Polarizationless P Systems with One Active Membrane

Artiom Alhazov[1], Rudolf Freund[2]

[1] Institute of Mathematics and Computer Science, Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
E-mail: artiom@math.md
[2] Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
E-mail: rudi@emcc.at

Summary. The aim of this paper is to study the computational power of P systems with one active membrane without polarizations. For P systems with active membranes, it is known that computational completeness can be obtained with either of the following combinations of features: 1)two polarizations, 2)membrane creation and dissolution, 3)four membranes with three labels, membrane division and dissolution, 4)seven membranes with two labels, membrane division and dissolution.

Clearly, with one membrane only object evolution rules and send-out rules are permitted. Two variants are considered: external output and internal output.

1 Introduction

Membrane computing is a theoretical framework of parallel distributed multiset processing. It has been introduced by Gheorghe Păun in 1998, and has been an active research area since then, see [10] for the comprehensive bibliography and [6],[8] for a systematic survey. Membrane systems are also called P systems.

It has been shown in [4] (some results being improvements of the results from [1] and [3]) that the following P systems with active membranes are computationally complete: 1) with one membrane and two polarizations, as acceptors, 2) polarizationless ones with membrane creation and dissolution, 3) polarizationless ones starting with four membranes and three labels, 4) polarizationless ones starting with seven membranes and two labels.

The object of study of this paper is the family of P systems with one active membrane without polarizations. Similar questions for non-cooperative transitional P systems without any additional features have been addressed in [2].

2 Definitions

2.1 Formal Language Preliminaries

Consider a finite set V. The set of all words over V is denoted by V^*, the concatenation operation is denoted by $\bullet$ (which is written only when necessary) and the empty word is denoted by λ. Any set $L \subseteq V^*$ is called a language. For a word $w \in V^*$ and a symbol $a \in V$, the number of occurrences of a in w is written as $|w|_a$. The permutations of a word $w \in V^*$ are $\texttt{Perm}(w) = \{x \in V^* \mid |x|_a = |w|_a \text{ for all a } \in \text{V}\}$. We denote the set of all permutations of the words in L by $\texttt{Perm}(L)$, and we extend this notation to families of languages. We use FIN, REG, LIN, CF, MAT, CS, RE to denote finite, regular, linear, context-free, matrix without appearance checking and with erasing rules, context-sensitive, and recursively enumerable families of languages, respectively. The family of languages generated by extended (tabled) interactionless L systems is denoted by $E(T)0L$. The family of sets of numbers generated by forbidden random context multiset grammars is denoted by $NfRC$. For more formal language preliminaries, we refer the reader to [9].

Throughout this paper we use string notation to denote the multisets. When speaking about membrane systems, keep in mind that the order in which symbols are written is irrelevant, unless we speak about the symbols sent to the environment. In particular, speaking about the contents of some membrane, when we write $a_1^{n_1} \cdots a_m^{n_m}$ (or any permutation of it), we mean a multiset consisting of n_i instances of symbol a_i, $1 \leq i \leq m$.

2.2 P systems with One (Active) Membrane

We present the definition of a *P system with active membranes*, simplified for studying the generative power in case of one membrane.

$\Pi = (O, \mu = [\ ]_1, w_1, R_1, i_0)$, where

O is a finite set of objects,

w_1 is the initial multiset in region 1,

R_1 is the set of rules associated to membrane 1,

i_0 is the output region; when languages are considered, $i_0 = 0$ is assumed.

The rules of a membrane system have the forms $(a_0)\ [\ a \to u\ ]_1$ (evolution of an object), and $(c_0)\ [\ a\ ]_1 \to [\ ]_1 b$ (sending an object out, possibly renaming it), where $a, b \in O$ and $u \in O^*$.

The rules are applied in maximally parallel way: no further rule should be applicable to the idle objects, except rules of type (c_0) may be applied to at most one object at any step.

