Covering Rules in P Systems: Some Preliminary Ideas [1]

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 propose a new kind of rules inside the regions of a P system. We have called them *covering rules* due to the fact that, if selected, they can manage all the objects of the region in an exhaustive manner (i.e., they *cover* all the objects of the region). First, we propose the formal definition of the rules and different ways of using them. This will introduce a second degree of nondeterminism in the complete behavior of a given P system. We will introduce an effective way to reduce the nondeterminism by defining *indexed covering rules*. Finally, we will initiate a study of several language families characterized in terms of the covering rules language families.

1 Introduction

Membrane Computing [3] 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 [4, 2] 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 [7], 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

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

(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 and how they can be managed to simulate different aspects of P systems proposed by other authors. We will make use of covering rules to take into account only the presence of objects without taking into account the number of objects. Also, they will be useful to simulate the equilibrium between regions or dealing with the descriptional complexity of the system.

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* and we will propose some simulation tasks for different aspects of P systems. Then, we will introduce a language setting framework by defining the sets of languages generated/accepted by different types of P systems depending on the use of different kind of covering rules. Finally, we will introduce some aspects to be studied in the near future about descriptional complexity, characterization of nonrecursive languages, etc.

2 Basic Definition and Notation

Here, we will introduce some basic concepts from formal language theory according to [1, 6], and from membrane computing according to [3].

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 [3], 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, $1 \leq i \leq m$ is a string representing a multiset over V associated with the region i,
- R_i, $1 \leq i \leq m$ 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 $u \to v$) where u is a string over V and $v = v'$ or $v = v'\delta$, where v' is a string over

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

and δ is an special symbol not in V (it defines the *membrane dissolving action*). From now on, we will denote the set $\{here, out, in_k : 1 \leq k \leq m\}$ 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 [3] 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 3.1 *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 3.1 *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 3.2 *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 non-deterministically produced. Let us illustrate this situation in the following example.

Example 3.3 *Consider the rules $r_1 : ab \rightarrow cd$ (non-covering rule) and $r_2 : ab^+ \rightarrow e^+f^+g^+$. Let us suppose that the present 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 3.2 *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 no "a priori" 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 3.3 *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 $dom_{\mathbb{N}}(L_u) = dom_{\mathbb{N}}(L_v)$* [2].

Example 3.4 *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 Some Applications of Covering Rules

Once we have introduced covering rules and some variants, we will show some applications of these rules to several aspects of P systems design and related topics.

4.1 Decreasing the Description Complexity

First, we assume that the number of rules of a given P system is a parameter of its descriptional complexity. So, several rules can be compressed into one single covering rule. In this sense we are compressing the information needed to describe the system, so we reduce its descriptional complexity. Here is an example.

Example 4.1 *Let the rules r_1 and r_2 be defined as $a \to b_{here} c_{here}$ and $a \to d_{here} e_{here}$. The rules r_1 and r_2 can be described as the indexed covering rule $a_1^* a_2^* \to (b_{here} c_{here})_1^* (d_{here} e_{here})_2^*$. Observe that in a conservative mode the effect of the covering rule is identical to the application of r_1 together with r_2.*

4.2 Controlling the Predominance of Objects

Other aspect that come from biology is the fact that in several reactions that happen in the cell, what is important is not the exact quantities of objects involved in the reaction, but some relationship between them. So, the equilibrium between substances is important to the functioning of the cell. For example, a given reaction can happen only if the total amount of a substance, let say a, is greater than the total amount of substance b. Not the exact quantities matter, but the broken equilibrium. We can easily formalize such a situation by using covering rules.

Example 4.2 *Let the covering rule r be defined as $\{a^n b^m : n \geq m\} \to \{a^n_{here} b^m_{here} : n < m\}$. This rule changes the relationship between the objects a and b. That is, whenever the number of objects a is greater than or equal to the number of objects b, the equilibrium is changed by adding more objects b than objects a.*

[2] 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.

4.3 Putting Thermodynamic Equilibrium to Work

In the last Brainstorming Week on Membrane Computing (Sevilla, 2-7 february 2004) G. Ciobanu proposed an initial work on thermodynamics equilibrium between adjacent regions of a membrane system. The passage of objects from one region to a neighboring one through protein channels depends not only on the presence of several objects in the region but on the number of objects in the neighboring regions in order to preserve some thermodynamic-like equilibrium between regions. Here, we propose a modification of covering rules that take into account the objects of adjacent regions to maintain the equilibrium of the system. So, we go beyond the kind of equilibrium described in the previous section.

