P Systems without Multiplicities of Symbol-Objects

Artiom Alhazov *

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: artiome.alhazov@estudiants.urv.es

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

Abstract

In this paper we investigate P systems whose compartments contain sets of symbol-objects rather than multisets of objects, as it is common in membrane computing. If the number of membranes cannot grow, then in this framework we can characterize exactly the regular languages. If membrane creation or membrane division is allowed, then the Parikh sets of recursively enumerable languages can be generated. The last result also implies the universality of P systems with active membranes (with multisets of symbol-objects) without polarizations.

1 Introduction

Membrane computing is a vivid branch of natural computing inspired from the structure and functioning of living cells. Details can be found in [5] (see also http://psystems.disco.unimib.it for the the state-of-the-art of the domain).

Very shortly, in the compartments of a hierarchical arrangement of membranes one places multisets of symbol-objects and rules by which these objects can evolve. Using the rules in a non-deterministic maximally parallel manner, for evolving the objects from the compartment with which the rules are associated, we obtain transitions from a configuration of the system to another configuration. Such a sequence of transitions is called a computation; with a halting computation we associate a result, given by the multiplicity of objects sent to the environment during the computation.

Clearly, the hierarchy of membranes correspond to the cell membranes and vesicles, the objects remind the chemicals and the evolution rules remind the reactions by which these chemicals evolve.

However, counting the chemicals present in a cell is not a very realistic assumption, the biochemists can in general detect in an easier way the presence of a substance than to evaluate its abundance. Thus, it is natural to consider P systems in whose compartments

*Work supported by project TIC2002-04220-C03-02 of the Research Group on Mathematical Linguistics, Tarragona, the Moldovan Research and Development Association (MRDA), and the U.S. Civilian Research and Development Foundation (CRDF), Award No. MM2-3034.

we have *sets* of objects, rather than multisets (sets with multiplicities associated with the elements).

Because in such a system we cannot encode "too much" information in the objects themselves, in order to compute at a higher level than at that of finite automata, we need to use other ingredients, too, and the idea we follow is to create (and dissolve) arbitrarily many membranes. Rather surprisingly, by making use of this possibility we can reach the computational universality: all sets of vectors of natural numbers which are Turing computable can be computed by P systems with membrane creation and dissolution, working with sets of objects. This result is proved also for P systems with active membranes without polarizations.

Extensions of our main results to other classes of P systems are briefly discussed.

2 Preliminaries

The reader is assumed to be familiar with basic elements of formal language theory, e.g., from [4], [7], [5]. In particular, the notations we use are standard: by V^* we denote the free monoid generated by an alphabet V; the empty string is denoted by λ, and $|x|$ is the length of $x \in V^*$; the Parikh mapping (associated with an alphabet V) is denoted by Ψ_V. The families of regular and recursively enumerable languages are denoted by REG and RE, respectively. For a family FL of languages, we denote by $PsFL$ the family of Parikh images of languages in FL (thus, $PsRE$ is the family of Turing computable sets of vectors of natural numbers).

A *multiset* over a finite set V is a mapping $M : V \longrightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. The support of a multiset M is the set $supp(M) = \{a \in V \mid M(a) > 0\}$. We represent multisets by strings: a string $w \in V^*$ identifies the multiset M whose multiplicities (values of $M(a), a \in V$) are given by $Ps_V(w)$; clearly, all permutations of the string w represent the same multiset.

In the universality proofs below we use *matrix grammars with appearance checking* in the binary normal form. Such a grammar is a construct $G = (N, T, S, P, F)$, with $N = \{S, \#\} \cup N_1 \cup N_2$ (these three sets are mutually disjoint, the elements of N_1 are the "control nonterminals", the elements of N_2 are the "literal nonterminals"), and with the matrices from P in one of the following forms: $(S \to X_{init}A_{init})$ (unique), $(X \to Y, A \to x)$, $(X \to Y, A \to \#)$, $(X \to \lambda, A \to x)$, where $X_{init}, X, Y \in N_1$, $A_{init}, A \in N_2$ and $x \in (N_2 \cup T)^*, |x| \leq 2$. F is the set of the rules that can be used in appearance-checking mode (skipped if not applicable), and this set consists of exactly all the rules producing $\#$.