A *catalytic P system* (with one membrane) is a construct

$\Pi = (O, C, \mu = [\ ]_1, w_1, R_1, i_0)$, where

O is a finite set of objects,

C is a special subset of Owhose elements are called catalysts,

w_1 is the initial multiset in region 1,

R_1 is the set of rules associated to membrane 1,

i_0 is the output region; when languages are considered, $i_0 = 0$ is assumed.

The rules in R are either non-cooperative rules of the form $a \to (b_1, tar_1) \cdots (b_k, tar_k)$ with a and the b_i, $1 \leq i \leq k$, being from $O \setminus C$ and the $tar_i \in \{here, out\}$ being the targets for the corresponding symbols b_i, or catalytic rules of the form $ca \to c(b_1, tar_1) \cdots (b_k, tar_k)$ with $c \in C$.

A configuration of a P system is a construct which contains the information about the contents of the skin membrane as well as the sequence of objects sent out. A sequence of transitions between the configurations is called a computation. The computation halts when such a configuration is reached that no rules are applicable. In case of external output ($i_0 = 0$), as the result of a (halting) computation we may consider the *sequence* of objects sent to the environment; we denote it by $L(\Pi)$. Both in case of internal output ($i_0 = 1$) and in case of external output, we may consider as the result the vector of multiplicities of objects in region i_0, we denote it by $Ps(\Pi)$, or the total number of objects in region i_0, which we denote by $N(\Pi)$.

The family of P systems with one polarizationless active membrane may be denoted by $OP_1(a_0, c_0)$. The class of sets of numbers/vectors/words generated by a family $\mathbf{F}$ of P system is denoted by $N\mathbf{F}$, $Ps\mathbf{F}$ and $L\mathbf{F}$, respectively. We use a superscript *int* or *ext* when speaking about internal and external output, respectively, and we may omit subscript *ext* in the case of generating languages, i.e., external output is assumed for $L\mathbf{F}$.

Moreover, we may use a subscript T to denote terminal filtering of the result; in this case, a subset $T \subset O$ is additionally specified for Π, and the objects not belonging to T are not considered in the result. For example, the family of sets of vectors of non-negative integers generated internally by P systems with one polarizationless active membrane with terminal filtering are denoted by $Ps_T^{int}OP_1(a_0, c_0)$.

Example 1. To illustrate generation, consider the following P system:

$$\begin{aligned}\Pi &= (O = \{S, a, b, c, d, f\}, \mu = [\ ]_1, w_1 = a, R_1, i_0),\\ R_1 &= \{[\ S \to Sabcd\ ]_1,\ [\ S \to f\ ]_1,\\ &\quad [\ a\ ]_1 \to [\ ]_1 a,\ [\ b\ ]_1 \to [\ ]_1 b,\ [\ c\ ]_1 \to [\ ]_1 c\}.\end{aligned}$$

Object S produces objects a, b, c, d in arbitrary but equal amounts. Objects a, b, c are sent out in arbitrary order. Hence, if $i_0 = 1$ then $N(\Pi) = \mathbb{N}_1$ (i.e., the set of all positive integers), and if $i_0 = 0$ then $L(\Pi) = \bigcup_{n \geq 0} \mathtt{Perm}(a^n b^n c^n) = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}$.

P systems can be also viewed as acceptors. In that case, an input subalphabet Σ is additionally specified in the tuple defining P system before μ, and $i_0 = 1$ is the input region. An input multiset over Σ is additionally placed inside the membrane before the computation starts, and it is accepted if and only if the computation halts. The result $Ps_{acc}(\Pi)$ is the set of all accepted inputs, and the family of vector sets accepted by P systems with one active membrane is $Ps_{acc}OP_1(a_0, c_0)$.

