Computing Using Signals: From Cells to P Systems

Ioan I. ARDELEAN
Centre of Microbiology
Institute of Biology of the Romanian Academy
Splaiul Independenţei 296
PO Box 56-53, Bucureşti 79651, România
E-mail: ioan.ardelean@ibiol.ro

Matteo CAVALIERE
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: matteo.cavaliere@estudiants.urv.es

Dragoş SBURLAN
Department of Computer Science
Ovidius University of Constanţa
Bd. Mamaia 124, Constanţa, România
E-mail: dsburlan@univ-ovidius.ro

Abstract. In cell biology one of the fundamental topic is the study of how biological signals are managed by cells. Signals can arise from inside the cell or from the external environment and the correct answer to certain signals is essential for bacteria to survive in a certain environment. Starting from these biological motivations we consider a model of P systems where the computation is controlled by signals which move across the regions. In particular, we consider Signals-Based P systems where the symbol-objects cannot be moved and the rules can be activated/inactivated using a finite number of signals (signal-promoters) moved across the membranes; differently from standard P systems using promoters, in our case promoters cannot be created during the computation. After discussing the biological motivations we show how this model becomes universal when it uses one catalyst, and a bounded number of signal-promoters.

1 Introduction

In cell biology it is known that many chemical reactions in the cell are catalyzed by the presence of the respective enzyme (in other words, the enzyme permits the chemical rule to happen; it is possible to get more information on this topic consulting [8]). On the other hand, in bacteria, the enzyme (protein) can be activated/inactivated during the cellular process (in other words, an inactivated enzyme is not able to catalyze the corresponding reaction).

In bacteria *enzyme activation/inactivation*, as well as other important processes, are controlled by covalent modification of proteins (so called post-translational *covalent modifications of proteins*), [12].

The covalent modification involves the attachment of objects (chemical substances) at different positions along the string represented by the protein (i.e., the enzyme). The attached object could belong to different type of substances such as phosphoryl or methyl (details can be found in [1]). One basic fact in biology is that the attachment of the object (either phosphoryl, methyl or other chemicals) occurs at specific places along the protein; during the cellular processes the substance (for example, phosphoryl) travels along the cell and it is attached to the enzyme where such enzyme must be activated. We can imagine the substance like a *promoter* that travels along the regions of the cell and activates the corresponding enzyme (and then the corresponding catalyzed reaction) when it is attached/unattached from the enzyme.

In what follows we will briefly describe how the phosphoryl movement can control the process of enzyme activation/inactivation in Escherichia Coli.

The enzyme activation/inactivation by using covalent bond of phosphate (as in the case of Escherichia Coli) was first described in mammals (liver) around half a century ago when Fisher and Krebs showed that an enzyme involved in metabolism was regulated by the addition (reaction called phosphorylation) or the removal (reaction called dephosphorylation) of phosphoryl, suggesting that reversible phosphorylation could control enzyme activity.

Since then the study of protein phosphorylation has flourished in Biology, many scientists being involved in the study of protein phosphorylation and its biological significance; for their pioneering work Fisher and Krebs received the Nobel Prize in 1992.

One classical example of enzyme activation/inactivation by covalent attaching of phosphoryl occurs in Escherichia Coli for *isocitrate dehydrogenase*. Isocitrate dehydrogenase is an enzyme which in the active state takes away two atoms of hydrogen (thus its name "dehydrogenase") and one molecule of carbon dioxide from a chemical called isocitrate; thus isocitrate is converted to another chemical called 2-oxoglutarate. The enzyme isocitrate dehydrogenase is *active when phosphoryl is NOT attached* on it. The enzyme is inactivated by attaching the phosphoryl and the inactivated enzyme is not able to perform the conversion of isocitrate to 2-oxoglutarate.

The bond of phosphoryl to isocitrate dehydrogenase is catalyzed by an enzyme called kinase whereas the removal of phosphoryl from phosphorylated isocitrate dehydrogenase is catalyzed by a so called phosphatase.

In particular, phosphate is attached by the kinase at a very precise position within the protein, exactly at the level of the $113rd$ aminoacid; these reactions are reversible and they enable the cell to either activate (by de phosphorylation) or to inactivate (by phosphorylation) the enzyme isocitrate dehydrogenase. The idea of this process is described in Figure 1.

The phosphoryl movement can also control the so-called *two-component regulatory system* present, for example, in Escherichia Coli. A two-component regulatory system is a system composed by two proteins: a sensor (S) and a response regulator (RS). Such system responses to environmental signals (*stimuli*) like changes in oxygen concentration, light intensity, starvation, water activity, and so on (for more details it is possible to consult [11]). The responses of a two-component regulatory system are essential for bacterial cells (as well as other types of cells) to survive in a given environment (for example, in Escherichia Coli, it seems to exist around 50 different two-component regulatory systems).