3 P Systems with Multisets and with Sets of Objects

We first introduce briefly the classes of P systems we investigate here, in their standard form, that is, with multisets of objects in their compartments.

The first class is that of *P systems with multiset-rewriting* rules, also provided with *membrane creation* ($mcre$) and *membrane dissolution* (δ) possibilities. Such a system (of initial degree $m \geq 1$) is of the form

$$\Pi = (O, H, \mu, w_1, \ldots, w_m, R_1, \ldots, R_n),$$

where O is the alphabet of objects, H is the set of labels for membranes (we assume here that H contains n labels), μ is the initial membrane structure, consisting of m membranes labelled (not necessarily in a one-to-one manner) with elements of H, $w_1, \ldots, w_m$ are strings over O representing the multisets of objects present in the m compartments (also called regions) of μ, and $R_1, \ldots, R_n$ are the (finite) sets of rules associated with the n labels from H. These rules can be of the following two forms: (1) $u \to v$, (2) $a \to [_i w]_i$, where $u, w \in O^*, a \in O, i \in H$, and either $v \in (O \times Tar)^*$, or $v \in (O \times Tar)^*\delta$, with $Tar = \{here, out\} \cup \{in_j \mid j \in H\}$. The presence of δ in the right-hand side of a rule means that application of the rule leads to the dissolution of the membrane.

The meaning of a rule of type 1 is that the multiset of objects u from the region associated with the rule "reacts", and as a result the objects specified by v are produced. The objects from v have associated *target commands*, of the forms *here, out*, in_j, which specify where the object should be placed (*here* means that the object remains in the region where it is produced, in_j means that it has to go to the membrane j, provided that it is directly inside the membrane where the rule is applied (otherwise the rule cannot be used), and *out* indicates that the object should exit the current membrane, going to the surrounding region – which is the environment in the case of the skin membrane of the system). In general, the indication *here* is not explicitly written. If the special symbol δ is present, this means that after using the rule the membrane is dissolved, and all its contents, objects and membranes alike, become elements of the surrounding region. The skin membrane is never dissolved.

In turn, a rule $a \to [_i v]_i$ of type (2) means that object a produces a new membrane, with label i, containing the objects specified by v. Knowing the label of the membrane, we also know the rules associated with it. We recall that the number of membrane labels (i.e., kinds of membranes) is n and the rules are associated to the membrane labels, while the number of membranes is m initially and can change.

If all rules of type (1) have $|u| = 1$ (hence consisting of only one symbol), then the system is said to be *non-cooperative* (the general case is then called *cooperative*).

As mentioned above, the rules are used in the non-deterministic (the objects and the rules are chosen non-deterministically) maximally parallel way (no further object can evolve after a chosen assignation of objects to rules), thus obtaining transitions between a configuration of the system to another configuration (of course, a configuration is described by the current membrane structure and the contents of the compartments of this membrane structure; note that the number of membranes can grow by membrane creation and can decrease by membrane dissolution). When a configuration is reached where no rule can be applied, the computation stops, and the multiplicity of objects sent into environment during the computation is said to be computed by the system along that computation. We denote by $Ps(\Pi)$ the set of vectors computed in this way (by means of all computations) by a system Π. The family of all sets $Ps(\Pi)$ computed by systems Π using non-cooperative rules is denoted by $PsOP(ncoo, mcre, \delta)$.

Another class of P systems central in membrane computing is that *of P systems with active membranes*, which are constructs $\Pi = (O, H, \mu, w_1, \ldots, w_m, R)$, with the components $O, H, \mu, w_1, \ldots, w_m$ defined in the same way as above, with membranes of μ also having associated *polarizations*, one of the symbols $+, -, 0$, and with the rules from R of the following forms: (a): $[_h a \to v]^e_h$ – rewriting-like, (b): $a[_h]^e_h \to [_h b]^{e'}_h$ and (c): $[_h a]^e_h \to [_h]^{e'}_h b$ – bring an object inside the membrane/send an object out of the membrane (possibly changing its polarization), (d): $[_h a]^e_h \to b$ – dissolve the membrane, producing another object (the contents of the dissolved membrane is released in the surrounding region),

(e): $[_h a]^e_h \to [_h b]^{e'}_h [_h c]^{e''}_h$ – membrane division, where two membranes with the same label (but possibly different polarizations) are produced, each containing a new object (all other objects are duplicated). In all cases, $a, b \in O, v \in O^*, e, e' \in \{+, -, 0\}$.

