Complexity Applications of Covering Rules in P Systems

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n 46020 Valencia, Spain
E-mail: jsempere@dsic.upv.es

Abstract

In this paper we use covering rules to explore several complexity aspects of P systems. First, we study the effect of covering rules over the description complexity of any P system. Then, we will use covering rules to introduce a speed-up time complexity result. Finally, we introduce a new problem that arises from the use of covering rules and we will define new complexity classes based on such problem.

1 Introduction

Membrane Computing [4] is a rapidly increasing research area motivated by some aspects of the biology of the cell and how these aspects can be adapted to formalize universal computational models that show high parallelism, distributed and cooperative computation and formal language (or r.e. number sets) acceptance or generation.

Several variants of P systems (as the main membrane computing model) have been proposed along the time. For instance, the importance of the catalysts on the evolution rules, the symport/antyport behavior of the membranes and the use of promoters/inhibitors have been studied, among other aspects, in order to produce different universal models of computation. We refer to [5, 3] for some of those variants.

A P system consists of a hierarchical finite set of regions where there are an undefined number of objects that react according to a previously defined set of rules. The reactions take part in every region in a parallel nondeterministic manner and the result of the reactions can be communicated to other regions by allowing the pass of objects from one region to a closest one through the membranes. In a previous work [9], we introduced some aspects about the influence of the external environment over the behavior of a P system. We proposed some differences between *persistent* and *nonpersistent* environments depending on the way in which the external information was introduced in the outer region (through the skin membrane). In the same work, we introduced a new kind of rules that could manage an undefined number of objects coming from the external environment every time unit. We named those rules *covering* rules.

In this work we initiate a study of the use of covering rules regarding to complexity aspects of P systems. Mainly, we will make use of covering rules to manage description complexity and time complexity of P systems.

The structure of this work is as follows. First, we will give the basic definitions and notation to be used in the sequel. We will formally define the notion of *covering rule*. Then, we will study the description complexity of P systems and how it can be reduced by using covering rules. This will be formalized under the Kolmogorov complexity framework. Finally, we will study the time complexity of P systems and we will provide a speed-up time result together with the definition of new complexity classes related to the membership problem of covering rules.

2 Basic Definition and Notation

Here, we will introduce some basic concepts from formal language theory according to [1, 7], and from membrane computing according to [4].

An alphabet Σ is a finite nonempty set of elements named symbols. A string defined over Σ is a finite ordered sequence of symbols from Σ. The infinite set of all the strings defined over Σ will be denoted by Σ^*. The empty string will be denoted by λ and Σ^+ will denote $\Sigma^* - \{\lambda\}$. A language L defined over Σ is a set of strings from Σ. L can be empty, finite or infinite. The number of strings that belong to a language L is its cardinality.

Now, we will introduce some basic concepts about P systems. A general P system of degree m, according to [4], is a construct

$$\Pi = (V, T, C, \mu, w_1, \cdots, w_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0),$$

where:

- V is an alphabet (the *objects*),
- $T \subseteq V$ (the *output alphabet*),
- $C \subseteq V, C \cap T = \emptyset$ (the *catalysts*),
- μ is a membrane structure consisting of m membranes,
- w_i, $1 \leqslant i \leqslant m$ is a string representing a multiset over V associated with the region i,
- R_i, $1 \leqslant i \leqslant m$ is a finite set of *evolution rules* over V associated with the ith region and ρ_i is a partial order relation over R_i specifying a *priority*.

  An evolution rule is a pair (u, v) (or $u \rightarrow v$) where u is a string over V and $v = v'$ or $v = v'\delta$, where v' is a string over

$$\{a_{here}, a_{out}, a_{in_j} \mid a \in V, 1 \leqslant j \leqslant m\}$$

and δ is an special symbol not in V (it defines the *membrane dissolving action*). From now on, we will denote the set $\{here, out, in_k : 1 \leqslant k \leqslant m\}$ by tar.

- i_0 is a number between 1 and m and it specifies the *output* membrane of Π (in the case that it equals to ∞ the output is read outside the system).

The language generated by Π in external mode ($i_0 = \infty$) is denoted by $L(\Pi)$ and it is defined as the set of strings that can be defined by collecting the objects that leave the system by arranging them in the leaving order (if several objects leave the system at the same time, then permutations are allowed). The set of numbers that represent the objects in the output membrane i_0 will be denote by $N(\Pi)$. Obviously, both sets $L(\Pi)$ and $N(\Pi)$ are defined only for *halting computations*. We suggest to the reader Păun's book [4] to learn more about P systems.