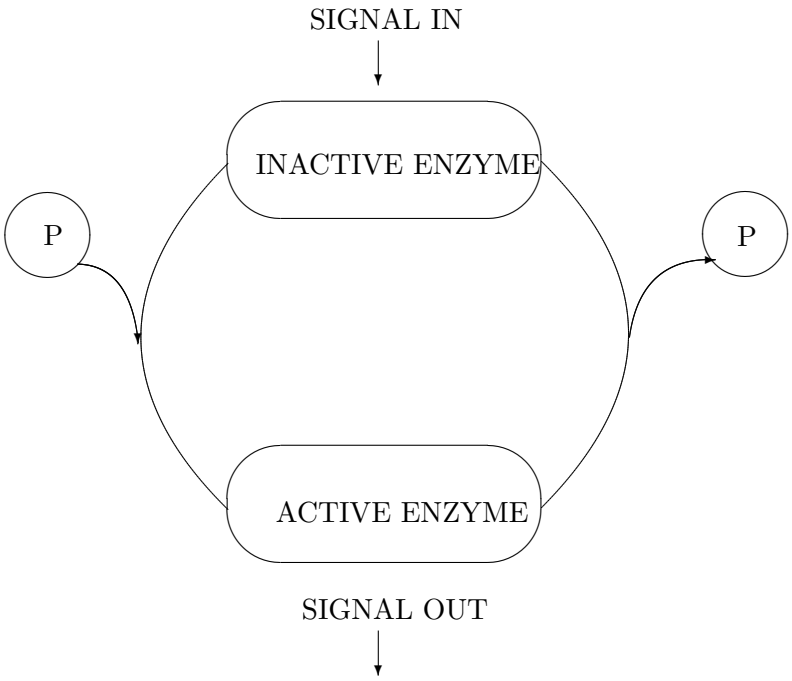

Figure 1: Enzyme is activated/deactivated removing/adding phosphoryl

The responses of a two-component regulatory systems are based, again, on the phosphorylation/ dephosphorylation of proteins described before for enzyme activation/inactivation (for details about two-component regulatory systems the reader can consult [1, 11]).

Two-component regulatory systems and enzyme activation/inactivation process are important because they illustrate how within the same cell (in this case, Escherichia Coli) the attaching of the same object, phosphoryl (that moves around the cell), can modulate (activate/deactivate) different functions such as the conversion of izocitrate to 2-oxoglutare (in the case of enzyme activation/inactivation) or it can change the proportion of some transport proteins at the plasma membrane (in the case of a two-component regulatory systems).

Using (some of) the biological motivations described above a new model of P systems, called *Signals-Based P system*, can be introduced: it uses symbol-objects (chemical substances) and chemical reactions (evolution rules) present in the regions; the symbol-objects evolve using the evolution rules but they cannot be moved; the rules present in the regions can be activated/deactivated using signals called *signal-promoters*. The signal-promoters do not participate in any evolution rule but they are able to move around the regions of the system (they play the role of the phosphoryl described before).

In other words, this new model can be considered like the classical P systems using promoters but with two restrictions: the symbol-objects cannot be moved and the signal-promoters cannot be created but only moved across the membranes (at the beginning of the computation a fixed number of signal-promoters is given in the regions; for some aspects the idea of signal-promoters recall the concept of mobile catalyst introduced in [6]). Informally, we can say that in a Signals-Based P system the computation is realized only

moving signals around the regions (from here "computing using signals" in the title of the paper) and executing the indicated evolution rules. The model considered is related with the idea of considering signal transduction in cells as computational process (for instance, see [9]).

In this preliminary paper we introduce the formal definition of Signals-Based P systems and we prove that they are universal using one catalyst and a bounded number of signal-promoters. When using non cooperative rules and two promoters, Signals-Based P systems are able to generate at least the Parikh sets of ET0L languages. In the final section several open problems are also proposed. In the following sections we suppose the reader familiar with the basic knowledge of P systems and in particular with P systems using promoters/inhibitors (for details the reader can consult [2] and [7]).

2 Signals-Based P Systems: Definition

A Signals-Based P system (in short, an SB P system), of degree $m \geq 1$, with symbol–objects is a construct

$$\Pi = (V, C, P, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, R'_1, \ldots, R'_m, i_o),$$

where:

- V is the alphabet of Π; its elements are called *objects*;
- $C \subseteq V$ is the set of catalysts;
- $P \subseteq V$ is the set of signal-promoters;
- μ is a membrane structure consisting of m membranes labeled $1, 2, \cdots, m$;
- w_i, $1 \leq i \leq m$, specify the multisets of objects present in the corresponding regions i at the beginning of a computation;
- R_i, $1 \leq i \leq m$, are finite sets of simple evolution rules over V associated with regions $1, 2, \ldots, m$ of μ; these evolution rules are of the form $a \to v$ or $ca \to cv$, where a is an object from $V \setminus \{C \cup P\}$ and v is a string over $V \setminus \{C \cup P\}$;
- R'_i, $1 \leq i \leq m$, are finite set of signaling rules over P associated with regions $1, 2, \ldots, m$ of μ; these signaling rules are of the form $a \to v|_{p_{tar}}$ or $ca \to cv|_{p_{tar}}$, where a is an object from $V \setminus \{C \cup P\}$, v is a string over $V \setminus \{C \cup P\}$, $p \in P$, $tar \in \{here, in_j, tar\}, j \in \{1, \cdots, m\}$;
- i_o is a number between 0 and m and specifies the output membrane of Π (if $i_0 = 0$ then the environment is used for the output).

