)$ is in a strong binary normal form if the non-terminal alphabet $N = N_1 \cup N_2 \cup \{S, \dagger\}$

Table 1: Summary of Results

Number n of membranes	Rules (op_1, op_2)	Weights (w_1, w_2)	Article
8	(pino, exo)	(4,3)	Theorem 6.1 [13]
3	(pino, exo)	(5,4)	Theorem 4
9	(phago, exo)	(5,2)	Theorem 6.2 [13]
9	(phago, exo)	(4,3)	Theorem 6.2 [13]
4	(phago, exo)	(6,3)	Theorem 5

is a union of mutually disjoint sets, T is a terminal alphabet such that $N \cap T = \emptyset$, $S \in N$ is the initial symbol, Z is a finite set of matrices of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, with $A_i \to x_i$ a rewriting rule, $A_i \in N$, $x_i \in (N \cup T)^*$, for all $1 \leq i \leq n$, and F is a subset of the rules appearing in the set of matrices Z. Moreover, the matrices in Z are:

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. $(X \to Y, A \to \dagger)$, with $X, Y \in N_1$, $A \in N_2$;
4. $(X \to \lambda, A \to x)$, with $X \in N_1$, $A \in N_2$, $x \in T^*$, $|x| \leq 2$.

It is known that systems of eight mobile membranes with objects on surface and using pino and exo rules of weights four and three are universal [13]. We show here that the number of membranes can be reduced from eight to three by increasing the weight of the pino and exo operations with one, namely from four to five and from three to four, respectively.

Theorem 4. *Systems of three mutual mobile membranes with objects on surface using the rules* pino *of weight five and* exo *of weight four have the same computational power as a Turing machine.*

Proof. The inclusion into $PsRE$ is assumed true by invoking the Church-Turing thesis. This implies that we have to prove only the inclusion of $PsRE$ into the language generated by a system $\Pi = (V, \mu, u_1, u_2, u_3, R)$ of three mutual membranes with objects on surface able to simulate a matrix grammar with appearance checking in the binary normal form.

The system $\Pi = (V, \mu, u_1, u_2, u_3, R)$ is defined by

$$\mu = \{(1,2);(2,3)\}, \quad u_1 = \lambda, \quad u_2 = \overline{\beta}X, \quad u_3 = \beta A$$
$$V = \{\beta, \overline{\beta}\} \cup \{\alpha, \alpha' \mid \alpha \in N_2 \cup T\}$$
$$\cup \{X, X_l, X_l', X_l'', X_l^{(j)}, X_l^{(j)'} \mid X \in N_1, 1 \le l \le n_1 + n_2, 1 \le j \le 2\},$$

and the following set R of rules.

The initial type-1 matrix is $(S \to XA)$ with $X \in N_1, A \in N_2$.

1. For each type-2 matrix $m_l : (X \to Y, A \to x)$, $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$, with $1 \le l \le n_1$, we consider the rules:

    1. $[[\ ]_{\beta A}]_{\overline{\beta}X} \to [\ ]_{\beta\overline{\beta}X_l A}$ (exo)
       $[[\ ]_{\beta}]_{\overline{\beta}XA} \to [\ ]_{\beta\overline{\beta}X_l A}$ (exo)
    2. $[\ ]_{\beta\overline{\beta}X_l A} \to [[\ ]_{\beta x'}]_{\overline{\beta}X_l}$, if $x \neq \lambda$ (pino)
       (If $m_l : (X \to Y, A \to \alpha_1\alpha_2)$ then $x' = \alpha_1'\alpha_2$ or $x' = \alpha_1\alpha_2'$, and if $m_l : (X \to Y, A \to \alpha_1)$ then $x' = \alpha_1'$)
       $[\ ]_{\beta\overline{\beta}X_l A} \to [[\ ]_{\beta f}]_{\overline{\beta}X_l}$, if $x = \lambda$ (pino)
    3. $[[\ ]_{\beta\alpha'}]_{\overline{\beta}X_l} \to [\ ]_{\beta\overline{\beta}\alpha' X_l'}$ (exo)
       $[[\ ]_{\beta f}]_{\overline{\beta}X_l} \to [\ ]_{\beta\overline{\beta}f X_l'}$ (exo)
    4. $[\ ]_{\beta\overline{\beta}\alpha' X_l'} \to [[\ ]_{\beta\alpha}]_{\overline{\beta}X_l'}$ (pino)
       $[\ ]_{\beta\overline{\beta}f X_l'} \to [[\ ]_{\beta}]_{\overline{\beta}X_l'}$ (pino)
    5. $[[\ ]_{\beta}]_{\overline{\beta}X_l'} \to [\ ]_{\beta\overline{\beta}X_l''}$ (exo)
    6. $[\ ]_{\beta\overline{\beta}X_l''} \to [[\ ]_{\beta}]_{\overline{\beta}Y}$ (pino)
    7. $[[\ ]_{\beta}]_{\overline{\beta}X} \to [\ ]_{\beta\overline{\beta}\dagger}$ (exo)
    8. $[\ ]_{\beta\overline{\beta}\dagger} \to [[\ ]_{\beta}]_{\overline{\beta}\dagger}$ (pino)
    9. $[[\ ]_{\beta}]_{\overline{\beta}\dagger} \to [\ ]_{\beta\overline{\beta}\dagger}$ (exo)

   Using rule 1, we replace X by X_l marking the beginning of the simulation. This is followed by rule 2, where X_l is used to replace A by either x' (in case $x \neq \lambda$) or f (in case $x = \lambda$). Next, we apply rule 3 to replace X_l by X_l', preventing so to replace other A's. In rule 4 we use X_l' to replace α' by α, while rule 5 replaces X_l' by X_l''. Rule 6 is replacing X_l'' by Y, thus successfully simulating a type-2 matrix and returning to the initial membrane structure. If the corresponding symbol $A \in N_2$ is not present (i.e., we cannot apply rule 1), rule 7 introduces a trap symbol which leads to an infinite computation (rules 8 and 9).

2. For each type-3 matrix $m'_l : (X \to Y, B^{(j)} \to \dagger)$, $X, Y \in N_1$, $A \in N_2$, where $n_1 + 1 \leq l \leq n_1 + n_2$, $l \in lab_j$, $j = 1, 2$, we consider the rules:

   10. $[[\ ]_\beta]_{\overline{\beta}X} \to [\ ]_{\beta\overline{\beta}X_l^{(j)}}$ (exo)
   11. $[\ ]_{\beta\overline{\beta}X_l^{(j)}} \to [[\ ]_\beta]_{\overline{\beta}X_l^{(j)}}$ (pino)
   12. $[[\ ]_{\beta B^{(j)}}]_{\overline{\beta}X_l^{(j)}} \to [\ ]_{\beta\overline{\beta}X_l^{(j)}\dagger}$ (exo)
       $[[\ ]_\beta]_{\overline{\beta}B^{(j)}X_l^{(j)}} \to [\ ]_{\beta\overline{\beta}X_l^{(j)}\dagger}$ (exo)
   13. $[[\ ]_\beta]_{\overline{\beta}X_l^{(j)}} \to [\ ]_{\beta\overline{\beta}X_l^{(j)\prime}}$ (exo)
   14. $[\ ]_{\beta\overline{\beta}X_l^{(j)\prime}} \to [[\ ]_\beta]_{\overline{\beta}Y}$ (pino)

   Rule 10 starts the simulation of a type-3 matrix by replacing X with $X_l^{(j)}$, thereby remembering the index l of the matrix and the index j of the possible present symbol $B^{(j)}$. This rule is followed by rule 11. At this step we need to check if the corresponding symbol $B^{(j)} \in N_2$ is present. The symbol $B^{(j)}$ (if present) is part of either the membrane containing β or the membrane containing $\overline{\beta}$. If $B^{(j)}$ is present, rule 12 replaces it with $\dagger$ and by applying rule 11, we go to the configuration before replacing $B^{(j)}$. Regardless the presence of $B^{(j)}$, rule 13 is replacing $X_l^{(j)}$ with $X_l^{(j)\prime}$. Finally rule 14 is replacing $X_l^{(j)}$ with Y, thus successfully simulating a type-3 matrix and returning to the initial membrane structure.