The rules of type (a) are applied in the maximally parallel way, while at most one rule of types (b), (c), (d), (e) can be applied for each membrane at any step of the computation. If the membranes are not polarized (equivalently, we may say that all of them have the neutral polarization, 0), then we also attach the subscript 0 to the above notation for types of rules, thus obtaining (a_0), (b_0), (c_0), (d_0), (e_0). The computations and the result of (halting) computations are defined as for the first type of P systems mentioned above. The family of vectors computed by such systems is denoted by $PsOP(a, b, c, d, e)$; when one type of rules is missing, we omit it from the notation; when polarizationless membranes are used, we add the subscript 0 to the "names" of rules.

It is known (see, e.g., [5]) that P systems with active membranes (with polarizations) are universal, they characterize $PsRE$ (even for reduced combinations of ingredients, such as number of initial membranes, and types of rules used). In turn, the possibility to create exponentially many membranes in a linear time by using division rules can be used for trading-off space for time and solve, e.g., **NP**-complete problems in a polynomial time.

Let us now ignore the multiplicities of objects from the regions of a P system. This amounts to consider only the support of multisets as the only information provided by the objects from each region. Thus, a transition in a system without multiplicities can be viewed as a transition in a system with multiplicities, followed by ignoring multiplicities of all objects in all regions (except the environment, where we collect the result). (For example, if in a region i we have the objects a, b and the rules $a \to c$, $b \to c$, then, after using these rules we get a configuration which has only the object c in region i.)

This restriction can be considered for all classes of P systems, in particular for the two classes introduced above. The generated set of vectors is defined in the same way (taking into account the objects sent to the environment). The corresponding families of vectors are denoted by using $PsoP(\ldots)$ instead of $PsOP(\ldots)$.

4 No Membrane Creation/Division Means Regularity

When no new membrane can be created, the configuration of a system working with sets of objects has a limited size, hence the fact that such systems have a very limited computing power is quite expected:

Theorem 1 *P systems with a bounded number of membranes without object multiplicities generate exactly the Parikh images of regular languages. This holds for all systems without membrane creation and without membrane division.*

Proof. For any fixed membrane structure, say with n membranes, and a given set O of objects, the number of possible configurations is $\left(2^{card(O)}\right)^n$. Starting from the initial configuration, any system as in the theorem will pass through configuration from a finite set (the number of membranes can decrease by dissolution, but it cannot increase); we denote this set by $\mathcal{C}$.

From every configuration C of a P system Π of any type as in the theorem, there is a finite number of transitions to other configurations C', ejecting a finite set of objects w into the environment. We denote such a transition by $C \Rightarrow (C', w_{out})$. Then, with any system Π we can associate a finite automaton $A = (\mathcal{C}, T, \delta, C_0, \mathcal{H})$, where $C_0 \in \mathcal{C}$ is the initial

configuration of Π, $\mathcal{H} \subseteq \mathcal{C}$ is the set of halting configurations, $T = \{\langle w \rangle \mid \exists C, C' \mid C \Rightarrow (C', w_{out})\}$ (symbols associated to the multisets that can be sent in the environment in one computation step), $\delta(C, \langle w \rangle) = \{C' \mid C \Rightarrow (C', w_{out})\}$ for all $C, C' \in \mathcal{C}$. Then A accepts a regular language over T whose Parikh image is equal to $Ps(\Pi)$, hence $Ps(\Pi) \in PsREG$.

Conversely, starting from a finite automaton $A = (Q, T, \delta, q_0, F)$ we can construct a P system $\Pi = (Q \cup T \cup \{\#\}, \{1\}, [_1\,]_1, q_0, R_1)$, where $R_1 = \{q \to a_{out} q' \mid \delta(q, a) = q'\} \cup \{q \to \lambda \mid q \in F\} \cup \{q \to \# \mid q \in Q \cup \{\#\}\}$. (A transition of A from state q to state q' reading a is simulated by a rule transforming q into q' and sending object a outside. As long as a non-final state is present, the computation must continue, possibly introducing the trap-symbol $\#$. If a state q is final, then the corresponding object may be erased, leading to a halting configuration.) The equality $\Psi_T(L(A)) = Ps(\Pi)$ is obvious. It is also obvious how to write the rules of the system Π as rules of a system with active membranes (only rules of types (a_0) and (c_0) are used). □