3 Covering Rules

Now, we will introduce a variant of P systems by defining a new kind of evolution rules that we will name *covering rules*. Observe that in general P systems, as described in the previous section, different rules can manage identical objects (e.g., $a \rightarrow bc$ and $a \rightarrow de$ will transform a objects into objects b, c, d, and e). Here, the application of a covering rule can manage the objects in an *exclusive* manner. The term *covering* refers to the situation in which the rule *covers* an undefined number of objects.

In addition, we can see that general P systems are systems with covering rules in which the languages used in the right and left parts of the evolution rules are composed only by languages with cardinality equal to 1.

We provide the formal definition of covering rules, as follows.

Definition 1. Let Π be a P system. We will say that r is a covering rule if $r : L_u \rightarrow L_v$ or $r : L_u \rightarrow L_v\delta$, where $L_u \subseteq V^*$ and $L_v \subseteq (V \times tar)^*$ and δ implies membrane dissolving.

Now, we will show how covering rules manage the objects of the region.

Example 1. Let $ab^* \rightarrow (c_{here})^*$ be a covering rule and *abab* be the set of objects of its region. Then, after applying the rule, we will obtain the set of objects *accc*.

In the previous example we have managed the objects in a *conservative* manner. That is, the number of objects after applying the rule does not decrease. The *non conservative* choice implies that the result of the rule application could be *a, ac, acc,* or *accc,* given that the object a or the objects b could be substituted by λ (which belongs to c^*), so they disappear.

Example 2. Let the following covering rules be in the same region: r_1 : $ab^+ \rightarrow (c_{here})^*$ and r_2 : $ab^+ \rightarrow (d_{here})^*$. Let us suppose that the objects in the region before applying the rules are *aabbb*. If we manage the rules in the *exclusive* mode, then the result will be *acccc* or *adddd* (both in conservative mode). That is, the selected rule r_1 or r_2 covers all the objects b.

If we apply the rules in non exclusive manner, then the combinatorics increase the number of results: we can obtain *cccdd* or *ccddd* or *acccc* or *adddd* depending on the number of objects b that every rule covers.

We can combine different ways of application of every rule (*exclusive* vs. *non exclusive* together with *conservative* vs. *non-conservative*). Furthermore, in the case that the rules are applied in non-conservative manner we can arrive to an *extremely non-conservative* mode. So, in example 3.1 the rule can be applied in a non-conservative manner by eliminating some objects, as explained before, or it can increase the number of objects so an undefined number of objects are presented at a given computation step.

All the mentioned ways of application of the covering rules imply that a *second degree of nondeterminism* appears in P systems. Obviously, general P systems are nondeterministic in the first sight. That is, whenever two or more rules can be applied at a given computation step, then the election of the rules to work is made in a non-deterministic manner, so all the combinatorics must be taken into account in order to study the different computation sequences. Here, the covering rules introduce a second degree of nondeterminism given that, first a rule is nondeterministically selected and then, if it is a covering rule, the result of its application is again nondeterministically produced. Let us illustrate this situation in the following example.

Example 3. Consider the rules $r_1 : ab \rightarrow cd$ (non-covering rule) and $r_2 : ab^+ \rightarrow e^+f^+g^+$. Let us suppose that the objects in the region are $aabbb$. Then, if rule r_1 is selected twice, the result is $ccddb$, if rule r_2 is selected and it works in the exclusive conservative manner, then the result can be $aeefg$ or $aeffg$ or $aefgg$. If rule r_2 works in non-exclusive manner the rule r_1 could be applied together with the covering rule. In the case that rule r_2 is applied in extremely non-conservative exclusive manner, then an infinite number of results can be obtained.

We can summarize all the application modes by means of the following definition.

Definition 2. Let Π be a P system, and $r : \alpha \rightarrow \beta$ a covering rule of the system. We will say that P works in

(a) **conservative mode** if the application of r will never decrease the number of selected objects in the system.

(b) **non-conservative mode** if the application of r can decrease the number of selected objects in the system.

(c) **exclusive mode**: if rule r is selected and applied, then it covers all the objects according to expression α and no object that belong to α remains free.