3. For a type-4 matrix $m_l : (X \to a, A \to x)$, $X \in N_1$, $a \in T$, $A \in N_2$, $x \in T^*$, where $1 \leq l \leq n_1$.

   15. $[\ ]_{\beta\overline{\beta}X_l''} \to [[\ ]_\beta]_{\overline{\beta}a}$ (pino)
   16. $[[\ ]_\beta]_{\overline{\beta}A} \to [\ ]_{\beta\overline{\beta}\dagger}$ (exo)
       $[[\ ]_{\beta A}]_{\overline{\beta}} \to [\ ]_{\beta\overline{\beta}\dagger}$ (exo)

   Starting with rules 1 to 5 and continuing with rule 15 we successfully simulate the terminal matrix. Rule 16 together with rule 8 should be used if any $A \in N_2$ exists after simulation of a terminal matrix, leading to an infinite computation. It is enough to check the applicability of rule 16, after applying all possible rules. The result of a correct simulation is the set of all symbols present on all membranes without the symbols β and $\overline{\beta}$. $\square$

It is known that systems of nine membranes with objects on surface and using phago and exo rules of weight four and three (or five and two) are universal [13]. We prove here that the number of membranes can be reduced from nine to four, but in order to do this we increase the weight of the phago rule to six and of the exo rule to three.

Theorem 5. *Systems of four mutual membranes with objects on surface using rules* phago, exo *of weights six and three have the same computational power as a Turing machine.*

Proof. The inclusion into $PsRE$ is assumed true by invoking the Church-Turing thesis. This implies that we have to prove only the inclusion of $PsRE$ into our language. For this we construct a system $\Pi = (V, \mu, u_1, u_2, u_3, R)$ of three mutual mobile membranes with objects on surface able to simulate a matrix grammar with appearance checking in the binary normal form.

The system $\Pi = (V, \mu, u_1, u_2, u_3, R)$ is defined by

$$\mu = \{(1,2); (1,3)\}, \quad u_1 = \lambda, \quad u_2 = \beta\overline{\beta}X, \quad u_3 = \beta A$$
$$V = \{\beta, \overline{\beta}\} \cup \{\alpha, \alpha' \mid \alpha \in N_2 \cup T\}$$
$$\cup \{X, X_l, X_l', X_l'', X_l^{(j)}, X_l^{(j)'} \mid X \in N_1, 1 \leq l \leq n_1 + n_2, 1 \leq j \leq 2\},$$

and the following set R of rules.

The initial type-1 matrix is $(S \to XA)$ with $X \in N_1, A \in N_2$.

1. For each type-2 matrix $m_l : (X \to Y, A \to x)$, $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$, with $1 \leq l \leq n_1$, we consider the rules:

   1. $[\,]_{\beta A}[\,]_{\beta\overline{\beta}X} \to [[[\,]_{\beta A}]_{\beta}]_{\overline{\beta}X_l}$ (phago)
   2. $[[\,]_{\beta}]_{\overline{\beta}X_l} \to [\,]_{\beta\overline{\beta}X_l}$ (exo)
   3. $[\,]_{\beta A}[\,]_{\beta\overline{\beta}X_l} \to [[[\,]_{\beta x'}]_{\beta}]_{\overline{\beta}X_l'}$ (phago)
      (If $m_l : (X \to Y, A \to \alpha_1\alpha_2)$ then $x' = \alpha_1'\alpha_2$ or $x' = \alpha_1\alpha_2'$, and if $m_l : (X \to Y, A \to \alpha_1)$ then $x' = \alpha_1'$)
      $[\,]_{\beta A}[\,]_{\beta\overline{\beta}X_l} \to [[[\,]_{\beta f}]_{\beta}]_{\overline{\beta}X_l'}$ (phago), if $x = \lambda$