A configuration of a Signals-Based P system is described using the m-tuple of multiset of objects, present in the m regions of the system. To each region a finite number of objects (among them, signal promoters and catalysts) is associated, and a finite number of simple evolution rules and of signaling rules. The m-tuple $(w_1, w_2, \cdots, w_m)$ is the initial configuration of Π.

A transition between two configurations is governed by the *mixed application* of the s imple evolution rules R_i and of the signaling rules R'_i, for each $1 \leq i \leq m$. A sequence of transitions of configurations is called *computation*.

Simple evolution rules are rules with all targets fixed as "here" (see [3]). Signaling rules are simple evolution rules, promoted by some signal-promoter p. When a signaling rule $a \to v|_{p_{tar}}$ (or $ca \to cv|_{p_{tar}}$) is applied then the object a is transformed into the objects specified by v and the promoter p is sent into the (adjacent) region specified by tar.

In every region the signal-promoters are present in the *set sense*, i.e. we cannot have more than one copy of the same signal-promoter in one region.

All objects that can be "subject" of the rules of the sets $R_i, R'_i, 1 \leq i \leq m$, have to evolve by such rules.

There is no difference between simple evolution rules and signaling rules: both are applied in the non-deterministic maximally parallel manner.

The system continues parallel steps until there remain no applicable rules in any region of the system.

Then the system halts (the computation is *successful*), and we consider the number of objects contained in the output region i_o as the result of the computation of Π. This way to have a computation in a Signal-Based P system is called the *mixed approach*.

During the computation it can happen that two signaling rules, promoted by the same promoter p, and with different targets, are applied: in this case a *conflict* appears and the computation is considered not successful.

We use the following notation

$$PsSBP_m(\alpha, j), \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\},$$

to denote the family of sets of vectors of natural numbers generated by SB P systems with at most m membranes, j signal-promoters, simple evolution rules and signaling rules that can be non-cooperative ($ncoo$), or catalytic (cat_k), using at most k catalysts (as usual $*$ is used if the corresponding number of membranes, signal-promoters or catalysts is unbounded).

3 Using Non-Cooperative Rules: Simulating Lindenmayer Systems

In this section we show how a Signal-Based P systems, using non-cooperative rules and three signal-promoters are able to generate at least the Parikh sets of ET0L languages.

First we recall the basic notions about Lindenmayer systems (for a complete information we suggest [10]).

An ET0L system is a construct $G = (\Sigma, T, H, w')$, where the components fulfill the following requirements: Σ is the alphabet. $T \subseteq \Sigma$ is the terminal alphabet. H is a finite set of of finite substitutions $H = \{h_1, h_2, \cdots, h_t\}$ (t is the number of tables); each $h_i \in H$ can be represented by a list of context-free rules $A \to x$, such that $A \in \Sigma$ and $x \in \Sigma^*$ (this list for h_i should satisfy that each symbol of Σ appears as the left side of some rule in h_i). $w' \in \Sigma^*$ is the axiom.

G defines a derivation relation $\Rightarrow$ by $x \Rightarrow y$ iff $y \in h_i(x)$, for some $1 \leq i \leq t$, where h_i is interpreted as a substitution mapping.

The language generated by G is $L(G) = \{w \in \Sigma^* \mid w' \Rightarrow^* w\} \cap T^*$, where $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$.

The family of languages generated by ET0L systems is denoted by $ET0L$.

It is known, [10], that for each $L \in ET0L$, there exist an ET0L system G', with only 2 tables, such that $L = L(G')$.

We also need to present the following normal form for ET0L systems.

Lemma 3.1 *(Normal Form) For each $L \in ET0L$ there is an extended tabled Lindenmayer system $G = (\Sigma, T, H, w')$ with 2 tables ($H = \{h_1, h_2\}$) generating L, such that the terminals are only trivially rewritten: for each $a \in T$ if $(a \to \alpha) \in h_1 \cup h_2$ then $\alpha = a$.*

Of interest for the purpose of this paper are the following result

$$CF \subset ET0L \subset CS \subset RE.$$

where by CF, CS, and RE we denote the families of languages generated by context-free, context-sensitive, and type 0 grammars, respectively.

Theorem 1 $PsSBP_2(ncoo, 3) \supseteq PsET0L$.