3 Comparison with a Transitional Model: Catalytic P Systems with One Catalyst

The model of P systems with active membranes, for the case of one membrane, can be compared to the following case of transitional P systems: non-distributed P systems

with one catalyst. Indeed, for each P system with one active membrane, there exists a 1-catalytic non-distributed P system with the same behavior, as non-cooperative rules work equivalently in both models: $[\ A \to u\ ]_h$ is equivalent to $A \to u$, and sending out corresponds to particular rules with one catalyst, i.e., $[\ A\ ]_h \to [\ ]_h a$ corresponds with $cA \to c(a, out)$, or, if without restricting generality we assume the set of symbols that may appear inside the system to be disjoint from the set of symbols that may be sent to the environment, simply with $cA \to c(a, here)$.

Notice that for P system with external output, we may ignore the objects remaining inside the system when it halts (as explained in the next section), while for P systems with internal output, we should ignore the objects sent out. In this way, for the case of internal output, sending out corresponds to a catalytic erasing, while for the case of external output sending out corresponds to a catalytic renaming of a non-terminal symbol into a terminal symbol.

Hence, we can immediately conclude that

$$X^{\alpha}_{\beta}OP_1(a_0, c_0) \subseteq X_{\beta}OP_1(ncoo, cat_1) \text{ for } X \in \{N, Ps, L\},\ \alpha \in \{int, ext\},\ \beta \in \{-, T\},$$

where $\beta = -$ stands for not specifying a subscript.

One-catalytic P systems were investigated in [5], where some subclasses of P systems with one catalyst are defined and certain results on their generative power are presented. In particular, it was shown in [5] that $N_{-c}OP_1(wsepcat_1) = NREG$ and $N_{-c}OP_1(complcat_1) \subseteq NfRC$. Clearly, the corresponding restrictions might also be considered for polarizationless P systems with one active membrane, and such results can be claimed as upper bounds for the corresponding restrictions, e.g.,

$$NOP_1(wsep(a_0, c_0)) = NREG,$$

where the restriction of the *weak separation* can be reformulated for the model with active membranes as follows: the set O of objects is divided into three **disjoint** subsets O', O'' and O''', such that

- objects $a \in O'$ have no associated rules (they cannot evolve or be sent out, so if they are produced, they remain idle inside the system),
- objects $a \in O''$ have associated send-out rules, but no evolution rules,
- objects $a \in O'''$ have associated evolution rules, but no send-out rules.

It is worth mentioning that the additional requirement from [5] that the objects produced by a catalytic rule cannot undergo a non-cooperative rule is *automatically* satisfied after translation into the active membrane case, so the only restriction remaining in the case of weak separation is that a rule of type (a_0) and a rule of type (c_0) are not allowed to compete for the same object. This restriction means, for instance, that all objects that have associated send-out rules cannot evolve inside the system, they simply wait there until they are chosen to be sent out.

A different restriction considered in [5] is *complete* P systems (mentioned above as $complcat_1$). It can be reformulated in the model of polarizationless P systems with active membranes as follows: there is no object having associated rules of type (c_0) and no rules of type (a_0). This restriction means that no object is allowed to be temporarily idle; if it is not sent out, then it either evolves immediately, or remains idle throughout the computation. It follows that

$$NREG \subseteq NOP_1(compl(a_0, c_0)) \subseteq NfRC.$$

It is interesting to note that weak separation and completeness are, in some sense, two opposite requirements. While the latter one requires that *all* objects which can be sent out must evolve if they are not chosen to be sent out, the first special case requires that *no* objects which can be sent out are allowed to evolve. Of course, in the most general case there can be both kinds of objects which can be sent out.

4 External output

The first goal of this section is to present a reduction of any P system with one active membrane without polarizations and external output to an equivalent normal form. Then we will use this normal form to prove an upper bound result. We require the normal form mentioned above to satisfy the following conditions:

- Every object appears on the left side of some rule.
- The only erasing rule allowed is for the initial object; if so, the initial object does not appear on the right side of any rule. (If we have an initial multiset w, then we add the rule $S \to w$ where S is a new symbol now being the initial object.)