4. $[[\ ]_{\beta}]_{\overline{\beta}X'_l} \to [\ ]_{\beta\overline{\beta}X'_l}$ (exo)
5. $[\ ]_{\beta\alpha'}[\ ]_{\beta\overline{\beta}X'_l} \to [[[\ ]_{\beta\alpha}]_{\beta}]_{\overline{\beta}X''_l}$ (phago)
   $[\ ]_{\beta f}[\ ]_{\beta\overline{\beta}X'_l} \to [[[\ ]_{\beta}]_{\beta}]_{\overline{\beta}X''_l}$ (phago)
6. $[[\ ]_{\beta}]_{\overline{\beta}X''_l} \to [\ ]_{\beta\overline{\beta}Y}$ (exo)
7. $[\ ]_{\beta}[\ ]_{\beta\overline{\beta}X} \to [[[\ ]_{\beta}]_{\beta}]_{\overline{\beta}\dagger}$ (phago)
8. $[[\ ]_{\beta}]_{\overline{\beta}\dagger} \to [\ ]_{\beta\overline{\beta}\dagger}$ (exo)
9. $[\ ]_{\beta}[\ ]_{\beta\overline{\beta}\dagger} \to [[[\ ]_{\beta}]_{\beta}]_{\overline{\beta}\dagger}$ (phago)

Rule 1 is used to replace X by X_l, marking the beginning of the simulation; this rule is followed by rule 2. In rule 3, X_l is used to replace A by either x' (in case $x \neq \lambda$) or f (in case $x = \lambda$) while X_l is replaced by X'_l in order to prevent replacing A's anymore. This is followed by rule 4. In rule 5 we use X'_l to replace α' by α, while X'_l is replaced by X''_l. Rule 6 is replacing X''_l by Y, thus successfully simulating a type-2 matrix and returning to the initial membrane structure. If the corresponding symbol $A \in N_2$ is not present (i.e., we cannot apply rule 1), rule 7 introduces a trap symbol which leads to an infinite computation (by using rules 8 and 9).

2. For each type-3 matrix $m'_l : (X \to Y, B^{(j)} \to \dagger)$, $X, Y \in N_1$, $A \in N_2$, where $n_1 + 1 \leq l \leq n_1 + n_2$, $l \in lab_j$, $j = 1, 2$, we consider the rules:

10. $[\ ]_{\beta}[\ ]_{\beta\overline{\beta}X} \to [[[\ ]_{\beta}]_{\beta}]_{\overline{\beta}X_l^{(j)}}$ (phago)
11. $[[\ ]_{\beta}]_{\overline{\beta}X_l^{(j)}} \to [\ ]_{\beta\overline{\beta}X_l^{(j)}}$ (exo)
12. $[\ ]_{\beta B^{(j)}}[\ ]_{\beta\overline{\beta}X_l^{(j)}} \to [[[\ ]_{\beta}]_{\beta}]_{X_l^{(j)}\dagger}$ (phago)
13. $[\ ]_{\beta}[\ ]_{\beta\overline{\beta}X_l^{(j)}} \to [[[\ ]_{\beta}]_{\beta}]_{X_l^{(j)\prime}}$ (phago)
14. $[[\ ]_{\beta}]_{\overline{\beta}X_l^{(j)\prime}} \to [\ ]_{\beta\overline{\beta}Y}$ (exo)

Rule 10 starts the simulation of a type-3 matrix by replacing X with $X_l^{(j)}$, thereby remembering the index l of the matrix and the index j of the possible present symbol $B^{(j)}$. This is followed by rule 11. At this step we need to check if the corresponding symbol $B^{(j)} \in N_2$ is present. If $B^{(j)}$ is present, rule 12 replaces it with $\dagger$ and by applying rule 11, we go to the configuration before replacing $B^{(j)}$. Regardless the presence of $B^{(j)}$, rule 13 is applied to replace $X_l^{(j)}$ by $X_l^{(j)'}$. Rule 14 is replacing $X_l^{(j)}$ by Y, thus successfully simulating a type-3 matrix and returning to the initial structure.