Proof. Given an ET0L system $G = (\Sigma, T, H, w')$ with 2 tables ($H = \{h_1, h_2\}$) in the normal form described before generating L, we construct an SB P system Π generating the Parikh set of L (actually, we remove the trivial productions from h_1 and h_2). Let us denote $N' = \Sigma \setminus T$.

We take

$$\Pi = (V, C, P, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, R'_1, \ldots, R'_m, i_o),$$

where

- $V = \Sigma \cup \{\#, S'\}$;
- $C = \emptyset$;
- $P = \{p', p'', q\}$;
- $\mu = [_1[_2]_2]_1$;
- $w_1 = Sp'p''q, w_2 = w'$;
- $R_1 = \{S' \to S\}$;
- $R_2 = \{\# \to \#\}$;
- $R'_1 = \{S \to S'_{|p'_{in_2}}, S \to S'_{|p''_{in_2}}, S \to S'_{|q_{in_2}}\}$;
- $R'_2 = \{u \to v_{|p'_{out}} \mid u \to v \in h_1\} \cup \{u \to v_{|p''out} \mid u \to v \in h_2\}$
  $\cup \{N \to \#_{q_{here}} \mid N \in N'\}$;
- $i_o = 2$.

The system Π works in the following way. In region 2 (the output region) are simulated the applications of the rules of the first table h_1 or of the second table h_2 over the symbol-objects corresponding to a sentential form.

In region 1 the object S is changed to S' using, in a non deterministic way, one of the three signaling-rules in R'_1. If the rule $S \to S'_{|p'_{in_2}}$ is applied, then the signal-promoter p' is sent to region 2; in the same way, if the rule $S \to S'_{|p''_{in_2}}$ is applied, then the signal-promoter p'' is sent to region 2. When the signal-promoter p' is sent to region 2, then the rules of h_1 are activated and then, possibly, applied; when the signal-promoter p'' is sent

to region 2, then the rules of h_2 are activated (the presence of the signal-promoters p' and p'' guarantees that the rules of the first and second table are not applied in a mixed way in region 2).

To stop the movements of signal-promoters between the two regions the rule $S \to S'_{|q_{in_2}}$ must be applied in region 1. In this case the signal-promoter q is sent into region 2 and this activates the rules $N \to \#_{q_{here}} \mid N \in N'\}$ present in that region (checking in this way that, when the computation halts, only objects corresponding to terminal symbols have been obtained in region 2).

Therefore, the system Π generates, in the output region, exactly the Parikh set of $L(G)$.

□

4 Using Catalytic Rules: Universality

In this section we prove that, if we use one catalyst and a (bounded) number of signal-promoters, then Signal-Based P systems become universal. The proof is based on simulating matrix grammars with appearance checking.

4.1 Matrix Grammars

A *matrix grammar with appearance checking* is a construct $G = (N, T, S, M, C)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and C is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either (1) $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or (2) $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in C. The rules of a matrix are applied in order, possibly skipping the rules in C that cannot be applied – therefore we say that these rules are applied in the *appearance checking* mode.

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$.

By MAT_{ac} we denote the families of languages generated by matrix grammars with appearance checking. It is known that $MAT_{ac} = RE$.

We say that a matrix grammar with appearance checking (ac) $G = (N, T, S, M, F)$ is in the Z-binary normal form if $N = N_1 \cup N_2 \cup \{S, Z, \#\}$ with these three sets mutually disjoint and the matrices in M of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, \mid x \mid \leq 2$
3. $(X \to Y, A \to \#)$, with $X \in N_1, Y \in N_1 \cup \{Z\}, A \in N_2$,
4. $(Z \to \lambda)$

There is only one matrix of type 1, and F consists of all rules $A \to \#$ appearing in matrices of type 3, and if a sentential form in G contains Z, it is of the form Zw, with $w \in (T \cup \{\#\})^*$.

Lemma 4.1 *For each language $L \in RE$ there is a matrix grammar with appearance checking in the Z-binary normal form such that $L = L(G)$.*

Theorem 2 *$PsSBP_2(cat_1, t) = PsRE$, for some $t \in \mathbf{N}$.*

We only show that $PsSBP_2(cat_1, t) \supseteq PsRE$ (the reverse inclusion is based on Turing-Church thesis).

Take a grammar $G = (N, \{a\}, S, M, F)$ in the Z-binary normal form, in the standard notation, presented before. Also, let us consider that $\mid M \mid = n$ and let us enumerate the matrices of M; moreover let us consider that we are not in the trivial case, i.e., there exists:

- one rule $(S \to X_0 A_0) \in M$;
- $j \geq 1$ rules of type $(X_i \to Y_i, A_i \to x_i) \in M$, $1 \leq i \leq j$;
- $k \geq 1$ rules of type $(X_i \to Y_i, A_i \to \#) \in M$, $j+1 \leq i \leq j+k = n-2\}$, with $(A_i \to \#) \in F$;
- one rule of type $(Z \to \lambda)$.