We approach this goal in a few stages. First, we remark that, without restricting generality, we may assume that no objects may remain inside the system when it halts. Indeed, let O_λ be the set of all objects that do not have associated rules. By adding rules $R_\lambda = \{[\ a \to \lambda\ ]_1 \mid a \in O_\lambda\}$, we make sure that there are no objects that do not have associated rules. On the other side, adding rules R_λ does not affect the result of a P system with external output, since preserving/erasing objects from O_λ has no alternatives, and it does not affect the environment.

Second, we remark that, without restricting generality, we may assume that the initial multiset consists of only one object, say S, which does not appear in the right side of any rule. Indeed, for a P system starting with a multiset (represented by) w, consider an equivalent P system starting with a multiset consisting of a new object S, and adding $R_S = \{[\ S \to w\ ]_1\}$ to R_1.

Third, we claim that for any P system satisfying the assumptions mentioned above, there exists a P system without erasing rules (except, possibly, for S).

Proof. Indeed, let us first add rules $R_t = \{[\ a \to \#\ ]_1 \mid ([\ a \to \lambda\ ]_1) \in R_1 \text{ or } a = \#\}$, where $\#$ is a new symbol, shared for all such reductions, so if it appears in a configuration, the system will never halt, and will therefore not produce any result. This transformation will certainly not affect the result of the system, since every new computation branch will not be productive, while the existing branches will not be affected (since by construction, one can *always* apply some other rule to a instead of trapping).

Second, compute the set O_λ of erasable objects as follows:

- Set O_λ to $\{a \in O \mid [\ a \to \lambda\ ]_1 \in R_1$,
- If $[\ a \to u\ ]_1$ is in R_1 and $u \in O_\lambda^*$, then add a to O_λ,
- Iterate the previous procedure until no more elements can be added to O_λ.

Third, replace each rule $[\ a \to u\ ]_1$ by rules $[\ a \to u'\ ]_1$, where the u' are obtained from u by removing (in all possible combinations) some objects from O_λ. This will again yield an equivalent system, because every symbol that could eventually be deleted does not have to be produced in the first place.

Fourth, remove all erasing rules. We claim that the resulting P system is still equivalent to the original P system. Indeed, any object (other than S) that should be erased, could be "pre-erased" by not producing it in the first place. However, any object that should evolve can evolve by other rules, and any object that should be sent out can be sent out (unless some competing object is sent out, in which case the simulation would not be correct, so the computation is discarded by producing symbol #). □

Corollary 1. $LOP_1(a_0, c_0) \subseteq CS$.

Proof. Indeed, the total number of objects (inside and outside the membrane) never decreases throughout the computation (except, possibly, for the empty word, generated in one step), and the length of the result matches the total number of objects when the system halts. □

We now proceed with the lower bound result.

Theorem 1. $LOP_1(a_0, c_0) \supseteq REG \bullet \texttt{Perm}(REG)$.

Proof. Consider an alphabet T and two arbitrary regular languages over T. Then there exist reduced regular grammars $G_1 = (N_1, T, P_1, S_1)$ and $G_2 = (N_2, T, P_2, S_2)$ generating them, such as $N_1 \cap N_2 = \emptyset$. We construct the following P system:

$$
\begin{aligned}
\Pi &= (O = N_1 \cup N_2 \cup T \cup T', \mu = [\ ]_1, w_1 = S_1, R_1),\\
T' &= \{a' \mid a \in T\},\\
R_1 &= \{[\ A \to aB\ ]_1 \mid (A \to aB) \in P_1\} \cup \{[\ A \to S_2\ ]_1 \mid (A \to \lambda) \in P_1\}\\
&\quad \cup \{[\ A \to a'B\ ]_1 \mid (A \to aB) \in P_2\} \cup \{[\ A \to \lambda\ ]_1 \mid (A \to \lambda) \in P_2\}\\
&\quad \cup \{[\ a' \to a'\ ]_1 \mid a \in T\} \cup \{[\ a\ ]_1 \to [\ ]_1 a,\ [\ a'\ ]_1 \to [\ ]_1 a \mid a \in T\}.
\end{aligned}
$$

The P system constructed above generates $L(G_1) \bullet L(G_2)$, except the symbols generated by the second grammars are produced in a primed form, and may undergo trivial rewriting for an arbitrarily long time before they are sent out, which ensures that after generating a word from $L(G_1)$, any permutation of a word from $L(G_2)$ may be generated. □

We now present a few closure properties.