Example 1 *An automaton $A = (\{p, q\}, \{a\}, \delta, p, \{p, q\})$, where $\delta(p, a) = \{p\}$, $\delta(p, b) = \{q\}$, $\delta(q, a) = \emptyset$, $\delta(q, b) = \{q\}$ accepts $\{a^m b^n \mid m, n \geq 0\}$. It corresponds to the membrane system $\Pi = (\{p, q, a, b, \#\}, \{1\}, [_1\,]_1, p, R_1)$, $R_1 = \{p \to a_{out} p, p \to b_{out} q, q \to b_{out} q, q \to \lambda, p \to \#, q \to \#, \# \to \#\}$.*

Remark 1 *The previous proof can be easily adapted to all types of P systems without rules for increasing the number of membranes and working with sets of symbol-objects, for instance, to systems with boundary rules [2], with evolution-communication [3], etc. Moreover, in this framework we can obtain a characterization of regular languages, not only of their Parikh images, by considering as the result of a computation in a P system not the multiplicities of objects sent into the environment, but the sequence in which these objects are expelled; in this way, a computation generates a string (actually, a set of strings, because when several objects are sent out in the same step, then all their permutations are accepted as a substring of the generated string). We leave the easy technical modifications in the previous proof as a task for the reader.*

5 The Power of Membrane Creation/Division

In contrast to the previous theorem, when membrane creation or membrane division (without polarizations) is available, we reach the computational universality. We consider first the case of membrane creation.

Theorem 2 $PsoP(ncoo, mcre, \delta) = PsRE$.

Proof. Consider a matrix grammar G in the binary normal form (see the definition in the end of Section 2), $G = (N, T, S, P, F)$, say with k matrices.

The idea of the construction below is to represent every $A \in N_2$ by a membrane $[_A \$]_A$ and any $X \in N_1$ by an object with the same name (the terminals from T are also objects, which are immediately sent to the environment).

Let us define N_1' as $\{X' \mid X \in N_1\}$.

We construct the P system

$$\begin{aligned} \Pi &= (O, H = \{1\} \cup N_2 \cup N_1, [_1]_1, ss', \{R_h \mid h \in H\}), \\ O &= T \cup N_1 \cup N_1' \cup N_2 \cup \{\lambda', \$, \#, s, s'\} \cup \{m_r, m_{r,1}, m_{r,2} \mid 2 \leq r \leq k\}, \end{aligned}$$

with the following rules:

Each matrix $m_r : (X \to Y, A \to x)$, $Y \in N_1 \cup \{\lambda\}$, is simulated by the rules:

A1. $(X \to X_{in_A}) \in R_1$,

A2. $(X \to m_r\delta) \in R_A$,

A3. $(m_r \to m_{r,1}m_{r,2}Y') \in R_1$,

A4. $(m_{r,i} \to [_{x_i}\$]_{x_i}) \in R_1$ if $x_i \in N_1$,

A5. $(m_{r,i} \to x_{iout}) \in R_1$ if $x_i \in T$,

A6. $(m_{r,i} \to \lambda) \in R_1$ if $x_i = \lambda$,

where x_1 is the first symbol of x (or λ if $x = \lambda$, and $x = x_1x_2$). Here, an object X enters a membrane with label A, then it dissolves the membrane while changing to m_r, and then m_r evolves into $m_{r,1}m_{r,2}Y'$. The first two objects produced evolve into the representation of x, while the last one evolves into Y (by rule $C4$ below), thus finishing the simulation of m_r.

Each matrix $m_r : (X \to Y, A \to \#)$, is simulated by the rules:

B1. $(X \to m_rA) \in R_1$,

B2. $(A \to \#_{in_A}) \in R_1$,

B3. $(m_r \to [_Y\$]_Y) \in R_1$

B4. $(A \to A_{in_Y}) \in R_1$,

B5. $(A \to Y\delta) \in R_Y$.

Here, if a membrane with label A exists, then the computation is blocked, otherwise X changes to Y.