We will construct a Signals-Based P system $\Pi_{signals}$ which simulates any derivation of the grammar G. Formally we define

$$\Pi_{signals} = (V, C, P, \mu, w_1, w_2, R_1, R_2, 0),$$

where:

- $V = \{M_0, M_5, M_6, N_0, N_1, N_2\} \cup \{M_{i,t} \mid 1 \leq i \leq j,\ 1 \leq t \leq 4\} \cup$
  $\{M_{i,t} \mid j+1 \leq i \leq j+k,\ 1 \leq t \leq 3\} \cup C \cup P$;
- $C \subseteq V = \{c\}$ the set of catalysts;
- $P \subseteq V = \{R, Q, \overline{Q}, W\} \cup \{P_i \mid 1 \leq i \leq k+l\} \cup \{\overline{P}_i \mid 1 \leq i \leq k+l\} \cup \{R_i \mid 1 \leq i \leq l\}$ the set signaling promoters;
- $\mu = [_1[_2\ ]_2]_1$;
- $w_1 = \{P_i, \overline{P}_i \mid 1 \leq i \leq j+k\} \cup \{Q, \overline{Q}, M_0, R, W\}$
  $w_2 = \{c, N_0, W\} \cup \{X_0, A_0\}$.

The sets of rules, associated to the regions, are constructed in the following way.

1. For a matrix $m_i : (X_i \to Y_i, A_i \to x_i)$ with $1 \leq i \leq j$ we add to the sets R_1, R_1' and R_2' the following sequences of rules:

to R_1 :	to R_1' :	to R_2' :
$M_{i,4} \to M_5$	$M_0 \to M_{i,1}\mid_{P_{i_{in}}}$	$cX_i \to cY_i\mid_{P_{i_{out}}}$
$M_6 \to M_0$	$M_{i,1} \to M_{i,2}\mid_{Q_{in}}$	$N_0 \to N_1\mid_{Q_{out}}$
	$M_{i,2} \to M_{i,3}\mid_{\overline{P}_{i_{in}}}$	$N_1 \to \#\mid_{P_{i_{out}}}$
	$M_{i,3} \to M_{i,4}\mid_{\overline{Q}_{in}}$	$cA_i \to cx_i\mid_{\overline{P}_{i_{out}}}$
	$M_5 \to M_6\mid_{R_{in}}$	$N_1 \to N_2\mid_{\overline{Q}_{out}}$
		$N_2 \to \#\mid_{\overline{P}_{i_{out}}}$
		$N_2 \to N_0\mid_{R_{out}}$

2. For a matrix $m_i : (X_i \to Y_i, A_i \to \#)$ with $j+1 \leq i \leq j+k$ we add to the sets R_1, R_1' and R_2 the following sequences of rules:

to R_1 :	to R_1' :	to R_2' :
$M_{i,2} \to M_{i,3}$	$M_0 \to M_{i,1}\|_{P_{i_{in}}}$	$cX_i \to cY_iT_0\|_{P_{i_{out}}}$
	$M_{i,1} \to M_{i,2}\|_{Q_{in}}$	$N_0 \to N_1\|_{Q_{out}}$
	$M_{i,3} \to M_4\|_{\overline{P}_{i_{in}}}$	$N_1 \to \#\|_{P_{i_{out}}}$
	$M_4 \to M_0\|_{W_{in}}$	$T_0 \to T_1\|_{\overline{P}_{i_{out}}}$
		$cA_i \to c\#\|_{\overline{P}_{i_{out}}}$
		$N_1 \to N_0\|_{\overline{P}_{i_{out}}}$
		$T_1 \to \lambda\|_{W_{out}}$

3. For the matrix $m_{n-1} : (Z \to \lambda)$ we add to the sets R_1 and R_2' the following sequences of rules:

to R_1 :	to R_2' :	
$M_0 \to M_\#\|_{D_{in}}$	$Z \to \lambda\|_{D_{out}}$	
	$X \to \#\|_{D_{out}}$	$\forall X \in N \setminus \{Z\}$

4. Rule $\# \to \#$ is added to R_2;

5. No other rules are added to the sets R_1, R_2, R_1' and R_2'.

Proof. Let us now discuss how the system works by having initially an overview of the simulation and coming back to the details later on. At the beginning, region 1 contains the signaling-rules used to choose nondeterministically a certain matrix to have to be applied, while the region 2 contains the evolution rules that simulate the application of the matrix selected in region 1.

Informally, the whole computation is based on an exchange of signal-promoters between the two regions which coordinates the process by activating certain rules, at certain steps.

The P system uses a bounded number (depending on the grammar G) of signal-promoters.

In region 2 we will have rules that handle the correct application of the selected matrix, but also rules which manage the case when the selected matrix (of type 2 or type 3) cannot be simulated.