Lemma 1. *The family $LOP_1(a_0, c_0)$ is closed under renaming morphisms.*

Proof. The statement follows from applying the renaming morphism to the send-out rules. □

Theorem 2. *$LOP_1(a_0, c_0)$ is closed under union.*

Proof. The closure under union follows from adding a new axiom and productions of non-deterministic choice between multiple axioms. □

5 Internal output

In this case the environment is no longer relevant: it does not matter which symbol is written in the right side of a send-out rule. The object sent out no longer affects the result, so sending out is equivalent to a sequential version of erasing.

Of course, we can generate $PsREG$ with rules of type (a_0) corresponding to the rules of a reduced regular grammar. Hence,

$$Ps^{int}OP_1(a_0, c_0) \supseteq PsREG.$$

Is it an *open question* whether non-semilinear number sets can be generated, see also the partial results transferred from the one-catalytic model, recalled in Section 3.

6 P systems with input

In this section we show that, not very surprisingly, for P systems with one polarizationless active membrane, their accepting power is even smaller than their generative power. More exactly, unless such a P system accepts all allowed inputs, it only accepts specific finite sets. We start by establishing some useful facts (we remind that we use $\subseteq$ to denote the submultiset relation, $\cup$ to denote the union of multisets, and $\setminus$ to denote the difference of multisets).

Lemma 2. *Let $\Pi \in OP_1(a_0, c_0)$ be a P system with alphabet O, let $[\ u\ ]_1 \Rightarrow [\ v\ ]_1\alpha$ in Π ($\alpha \in O \cup \{\lambda\}$) Then for every multiset $u' \subseteq u$, either $[\ u'\ ]_1$ is already a halting configuration, or there exists a multiset $v' \subseteq v$ and $\beta \in O \cup \{\lambda\}$ such that $[\ u'\ ]_1 \Rightarrow [\ v'\ ]_1\beta$ in Π.*

Proof. In a transition $[\ u\ ]_1 \Rightarrow [\ v\ ]_1\alpha$, one of three possible cases happen for every (copy of) object a in u:

- a is rewritten by some rule of Π into a (possibly empty) multiset, contributing to v;
- a is sent out by some rule of Π as α;
- a remains idle, contributing to v.

Note that v consists exactly of the resulting objects from the first case and the objects of the third case. More precisely, let the union of multisets of the right side rules for all copies of rewritten objects be v_r, and let the multiset of idle objects be v_i; then, $v = v_r \cup v_i$. By definition of the model, the second case was applied to at most one (copy of) an object in u. Also by definition of the model, for each object in the third case, there exist no rules to evolve it, except, possibly, send-out rules, in which case $\alpha \neq \lambda$.

We recall that u' may be obtained from u by erasing some (copies) of objects. Fix some correspondence of (copies of) objects in u' to objects in u, and consider a transition from u' by the same behavior of objects in u' as of objects in u:

- rewritten objects will yield some submultiset v'_r of v_r;
- β' will be produced in the environment, $\beta' = \alpha$ or $\beta' = \lambda$;
- idle objects will yield some submultiset v'_i of v_i.

It is obvious that these rules are applicable, and that $v'_r \cup v'_i \subseteq v$. Maximality also holds, except in one special situation: when $\alpha \neq \lambda$, but it was produced from a (copy of) an object not in u', while there exists at least one object b that was idle in a transition $[\ u\ ]_1 \Rightarrow [\ v\ ]_1\alpha$.