(d) **extremely non-conservative mode** if the result of applying the rule r is any string that belongs to β and the number of selected objects can be increased.

Limiting the Nondeterminism: Indexed Covering Rules

As mentioned before, the introduction of covering rules in P systems increases the non-determinism of the system. Now, we will introduce a variant of covering rules that attempts to reduce this non-determinism. For example, let us take the rule $ab^+c^+ \rightarrow (c_{here})^+(d_{here})^+(e_{here})^+$. There is no explicit correspondence between symbols of left-hand side and right-hand side. So, the objects *abbcc* could be transformed in *cddee* or *cccde* or *cdeee*, etc. (always in the case that the conservative mode be applied). That is, there is not knowledge to make correspondences between every pair of symbols from left and right sides. In order to control this situation we will introduce indexes to make this correspondence explicit.

Definition 3. Let Π be a P system. We will say that r is an indexed covering rule if $r : L_u \to L_v$ or $r : L_u \to L_v\delta$, where $L_u \subseteq (V \times \mathbb{N})^*$ and $L_v \subseteq (V \times tar \times \mathbb{N})^*$, δ implies membrane dissolving and $dom_{\mathbb{N}}(L_u) = dom_{\mathbb{N}}(L_v)$ [1].

Example 4. Let $a_1 b_2^+ c_3^+ \to (c_{here})_1^+ (d_{here})_2^+ (e_{here})_3^+$ be an indexed covering rule. The meaning of the rule is that every object a is substituted by at least one object c, every object b is substituted by at least one object d and every object c is substituted by at least one object e.

Obviously, different objects can collapse to a single one: the rule $a_1 b_1^+ \to (c_{here})_1$ means that one single object a together with an undefined positive number of objects b are replaced by the object c. Observe that the previous rule always works in non conservative mode.

4 Description Complexity

We will refer to the amount of information needed to describe a P system as its description complexity. In order to analyze the description complexity of any P system, some preliminary considerations will be needed: First, we assume that the number of rules of a given P system is a parameter of its description complexity. The membrane structure and the initial objects in every region will be taken into account too. The description of other components of the system, such as the output region, the alphabets, etc. will depend on the previous ones and will not be considered. We will study how covering rules influences the decreasing of the information needed to describe the system.

We will analyze only the rules of the system. The membrane structure and the initial objects will not be affected by the use of covering

[1]Given $L \subseteq (V \times \mathbb{N})^*$ or $L \subseteq (V \times tar \times \mathbb{N})^*$ we will denote by $dom_{\mathbb{N}}(L)$ the set of positive integers that appear in the description of L.

rules. Given that different rules can be compressed into one single covering rule, we can compress the information needed to describe the system, so we reduce its description complexity. First, we will show an example.

Example 5. Let the rules r_1 and r_2 be defined as $a \rightarrow b_{here}c_{here}$ and $a \rightarrow d_{here}e_{here}$. The rules r_1 and r_2 can be described as the indexed covering rule $a_1^*a_2^* \rightarrow (b_{here}c_{here})_1^*(d_{here}e_{here})_2^*$. Observe that in a conservative mode the effect of the covering rule is identical to the application of r_1 together with r_2 except for the multiplicities of the objects in the right-hand side of the rule.

Now, we wonder if the last transformation can be applied always or there is any requirement to apply so. Let us see the following lemma that will answer this question

Lemma 1. *Let Π_1 be a P system without covering rules. Then, there exists a P system Π_2 which is equivalent to Π_1 and which has a number of rules less than or equal to the number of rules of Π_1.*

Proof. We will construct Π_2 from Π_1 by compressing several rules at the same region into one single covering rule. Let us consider that in region i from Π_1 there exists the following covering rules with the same priority: $r_1 : \alpha_1 \rightarrow \beta_1, r_2 : \alpha_2 \rightarrow \beta_2, \cdots, r_n : \alpha_n\beta_n$. We will construct the following indexed covering rule in Π_2 with the same priority than r_j:

$$r_{1n} : (\alpha_1)_1^*(\alpha_2)_2^* \cdots (\alpha_n)_n^* \rightarrow (\beta_1)_1^*(\beta_2)_2^* \cdots (\beta_n)_n^*$$