Definition 4.1 *Let Π be a P system of degree m. We say that rule r from region k is an inter-region covering rule if $r : L_u \rightarrow L_v$ or $r : L_u \rightarrow L_v\delta$, where $L_u \subseteq (V \times \{1, \cdots, m\})^*$ and $L_v \subseteq (V \times tar)^*$, and δ denotes the membrane dissolving.*

Example 4.3 *Let the inter-region covering rule r be $(a^j)^+(b^k)^+ \rightarrow (b_{here})^+$. Now, let us suppose that r belongs to the region k that has a neighboring region j. Then, if more than one object a is presented in region j and more than one object b is presented in region k then an undefined number of objects b are produced in region k depending on the conservative or extremely conservative functioning choice. On the other hand, the objects in region j are not modified by this rule (i.e., object transformations are only permitted for the rules of every region in which the objects are presented).*

Obviously, inter-region covering rules also admit the indexed version in which every object is replaced by a predefined one by using the adequate indexes.

4.4 Towards a Speed-up Result

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 $L_n^r \rightarrow L_f^r$ 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].

5 Introducing Formal Language Characterizations

Finally, we would like to introduce another topic related to covering rules. In the examples that we have presented in the previous sections we have used different languages L_u and L_v. Most of them are just regular languages described by regular expressions. We have not restricted in any sense the use of other language classes for defining L_u and L_v.

A natural question that arises from covering rules is the relationship between the language classes used to define the rules and the acceptance or generation power of P systems. We will take Chomsky's hierarchy [1] as our framework to introduce this topic. So, we will denote by $SING$, FIN, CF, CS, REC and RE the classes of *singleton*[3], finite, context-free, context sensitive, recursive and recursively enumerable languages. The relationship between those classes is the following

$$SING \subset FIN \subset REG \subset CF \subset CS \subset REC \subset RE.$$

We denote by $N_{cr}(class)$ the family of sets of numbers computed by P systems with covering rules defined by languages that belong to the language family *class*. In the same sense $L_{cr}(class)$ will refer to the family of languages instead of the family of sets of number.

A first result that is obvious from the definitions is that $N_{cr}(SING)$ defines the same family as general P systems without covering rules, given that the rules of general P systems can be defined as covering rules with singleton languages.

We can introduce the working modes defined in section 3 to refer to classes of languages or of sets of numbers. We denote by $N_{cr}(class, mode)$ (or $L_{cr}(class, mode)$) the family of sets of numbers (or of languages) computed by P systems with covering rules that belongs to *class* working in the specified *mode*. We denote by **ce**, **cne**, **nce**, **ncne ence encne** the working modes conservative and exclusive, conservative and non-exclusive, non-conservative and exclusive, non-conservative and non-exclusive, extremely non-conservative and exclusive, and extremely non-conservative and non-exclusive, respectively. For example, $N_{cr}(SING, ce)$ can be viewed as the family of sets of numbers where all the rules have priorities (that is, whenever a rule is selected then it covers as much objects as possible in a way similar to working in the exclusive mode).

Last, we wonder about $N_{cr}(RE, *)$. In this case, we can use covering rules such as $L \rightarrow 1_{out}$ where L is a recursively enumerable language. Observe that a region containing such kind of rules can act as an *oracle* in classical recursion theory [5]. This opens a new problem which has been maintained hidden up to this moment: the membership problem associated to the rules. Whenever we have a covering rule in any region, the problem of whether or not the rule can be applied over the objects of the region is established in terms of the membership problem. This problem did not appear in general P systems due to the fact that membership problem for singleton languages has a constant (or at most linear) complexity. Nevertheless, if we avoid this aspect to measure the complexity and decidability of the system we can characterize nonrecursively enumerable languages in $N_{cr}(RE, *)$. In the opposite case, we should include the membership complexity in order to define the precise complexity of the system.

6 Conclusions and Future Research

In this work we have defined a new kind of rules associated to P systems. There are a lot of new topics related to covering rules which will be studied in the next future: the relationship between working modes, the application of covering rules to simulate other topics of P systems such as the use of inhibitors/promoters and catalysts, the thermodynamic equilibria between regions, etc. The language characterization of the rules will probably define a hierarchy of P systems which will go beyond nonrecursively enumerable languages. In addition, language characterization will include the membership problem associated to

[3]Singleton languages have the cardinality equal to one.

the rules in order to measure the time complexity of the system. These aspects will be explored in future works.

References

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

[2] C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, A. Salomaa (Eds.). *Membrane Computing. International Workshop WMC-2003. LNCS*, Vol. 2933. Springer, 2004.

[3] G. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

[4] G. Păun, G. Rozenberg, A. Salomaa, and C. Zandron (Eds.). *Membrane Computing. International Workshop WMC-CdeA 2002. LNCS*, Vol. 2597. Springer, 2003.

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

[6] G. Rozenberg, A. Salomaa (Eds.). *Handbook of Formal Languages* Vol. 1. Springer, 1997.

[7] J.M. Sempere. P systems with external input and learning strategies. *Proceedings of the Workshop on Membrane Computing WMC03. LNCS* Vol. 2933, pp 341–356. Springer, 2004.