3. For a type-4 matrix $m_l : (X \to a, A \to x)$, $X \in N_1$, $a \in T$, $A \in N_2$, $x \in T^*$, where $1 \leq l \leq n_1$, we consider the rules

   15. $[[\ ]_\beta]_{\overline{\beta}X_l''} \to [\ ]_{\beta\overline{\beta}a}$ (exo)
   16. $[\ ]_{\beta A}[\ ]_{\beta\overline{\beta}} \to [[[\ ]_\beta]_\beta]_{\overline{\beta}\dagger}$ (phago)

   Starting with rules 1 to 5 and continuing with rule 15, we successfully simulate the terminal matrix. Rule 16 together with rule 8 should be used if a $A \in N_2$ exists after simulation of a terminal matrix, leading to an infinite computation. It is enough to check the applicability of rule 16 after applying all the possible rules. The result of a correct simulation is the set of all symbols present on all membranes without the symbols β and $\overline{\beta}$. □

4. Conclusion

Inspired by the biological cell movements given by endocytosis and exocytosis, we defined several classes of mobile membranes and prove that they have the same computability power as Turing machines. We improve the existing results by showing that systems of three mobile membranes with objects on surface using the rules pino and exo of weights five and four, as well as systems of four mobile membranes with objects on surface using the rules phago and exo of weights six and three have the same computational power as a Turing machine. These results emphasize the balance between minimizing the number of membranes and the weights of the moving operations when designing computationally universal systems of mobile membranes with objects on surface.

REFERENCES

1. Aman, B.; Ciobanu, G. – *Describing the immune system using enhanced mobile membranes,* Electr. Notes in Theoretical Computer Science, 194 (2008), 5–18.

2. Aman, B.; Ciobanu, G. – *Simple, enhanced and mutual mobile membranes,* Transactions on Computational Systems Biology XI, LNBI, 5750 (2009), 26–44.

3. Aman, B.; Ciobanu, G. – *Turing completeness using three mobile membranes,* Lecture Notes in Computer Science, 5715 (2009), 42–55.

4. Cardelli, L. – *Brane calculi. Interactions of biological membranes,* Lecture Notes in BioInformatics, Springer, 3082 (2004), 257–278.

5. Cardelli, L.; Păun, Gh. – *An universality result for a (Mem)brane calculus based on mate/drip operations,* Intern. J. Foundations of Computer Science, 17 (2006), 49–68.

6. Cavaliere, M.; Sedwards, S. – *Membrane systems with peripherial proteins: transport and evolution,* Electr. Notes in Theoretical Computer Science, 171 (2007), 37–53.

7. Ciobanu, G. – *Membrane Computing & Biologically Inspired Process Calculi*, "Al.I. Cuza" University Press, Iaşi, 2010.

8. Ciobanu, G.; Krishna, S.N. – *Enhanced mobile membranes: computability results,* Theory of Computing Systems, 78 (2011), 715–729.

9. Ciobanu, G.; Pérez-Jiménez, M.J.; Păun, Gh. (eds.) – *Applications of Membrane Computing*, Springer, 2006.

10. Cohen, J.E. – *Mathematics is Biology's Next Microscope, Only Better; Biology is Mathematics' Next Physics, Only Better,* PLoS Biology, 12 (2004), e439.

11. Dassow, J.; Păun, Gh. – *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, 1990.

12. Krishna, S.N. – *The power of mobility: four membranes suffice,* Lecture Notes in Computer Science, Springer, 3526 (2005), 242–251.

13. Krishna, S.N. – *Universality results for P systems based on brane calculi operations,* Theoretical Computer Science, 371 (2007), 83–105.

14. Krishna, S.N. – *Membrane computing with transport and embedded proteins,* Theoretical Computer Science, 410 (2009), 355–375.

15. Păun, Gh. – *Membrane Computing. An Introduction,* Springer-Verlang, Berlin, 2002.

16. Păun, Gh.; Rozenberg, G.; Salomaa, A. (eds.) – *Handbook of Membrane Computing*, Oxford University Press, 2010.

Received: 18.X.2010
Revised: 30.XI.2010

University "Al.I. Cuza",
11, Bd. Carol I, 700506, Iaşi,
ROMANIA
and
Institute of Computer Science,
Romanian Academy, Iaşi,
ROMANIA
baman@iit.tuiasi.ro
gabriel@info.uaic.ro