In this situation, one object b, instead of being idle, should be sent out as β, and the resulting multiset in the skin is $v' = v'_r \cup v'_i \setminus b$ (if this situation does not happen, we take $\beta = \beta'$ and $v' = v'_r \cup v'_i$).

Therefore, $[\ u'\ ]_1 \Rightarrow [\ v'\ ]_1\beta$ in Π if at least one (copy) of object from u' fell into the first or the second case, and otherwise $[\ u'\ ]_1$ is already a halting configuration. □

Lemma 3. *If $n \in N(\Pi)$, then also $n' \in N(\Pi)$ for any non-negative integer $n' \leq n$.*

Proof. Let the alphabet of Π be O, let the initial contents of the skin membrane of Π be w_1, and let the input subalphabet of Π be Σ. By definition of acceptance, a number n is accepted if there exists a halting computation in Π starting from configuration $[\ u\ ]_1$, for some $u \in w_1\Sigma^n$.

Consider the "sub-input" of only n' objects, i.e., $u' \in w_1\Sigma^n$ such that $u' \subseteq u$. If $[\ u\ ]_1$ is already halting, then so is $[\ u'\ ]_1$, so the statement of the lemma holds; now we assume the contrary: $[\ u\ ]_1 \Rightarrow [\ v\ ]_1\alpha$. By the previous lemma, in one step, either the computation with u' in the skin will immediately halt (and the statement of the lemma again holds), or there is a one-step transition $[\ u'\ ]_1 \Rightarrow [\ v'\ ]_1\beta$ with $v' \subseteq v$.

Iterating the application of the previous lemma, by induction, we conclude that there exists a computation starting from $[\ u'\ ]_1$ that will halt in at most as many step as the halting computation starting from $[\ u\ ]_1$ that we considered. Hence $n' \in N(\Pi)$. □

It follows that the accepted set of numbers is either $\mathbb{N}$, or empty, or it contains all integers less than or equal to the maximal accepted number, so accepting P systems with one polarizationless active membrane cannot be computationally complete, and P systems with one polarizationless active membrane are obviously weaker as acceptors than as generators:

$$N_{acc}OP_1(a_0, c_0) \subseteq \{\emptyset, \mathbb{N}\} \cup \{\{k \mid 0 \leq k \leq n\} \mid n \in \mathbb{N}\}.$$

In the rest of the section we show, by all necessary examples, that this inclusion is an equality:

$$\begin{aligned}
&\Pi_\emptyset = (O = \{a\}, \Sigma = \{a\}, \mu = [\ \ ]_1, w_1 = a, R_1 = \{[\ a \to a\ ]_1\}, i_0 = 1).\\
&\Pi_\mathbb{N} = (O = \{a\}, \Sigma = \{a\}, \mu = [\ \ ]_1, w_1 = \lambda, R_1 = \{[\ a \to \lambda\ ]_1\}, i_0 = 1).\\
&\Pi_n = (O = \{a_i \mid 0 \leq i \leq n\}, \Sigma = \{a_0\}, \mu = [\ \ ]_1, w_1 = \lambda, R_1, i_0 = 1), \text{ where}\\
&R_1 = \{[\ a_i \to a_{i+1}\ ]_1,\ [\ a_i\ ]_1 \to [\ \ ]_1 a_0 \mid 0 \leq i < n\} \cup \{[\ a_n \to a_n\ ]_1\}.
\end{aligned}$$

Clearly, $\Pi_\emptyset$ accepts nothing, since with any input it starts with at least one object, and carries out an infinite computation. On the other end of the spectrum, system $\Pi_\mathbb{N}$ accepts any input, by erasing it in one step and halting. Finally, we claim that system Π_n accepts exactly set $\{k \mid 0 \leq k \leq n\}$. Indeed, any object increments its index every step, unless the object is sent out, or the index reaches n (forcing an infinite computation). It is easy to see that at most n input objects may be sent out in this way; the system with input $(a_0)^k$ has a halting computation if and only if $k \leq n$.