We can observe that r_{1n} produces the same result than the application of the rules $r_1, r_2, \cdots, r_n$ if Π_2 works in conservative mode: the objects covered from r_{1n} are the same than the ones covered by the set of referred rules and the transformation of the objects are the same too (given that Π_2 works in conservative mode). There is only one aspect that makes the behavior of Π_2 different from Π_1: whenever the rule r_{1n} is applied, the objects from β_j could be multiplied an undefined number of times. The example 4.1 shows this behavior: if $a \rightarrow b_{here}c_{here}$ is applied then one symbol a is changed by two symbols b and c, while $a^* \rightarrow (b_{here}c_{here})^*$ could change one symbol a by any pair of symbols b and c due to the conservative working mode. □

The last proof shows how covering rules help to decrease the number of rules in P systems. The problem concerned to the conservative mode can be solved by introducing the *primitive working mode* which will be defined as follows.

Definition 4. Let Π be a P system, and $r : \alpha \rightarrow \beta$ a covering rule of the system. We will say that r is primitive if there exists a finite set of non covering rules which is equivalent to r (i.e. that produce the same result). The non covering rules that originate the covering one will be called *constructors*. We will say that P works in *primitive mode* if the application of the primitive covering rules is equivalent to the application of their constructors.

Obviously, the P system Π_2 proposed in lemma 4.1 is equivalent to Π_1 if primitive mode is applied. The covering rules that we have constructed are always primitives given that they are constructed from non covering rules.

Example 6. Let r be the covering rule defined as $a^+b^+ \rightarrow (cd)^+$. We can observe that r is not primitive. There is no set of constructors equivalent to r. The rule r changes at least one object a together with at least one object b by objects c and d. Observe that we cannot fix the relation between the number of object a and b. That is, the non covering rule $a^ib^j \rightarrow (cd)^k$ has not the same effect as the covering one.

We have showed that covering rules help to decrease the number of rules in P, but the question about the information needed to describe the system still remains open. The amount of information needed to describe the set of covering rules could be even greater than the information needed to describe non covering rules. We will analyze this question by introducing a description complexity measure such as the *Kolmogorov complexity*.

Introducing the Kolmogorov Complexity

Kolmogorov complexity [2] is a measure of the amount of information needed to describe objects. Here, binary strings from previously encoded objects can be used in order to make comparisons between their sizes. The main ingredient of Kolmogorov complexity is its algorithmic approach to describe the objects. Informally, the complexity of any object is the size of the program that outputs any of its descriptions.

Kolmogorov complexity was used in a previous work to measure the description complexity of formal languages [8]. Here, we will use it to study the effect of covering rules over the description complexity of P systems.

We will deal with the description complexity of multisets. Furthermore, here we are concerned with the sets of numbers that multisets represent. So, first, we assume that the Kolmogorov complexity of a multiset is the size of any description of the set of numbers that it represents. For a given multiset M, the Kolmogorov complexity of M will be denoted by $\mathcal{K}(M)$. It can be defined as the size of the minimum program that outputs the sequence of numbers represented by M. Obviously, given that this set of numbers can be managed by P systems (if they are r.e.) we can define a Kolmogorov complexity version linked to P systems (instead of any other effective programming system). We propose the following definition to make so.

Definition 5. Let M be a multiset of objects over a previously defined alphabet Γ. The Kolmogorov complexity of M related to P systems will be denoted by $\mathcal{K}_P(M)$ and is defined as $\mathcal{K}_P(M) = min\{|\Pi| : N(\Pi) = N(M)\}$. Here, $N(\Pi)$ denotes the set of numbers computed by the P system Π, $N(M)$ denotes the set of numbers related to the multiset M and $|\Pi|$ denotes the size of the P system

The size of any P system, as we have mentioned before, can be measure by the size of the initial objects at every region, its membrane structure size and the size of the rules at every region. We will analyze every one of these aspects

- The cardinality of the objects at every region in the initial configuration will measure its size. We will denote it as $size(Obj)$.

- The size of the membrane structure can be measure by taking into account the number of the regions. Let us suppose that m denotes it.

- The size of the rules can be measure by taking into account their *radius* and the size of the right-hand side. So, we propose the following expression to give an effective size measure. We will denote by R the set of rules in the system and by R_i the set of rules at region i

$$size(R) = \sum_{(u \to v) \in R_i} (radius(u) + |v|)$$

So, a first bound of the Kolmogorov complexity of any multiset M can be stated as

$$\mathcal{K}(M) \leqslant \mathcal{K}_P(M) \leqslant size(Obj) + m + size(R) + \mathcal{O}(1)$$

Now, we are concerned about the decreasing effect of covering rules over the Kolmogorov complexity. Here, we will pay our attention to the set of rules given that, as showed in lemma 4.1, neither the membrane structure nor the alphabet is affected by them.