We also add the rules:

C0. $(s \to X_{init}) \in R_1$,
$(s' \to [_{A_{init}}]_{A_{init}}) \in R_1$,

C1. $(\$ \to \$ \in R_A)$, $A \in N_2$,

C2. $(\# \to \# \in R_A)$, $A \in N_2$,

C3. $(X \to X \in R_1)$, $X \in N_1$,

C4. $(X' \to X \in R_1)$, $X \in N_1$,

C5. $(\$ \to \lambda) \in R_1$,

to ensure that, if a computation halts, no trap symbol was introduced, all nonterminals from N_2 were rewritten, and the nonterminal from N_1 was erased. Once a membrane with label $A \in N_2$ is dissolved, object \$ is erased.

The above explanations makes rather easy the verification of the equality $Ps(\Pi) = \Psi_T(L(G))$. □

A similar increase of the power is entailed by the division rules from P systems with active membranes, with the interesting details that the universality is obtained without using membrane polarizations.

Theorem 3 $PsoP(a_0, b_0, c_0, d_0, e_0) = PsRE$.

Proof. Consider again a matrix grammar in the binary normal form, $G = (N, T, S, P, F)$ as above, with k matrices. We replace $(X \to \lambda, A \to x)$ by $(X \to f, A \to x)$, where f is a new symbol. As in the previous proof, we will represent each $A \in N_2$ in the label of a membrane, $[_A\$]_A$, and each $X \in N_1$ is represented as a corresponding object.

Additionally, for every nonterminal $A \in N_2$, we consider two membranes labelled by A, for every nonterminal $X \in N_1$ there is one membrane labelled by X, and one membrane labelled λ in the skin. These extra membranes are needed because the number of membranes with a given label can only increase via membrane division. To check if the number of membranes labelled by A is greater than two, symbols A_i, $i \in \{1, 2\}$, are used, which must immediately enter some membrane with label A, thus using two such membranes. At the same time, another object can enter a membrane labelled A if and only if there are more than two of them. Checking that there is more than one membrane labelled by X is done in a similar way, using the object X_1.

We construct the P system

$$\begin{aligned}
\Pi &= (O, H = \{1, \lambda\} \cup N_2 \cup N_1, \mu, X_{init}, \{w_h \mid h \in H\}, R),\\
O &= T \cup N_1 \cup \{X'', X', X'_1, X_1 \mid X \in N_1\} \cup \{A_1, A_2, A'_1, A'_2 \mid A \in N_2\}\\
&\cup \{m_r, m'_r, m_{r,1}, m_{r,2} \mid 2 \le r \le k\} \cup \{\$, \$', \$'', \#, f, g\},\\
\mu &= [_1[_{X_1}]_{X_1} \cdots [_{X_s}]_{X_s}[_{A_1}]_{A_1}[_{A_1}]_{A_1} \cdots [_{A_k}]_{A_k}[_{A_k}]_{A_k}[_{A_{init}}]_{A_{init}}[_\lambda]_\lambda]_1,\\
w_1 &= X_{init},\\
w_A &= \$,\ A \in N_1 \cup N_2,\\
w_\lambda &= \$'',
\end{aligned}$$

with the following rules:

(For a word x, we will denote by x_i the i-th symbol of x, or λ if $|x| < i$.) Each matrix $m_r : (X \to Y, A \to x)$, $Y \in N_1 \cup \{f\}$, is simulated by the rules:

A0. $[_1 X \to A_1 A_2 m_r]_1$,

A1. $[_1 m_r \to \#]_1$,

A2. $m_r[_A]_A \to [_A m_r]_A$,

A3. $[_A m_r]_A \to m'_r$,

A4. $[_1 m'_r \to m_{r,1} m_{r,2} Y'']_1$,

A5. $[_1 m_{r,i}]_1 \to [_1]_1 x_i$ if $x_i \in T$,

A6. $[_1 m_{r,i} \to \lambda]_1$ if $x_i = \lambda$,

A7. $m_{r,i}[_{x_i}]_{x_i} \to [_{x_i}\$']_{x_i}$ if $x_i \in N_1$,

Here, object m_r enters a membrane with label A, dissolves it while changing to m'_r, and evolves into $m_{r,1} m_{r,2} Y'$. The first two objects produced evolve into the representation of x (for each nonterminal of x, object $\$'$ enters a corresponding membrane by rule A7, duplicating it by rule C4 below), while the last one evolves into Y, thus finishing the simulation of m_r.