In this last case (if a "wrong" matrix has been selected) the symbol # is generated (therefore the computation will never stop since the presence of the evolution rule $\# \to \#$).

In the derivation of a matrix grammar, a matrix is non-deterministically chosen and it is applied on the current sentential form. The process starts from the axiom S and is iterated up to the moment when all the symbols in the sentential form are terminal symbols or no matrix is applicable anymore despite the fact that in the sentential form there still exist nonterminals. One can modify the grammar in an equivalent one such that every possible derivation leads to a terminal sentential form, or, never halts.

A similar mechanism will be used in our simulation: a matrix will be randomly chosen (by means of the non-determinism) in region 1 and, if it is possible, then it will be simulated in region 2 and such process will be iterated; if the chosen matrix cannot be simulated, then the P system will cycle forever. Finally, when the object Z is produced in region 2

then we either stop the computation – in the case that the computation in G was terminal, or we use forever the rule $\# \rightarrow \#$. Since in the Z-binary normal of the grammar G there exists only one rule of type $(S \rightarrow X_0 A_0) \in M$, we can start our simulation directly by placing the objects X_0 and A_0 inside region 2.

In order to select a certain matrix, the object M_0 is non-deterministically changed into:

1. $M_{i,1}$, $1 \leq i \leq j$, by the rule $M_0 \rightarrow M_{i,1}|P_{i_{in}}$ if a matrix not used in appearance checking mode was chosen (matrices of type $(X_i \rightarrow Y_i, A_i \rightarrow x_i)$);

2. $M_{i,1}$, $j+1 \leq i \leq j+k$, by the rule $M_0 \rightarrow M_{i,1}|P_{i_{in}}$ if a matrix used in appearance checking mode was chosen (matrices of type $(X_i \rightarrow Y_i, A_i \rightarrow \#)$);

3. $M_\#$ by the rule $M_0 \rightarrow M_\#|D_{in}$

In the first case the object M_0 is rewritten into $M_{i,1}$, $1 \leq i \leq j$, and the signal-promoter P_i is sent into region 2.

The signal-promoter P_i corresponds to the matrix i of the grammar G, and when it is present the process of applying the rules of such matrix is started.

At the beginning the system attempts to execute the first rule of the matrix $X_i \rightarrow Y_i$. This will be done in our simulation by the rule $cX_i \rightarrow cY_i|_{P_{i_{out}}}$ in R'_2.

The catalyst is used to inhibit the parallelism of the system such that if the rule is applied, then it is applied only once in one step. Let us consider that the rule is indeed applied. Then, between regions 1 and 2 an exchange of signals takes place: signal-promoter Q enters region 2 ($M_{i,1} \rightarrow M_{i,2}|_{Q_{in}}$), while signal-promoter P_i comes back to region 1 ($cX_i \rightarrow cY_i|_{P_{i_{out}}}$). Since the signal-promoter P_i there is not anymore in region 2, the rule $cX_i \rightarrow cY_i|_{P_{i_{out}}}$ cannot be executed. Now, the system is ready to try to simulate the next rule of the matrix. In particular, the signal-promoter $\overline{P}_i$ (which corresponds to the second rule of the matrix i) enters into region 2, being sent by the rule $M_{i,2} \rightarrow M_{i,3}|_{\overline{P}_{i_{in}}} \in R'_1$. In the meantime, the signal-promoter Q returns to its initial region 1. Such signal-promoter was used only to promote the rule $N_0 \rightarrow N_1|_{Q_{out}}$. The object N_1 is used when we have tried without success to simulate the matrix i. Then, since the signal-promoter P_i is still in region 2, the rule $N_1 \rightarrow \#|_{P_{i_{out}}}$ is executed and then the trap symbol $\#$ is generated and the computation will never stop.

With a similar mechanism, the simulation of the application of the second rule is done.

If the simulation of a matrix works good then the initial configuration is re-established in region 1, while in region 2 we will have the symbol objects corresponding to the new sentential form obtained. In this way we can iterate this process again choosing in a non-deterministic way the next matrix to be applied. Therefore the system Π generates in the output region exactly the Parikh set of $L(G)$.

Below are the tables which represent the computations made by the P system considering both regions (here, only the significant computational steps of an iteration are presented). Each cell of the table contains the current configuration (for a given step and a certain region) and the rules that can be applied to the current multiset.