Let us suppose that the set of non covering rules $r_1 : \alpha_1 \to \beta_1, \cdots, r_n : \alpha_n \to \beta_n$ is substituted by the covering rule $r_{1n} : (\alpha_1)_1^*(\alpha_2)_2^* \cdots (\alpha_n)_n^* \to (\beta_1)_1^*(\beta_2)_2^* \cdots (\beta_n)_n^*$. The size of the set of non covering rules is measured as

$$\sum_{1 \leqslant i \leqslant n} (radius(\alpha_i) + |\beta_i|)$$

which is an upper bound of the size of the covering rule. There are cases where the size of the covering rule does not arrive to this upper bound. Let us see an example

Example 7. Let r_1 and r_2 be non covering rules defined as $a \to b_{here}$ and $a \to c_{here}$. Then the covering rule $r_{12} : a^* \to (bc)^*_{here}$ is equivalent to r_1 and r_2 if primitive working mode is used. Observe that the size of the rules r_1 and r_2 is greater than the size of the rule r_{12} due to the fact that we have summarized two objects a into one single object a.

We will denote the Kolmogorov complexity of a multiset M related to P systems with covering rules as $\mathcal{K}_{Pc}(M)$. The following inequality holds

$$\mathcal{K}(M) \leqslant \mathcal{K}_{Pc}(M) \leqslant \mathcal{K}_P(M)$$

5 Computational Time Complexity

Inspired by the classic result from complexity theory, which establishes that a constant speed-up time factor can always be applied to recognize any formal language (provided some initial conditions), we can propose a similar idea for P systems.

Given a P system Π, the computing sequence of the system, after n steps, is denoted by $C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_n$, where C_i is the description of the system at instant i. The description of the system at any given moment is defined by the membrane structure and the set of objects and rules at every region. We can collect the set of objects in every region for every instant, so we have a language L_n^r that denotes the set of objects in region r presented during n steps. In the same sense, we can denote by L_f^r the set of objects of the region r in the nth step. So, the covering rule $L_n^r \longrightarrow L_f^r$ summarizes in one step all the history of the region r during n steps. In this sense, we are speeding-up the time consumed by the system by a constant factor in a way similar to the classical result for Turing machines [1].

In order to formalize this idea we will propose a definition of the time complexity associated with P systems. Observe that complexity classes have been studied within these models and related to classical parallel families such as NC [4]. We will study only some aspects of time complexity without referring to other computation models.

Definition 6. We will denote by $TIME_P(f)$ the family of languages processed by P systems in at most $f(n)$ steps, where n is the size of the objects presented at the initial configuration in every region.

The last definition implies some assumptions that we made: we consider that every computation step is made in one time unit (a different situation will be considered later) and we also consider that the f is a function from integers to integers that measures the complexity growth rate. Now we can see how covering rules help to the speed-up time computation in P systems, at least when some types of rules are applied.

We will say that the rule $\alpha \longrightarrow \beta$ is *pure recursive* if all the objects that appear in α also appear in β with an increasing number of times. For example, $ab \longrightarrow a_{here}a_{here}b_{out}b_{out}$ is pure recursive given that the objects a and b in the left-hand side also appear in the right-hand side and the number of every object is increased. Given $\beta = (a_1)^{k_1}(a_2)^{k_2}\cdots(a_n)^{k_n}$, we will define $\rho(\beta, i) = (a_1)^{(k_1)^i}(a_2)^{(k_2)^i}\cdots(a_n)^{(k_n)^i}$

If a P system works only with primite rules, then we can speed-up the time complexity by a constant factor: Let Π be a P system accepting L. We will set $c = \lceil\frac{1}{k}\rceil$ where k will be defined as the number of computation steps that we will summarize through covering rules. The computing sequence of P in the first k steps is $C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_k$. Here we can summarize this sequences in just one single step by introducing a covering rule $\alpha \rightarrow \beta$ where α is defined by the language $\{w^0, w^1, \cdots, w^k\}$ such that w^i is the string that represent the objects at step i in the current region and β is defined by the language $\{y^0, y^1, \cdots, y^k\}$ where $y^j = (x^j)_{here}$. This transformation is applied to every region of the system. Observe that now, the application of the last covering rules summarizes the first k steps in just a single one.