Each matrix $m_r : (X \to Y, A \to \#)$, is simulated by the rules:

B0. $[_1 X \to A_1 A_2 A'_1 A'_2 Y'_1 m_r m'_r]_1$,

B1. $[_1 m_r \to \#]_1$,

B2. $m_r[_Y]_Y \to [_Y m_r]_Y$,

B3. $[_Y m_r]_Y \to [_Y \$]_Y [_Y \$]_Y$,
B4. $m'_r[_A]_A \to [_A \#]_A$,
B5. $m'_r[_Y]_Y \to [_Y m'_r]_Y$,
B6. $[_Y m'_r]_Y \to Y$.

Here, if a membrane with label A exists, then the computation is blocked with the help of rule B4, otherwise m_r duplicates membrane labelled Y, and then m'_r enters one of these membranes, dissolving it and producing Y.

R also contains:

C0. $[_1 A_i \to \#]_1$, $A \in N_2$, $i \in \{1, 2\}$,
C1. $A_i[_A]_A \to [_A \$]_A$, $A \in N_2$, $i \in \{1, 2\}$,
C2. $[_1 X_1 \to \#]_1$, $X \in N_1$,
C3. $X_1[_X]_X \to [_X \$]_X$, $X \in N_1$,
C4. $[_X \$']_X \to [_X \$]_X [_X \$]_X$, $X \in N_1$,
C5. $[_A \# \to \#]_A$, $A \in N_2 \cup \{1\}$,
C6. $[_1 X'' \to X']_1$,
$[_1 X' \to X]_1$, $X \in N_1$,
$[_1 X'_1 \to X_1]_1$, $X \in N_1$,
C7. $[_1 A'_i \to A_i]_1$, $A \in N_2$, $i \in \{1, 2\}$.

The first four rules are auxiliary, used to "keep busy" the two membranes with label A and one membrane with label X when it is needed. Rule C5 is used to lead to an endless computation once the trap symbol is introduced. Rules C6 are used for synchronization.

To finish the computation, the following rules are included in R:

D0. $[_\lambda \$'' \to \$'']_\lambda$,
D1. $f[_\lambda]_\lambda \to [_\lambda f]_\lambda$,
D2. $[_\lambda f]_\lambda \to g$,
D3. $g[_A]_A \to [_A \#]_A$.

The first rule prevents the system from halting until a terminal matrix is applied, producing object f, entering membrane labelled λ by rule D1 and dissolving it by rule D2. The last rule guarantees that all nonterminals from N_2 are rewritten.

The Parikh equivalence of G and Π can be easily proved. □

Open questions. Is there an *upper bound* on the number of membranes in the starting membrane structure, needed for the proof of Theorem 3? What is the generative power of these systems without *dissolution*?

Remark 2 *It is easy to see that Theorem 3 also holds for P systems with multiplicities. Thus,* $PsOP(a_0, b_0, c_0, d_0, e_0) = PsRE$, *improving the result* $PsMAT \subset PsOP(a_0, b_0, c_0, d_0, e_0)$ *from [1]. Notice that* (c_0) *is only used in rules* A_5 *(to output the result).*

6 Conclusion

In this paper, a variant of P systems is defined which does not keep track of multiplicities of objects in the regions. If the number of membranes cannot grow, then these systems have the power of finite automata. However, the systems using membrane creation or membrane division are computationally complete. The last result also implies the computational completeness (in the Parikh set sense) of polarizationless P systems with active membranes with symbol-object multiplicities.

References

[1] A. Alhazov, L. Pan, Gh. Păun, Trading polarizations for labels in P systems with active membranes, *Acta Informatica*, to appear.

[2] F. Bernardini, V. Manca, P systems with boundary rules, in [6], 107–118.

[3] M. Cavaliere, Evolution-communication P systems, in [6], 134–145.

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

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

[6] Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), *Membrane Computing. International Workshop WMC-CdeA 2002, Curtea de Argeş, Romania, Revised Papers*, LNCS 2597, Springer-Verlag, Berlin, 2003.

[7] G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.