	*Region*1	*Region*2
step 1	$P_i, \overline{P}_i, Q, \overline{Q}, M_0, R$ $M_0 \to M_{i,1}\|_{P_{i_{in}}}$	$c, N_0, X_i, A_i?$
step 2	$\overline{P}_i, Q, \overline{Q}, M_{i,1}, R$ $M_{i,1} \to M_{i,2}\|_{Q_{in}}$	$c, P_i, N_0, X_i, A_i?$ $cX_i \to cY_i\|_{P_{i_{out}}}$
step 3	$P_i, \overline{P}_i, \overline{Q}, M_{i,2}, R$ $M_{i,2} \to M_{i,3}\|_{\overline{P}_{i_{in}}}$	c, Q_i, N_0, Y_i, A_i $N_0 \to N_1\|_{Q_{out}}$
step 4	$P_i, Q, \overline{Q}, M_{i,3}, R$ $M_{i,3} \to M_{i,4}\|_{\overline{Q}_{in}}$	$c, \overline{P}_i, N_1, Y_i, A_i$ $cA_i \to cx_i\|_{\overline{P}_{i_{out}}}$
step 5	$P_i, \overline{P}_i, Q, M_{i,4}, R$ $M_{i,4} \to M_5$	$c, \overline{Q}, N_1, Y_i, x_i$ $N_1 \to N_2\|_{\overline{Q}_{out}}$
step 6	$P_i, \overline{P}_i, Q, \overline{Q}, M_5, R$ $M_5 \to M_6\|_{R_{in}}$	c, N_2, Y_i, x_i
step 7	$P_i, \overline{P}_i, Q, \overline{Q}, M_6$ $M_6 \to M_0$	c, N_2, R, Y_i, x_i $N_2 \to N_0\|_{R_{out}}$

Table 1. The simulation of matrices with rules not in appearance checking mode; the matrix is correctly applied.

For a failed simulation of the matrix i, the following table shows how the trap symbol # is generated. Once it is generated it cannot be removed. Therefore it does not matter what will happen with the rest of objects since the computation will never halt.

	*Region*1	*Region*2
step 1	$P_i, \overline{P}_i, Q, \overline{Q}, M_0, R$ $M_0 \to M_{i,1}\|_{P_{i_{in}}}$	$c, N_0, A_i?$
step 2	$\overline{P}_i, Q, \overline{Q}, M_{i,1}, R$ $M_{i,1} \to M_{i,2}\|_{Q_{in}}$	$c, P_i, N_0, A_i?$
step 3	$\overline{P}_i, \overline{Q}, M_{i,2}, R$ $M_{i,2} \to M_{i,3}\|_{\overline{P}_{i_{in}}}$	$c, P_i, Q_i, N_0, Y_i, A_i?$ $N_0 \to N_1\|_{Q_{out}}$
step 4	$Q, \overline{Q}, M_{i,3}, R$ $M_{i,3} \to M_{i,4}\|_{\overline{Q}_{in}}$	$c, P_i, \overline{P}_i, N_1, Y_i, A_i?$ $N_1 \to \#\|_{P_{i_{out}}}$ $cA_i \to cx_i\|_{\overline{P}_{i_{out}}}?$

Table 2. The simulation of matrices with rules not in appearance checking mode; the first rule of the matrix cannot be applied.

The following table describes the computation that happens when the second rule of matrix i cannot be simulated (since the corresponding object A_i is missing).

	*Region*1	*Region*2
step 1	$P_i, \overline{P}_i, Q, \overline{Q}, M_0, R$ $M_0 \to M_{i,1}\vert_{P_{i_{in}}}$	$c, N_0, X_i, A_i?$
step 2	$\overline{P}_i, Q, \overline{Q}, M_{i,1}, R$ $M_{i,1} \to M_{i,2}\vert_{Q_{in}}$	$c, P_i, N_0, X_i, A_i?$ $cX_i \to cY_i\vert_{P_{i_{out}}}$
step 3	$P_i, \overline{P}_i, \overline{Q}, M_{i,2}, R$ $M_{i,2} \to M_{i,3}\vert_{\overline{P}_{i_{in}}}$	c, Q_i, N_0, Y_i, A_i $N_0 \to N_1\vert_{Q_{out}}$
step 4	$P_i, Q, \overline{Q}, M_{i,3}, R$ $M_{i,3} \to M_{i,4}\vert_{\overline{Q}_{in}}$	$c, \overline{P}_i, N_1, Y_i, A_i?$
step 5	$P_i, Q, M_{i,4}, R$ $M_{i,4} \to M_5$	$c, \overline{P}_i, \overline{Q}, N_1, Y_i, A_i?$ $N_1 \to N_2\vert_{\overline{Q}_{out}}$
step 6	$P_i, Q, \overline{Q}, M_5, R$ $M_5 \to M_6\vert_{R_{in}}$	$c, \overline{P}_i, \overline{Q}, N_2, Y_i, A_i?$ $N_2 \to \#\vert_{\overline{P}_{i_{out}}}$

Table 3. The simulation of matrices with rules not in appearance checking mode; the second rule of the matrix cannot be applied.