Overall, we have established the following results:

$$\begin{aligned} REG \bullet \texttt{Perm}(REG) \subseteq LOP_1(a_0, c_0) &\subseteq CS, \\ Ps^{int}OP_1(a_0, c_0) &\subseteq PsREG, \\ N\alpha OP_1(wsep(a_0, c_0)) &= NREG,\ \alpha \in \{int, ext\}, \\ NREG \subseteq N\alpha OP_1(compl(a_0, c_0)) &\subseteq NfRC,\ \alpha \in \{int, ext\}, \\ N_{acc}OP_1(a_0, c_0) &= \{\{k \mid 0 \leq k \leq n\} \mid n \in \mathbb{N}\} \cup \{\emptyset, \mathbb{N}\}. \end{aligned}$$

7 Conclusions

In this paper we have considered the family of languages generated by polarizationless P systems with one active membrane. A normal form was given for external output case. It was than shown that the family of generated languages lies between $REG \bullet \texttt{Perm}(REG)$ and CS, and is closed under union and renaming morphisms. The exact characterization is an *open question*, but polarizationless P systems with one active membrane can be simulated by (and are, therefore, *at most* as powerful as) P systems with one catalyst, transferring two results on the generative power of two restricted classes, independently from the output region.

Then we also considered sets of vectors or numbers generated internally, as well as sets of vectors or numbers accepted by polarizationless P systems with one active membrane. Several questions about the families of these sets are still *open*, too.

Another possible generalization that can be considered is to also allow rules of type (b_0) to bring objects from the environment back to the skin. Note that such systems would still correspond to a subclass of 1-catalytic P systems, but some definitions would have to be revised, as well as all related results.

We have proved that accepting P systems with one polarizationless active membrane are not computationally complete, unlike those with two polarizations or like those with membrane creation and dissolution, or with multiple membranes and membrane dissolution.

The questions about the computational power of polarizationless P systems with active membranes with 2 and 3 membranes in the initial configuration are still *open*, as well as of polarizationless systems with less than 7 membranes and two labels, or of all polarizationless systems with only one label.

References

1. A. Alhazov: P Systems without Multiplicities of Symbol-Objects. *Information Processing Letters* **100**, 3, 2006, 124–129.
2. A. Alhazov, C. Ciubotaru, Yu. Rogozhin, S. Ivanov: The Family of Languages Generated by Non-Cooperative Membrane Systems. In: Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa: *Membrane Computing, 11th International Conference*, CMC11, Jena, Revised Selected Papers, Lecture Notes in Computer Science **6501**, 2011, 65–79.

3. A. Alhazov, R. Freund, Gh. Păun: Computational Completeness of P Systems with Active Membranes and Two Polarizations. In: M. Margenstern: *Machines, Computations, and Universality*, 4th International Conference, MCU 2004, Saint Petersburg, Revised Selected Papers, Lecture Notes in Computer Science **3354**, Springer, 2005, 82–92.
4. A. Alhazov, R. Freund, A. Riscos-Núñez: Membrane Division, Restricted Membrane Creation and Object Complexity in P Systems. *International Journal of Computer Mathematics* **83**, 7, 2006, 529–548.
5. R. Freund:
Special Variants of P Systems with One Catalyst in One Membrane. In: H. Leung, G. Pighizzini: 8th International Workshop on *Descriptional Complexity of Formal Systems* - DCFS 2006, Las Cruces, New Mexico, 2006. Proceedings, 2006, 250–258.
6. Gh. Păun: Membrane Computing. An Introduction, Springer, 2002.
7. Gh. Păun, G. Rozenberg, A. Salomaa: Membrane Computing with an External Output. *Fundamenta Informaticae* **41**, 3, 2000, 313–340.
8. Gh. Păun, G., Rozenberg, A. Salomaa, A. (eds.): *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
9. G. Rozenberg, A. Salomaa (eds.): *Handbook of Formal Languages*, 1-3 vol., Springer, 1997.
10. P systems webpage. `http://ppage.psystems.eu/`