Now, we will summarize the rest of the $f(n)$ steps computation sequence as follows: for every pure recursive rule $r : \alpha \rightarrow \beta$ we will define the languages $L_{r1} = \{\alpha^i : 1 \leqslant i \leqslant k\}$ and $L_{r2} = \{\rho(\beta, i) : 1 \leqslant i \leqslant k\}$ and the covering rule $L_{r1} \rightarrow L_{r2}$. Observe that the previous rule summarizes in one single step what rule r could perform in k steps.

So, we can summarize the first k steps by using the initial covering rules and then, by using the rules $L_{r1} \rightarrow L_{r2}$ we can summarize the rest of k steps. As a result of that, the time complexity of the system is $\lceil\frac{f(n)}{k}\rceil$ which corresponds to $c \cdot f(n)$, where c is a constant factor.

The Membership Problem Appears

One assumption that we have made before is not realistic when working with covering rules. We have assumed that every computation step takes one time unit. Obviously, the application of non covering rules can be measured in such sense, given that the only work to carry out is checking the appearance of the objects of every rule in order to apply their transformation and it can be performed in constant time.

The case of covering rules is quite different. If we take a covering rule $\alpha \rightarrow \beta$ then the application of the rule needs some additional time in order to check if the objects of the region belong to the language defined by α. Here, the status complexity of α is crucial in order to measure the time needed to apply a computation step. For example, if α is a regular language then only linear time is needed to check if the rule can be applied. In the opposite case, if α is a NP-complete language, then, probably, an exponential time will be needed to perform every computation step. So, here the membership problem for the languages involved in the covering rules is crucial to define the time complexity

We will formalize this by introducing a new complexity definition.

Definition 7. We will denote by $TIME_P(f, g)$ the family of languages processed by P systems in at most $f(n)$ steps with covering rules with complexity $g(n)$, where n is the size of the objects presented at the initial configuration in every region.

We can use families of functions instead of functions f and g. So, for example, $TIME_P(poly, exp)$ will denote the family of languages which can be processed in polynomial time by using covering rules defined by languages of exponential complexity. So, $TIME_P(C_1, C_2)$ will denote the family of languages processed by P systems with functions $f \in C_1$ and $g \in C_2$.

6 Conclusions

In this work we have used covering rules to give some results of the complexity of P systems. First, covering rules help to reduce the description complexity of P systems given that they can compress the information that the system holds. Second, covering rules help to reduce in a constant factor the time complexity of P systems. However, the use of covering rules implies a new problem concerning the computational complexity of the membership problem associated with the rules. Here, we have introduced a new definition of complexity classes by taking into account that problem. In the future we will explore the relationships between such complexity classes.

Acknowledgement

Work partially supported by the Ministerio de Ciencia y Tecnología under project TIC2003-09319-C03-02.

References

[1] Hopcroft, J.; Ullman, J. *Introduction to Automata Theory, Languages and Computation*, Addison Wesley Publishing Co., 1979.

[2] Li, M.; Vitànyi, P. *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, 1993.

[3] Martín-Vide, C.; Mauri, G.; Păun, Gh.; Rozenberg, G.; Salomaa, A. (eds.), *Membrane Computing. International Workshop WMC-2003*, LNCS **2933**, Springer-Verlag, 2004.

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

[5] Păun, Gh.; Rozenberg, G.; Salomaa, A.; Zandron, C. (eds.), *Membrane Computing. International Workshop WMC-CdeA 2002*, LNCS **2597**, Springer-Verlag, 2003.

[6] Rogers Jr, H. *Theory of Recursive Functions and Effective Computability*, MIT Press, 1987.

[7] Rozenberg, G.; Salomaa, A. (eds.), *Handbook of Formal Languages, vol. 1*, Springer-Verlag, 1997.

[8] Sempere, J. M. A note on the equivalence and complexity of linear grammars, *Grammars*, **6**, 2 (2003), 115-126.

[9] Sempere, J. M. P systems with external input and learning strategies, *Proceedings of the Workshop on Membrane Computing WMC03*, LNCS **2933**, Springer-Verlag (2004), 341–356.