The appearance checking case is described by the following tables:

	*Region*1	*Region*2
step 1	$P_i, \overline{P}_i, Q, M_0$ $M_0 \to M_{i,1}\vert_{P_{i_{in}}}$	$c, N_0, X_i, A_i?, W$
step 2	$\overline{P}_i, Q, M_{i,1}$ $M_{i,1} \to M_{i,2}\vert_{Q_{in}}$	$c, P_i, N_0, X_i, A_i?, W$ $cX_i \to cY_iT_0\vert_{P_{i_{out}}}$
step 3	$P_i, \overline{P}_i, M_{i,2}$ $M_{i,2} \to M_{i,3}$	$c, Q_i, N_0, Y_i, T_0, A_i?, W$ $N_0 \to N_1\vert_{Q_{out}}$
step 4	$P_i, \overline{P}_i, Q, M_{i,3}$ $M_{i,3} \to M_4\vert_{\overline{P}_{i_{in}}}$	$c, N_1, Y_i, T_0, A_i?, W$
step 5	P_i, Q, M_4	$c, \overline{P}_i, N_1, Y_i, T_0, A_i?, W$ $cA_i \to c\#\vert_{\overline{P}_{i_{out}}}?$ $T_0 \to T_1\vert_{\overline{P}_{i_{out}}}$ $N_1 \to N_0\vert_{\overline{P}_{i_{out}}}$
step 6	$P_i, \overline{P}_i, Q, M_4$	$c, N_0, Y_i, T_1, \#?, W$ $T_1 \to \lambda\vert_{W_{out}}$
step 7	$P_i, \overline{P}_i, Q, M_4, W$ $M_4 \to M_0\vert_{W_{in}}$	$c, N_0, Y_i, \#?$

Table 4. The simulation of matrices with a rule in appearance checking mode; the first rule of the matrix is applied; the second one is applied if it can be applied, or, otherwise, it can be skipped.

	*Region*1	*Region*2
step 1	$P_i, \overline{P}_i, Q, M_0$ $M_0 \to M_{i,1}\|_{P_{i_{in}}}$	$c, N_0, A_i?, W$
step 2	$\overline{P}_i, Q, M_{i,1}$ $M_{i,1} \to M_{i,2}\|_{Q_{in}}$	$c, P_i, N_0, A_i?, W$
step 3	$\overline{P}_i, M_{i,2}$ $M_{i,2} \to M_{i,3}$	$c, P_i, Q, N_0, Y_i, T_0, A_i?, W$ $N_0 \to N_1\|_{Q_{out}}$
step 4	$\overline{P}_i, Q, M_{i,3}$ $M_{i,3} \to M_4\|_{\overline{P}_{i_{in}}}$	$c, P_i, N_1, Y_i, T_0, A_i?, W$ $N_1 \to \#\|_{P_{i_{out}}}$

Table 5. The simulation of matrices with a rule in appearance checking mode; the first rule cannot be applied.

At the beginning of each iteration, the object M_0 can be also rewritten into $M\#$ (the rule $M_0 \to M\#|_{D_{in}}$ is applied in region 1). In this case, the system non-deterministically check if the end of computation was reached, i.e., the symbol Z was produced. If the object Z is present in region 2, then it is deleted (the rule $Z \to \lambda|_{D_{out}}$ is applied in region 2) and, if no nonterminal of G is still present, then the computation stops. Otherwise, the trap symbol is produced and the computation will not stop.

□

5 Concluding Remarks and Open Problems

A new type of P systems has been introduced in this preliminary paper: Signals-Based P systems.

The idea of the model is simple: in a Signals-Based P system the symbol-objects cannot be moved across the membranes, but only a finite number of promoters can be moved through the regions of the system. The evolution rules are activated/deactivated by the presence/absence of these promoters, which are used like signals (from here the name of signal-promoter). We have discussed the biological motivations of this model and we have proved the universality when the systems use one catalyst and a bounded number of signal-promoters.

On the other hand, several open problems have been left open. Here are some of them. What we get when the system uses non cooperative rules? We suspect that the power of such systems is not more than the Parikh set of ET0L languages.

Moreover, in the universality proof we have used a bounded number of signal-promoters. What happens if we use a unique signal-promoter? Maybe one signal-promoter is enough to reach universality if we move also symbol-objects. What is the minimal number of different signal-promoters that we need to get universality?

Moreover in the universality proof we have also used a global clock. Is it possible to get universality only using signal-promoters? In other words is it possible to compute just using signals and to skip the help of the global clock? (That would constitute a first step in the construction of an asynchronous model of P systems.)

Of course, it would be also possible to consider Signals-Based P systems using signal-inhibitors instead of signal-promoters. Which results are possible get in this case?

Also, is it possible to compute just observing the traces of the signals (following the approach proposed in [5])?

A more general suggestion is to consider the idea of computing using signals in a more general framework: for example, what does this mean in the formal language area? or in the area of distributed computing (already linked with P systems area, [4]).

References

[1] I.I. Ardelean, Molecular Biology of Bacteria and its Relevance for P systems, *Membrane Computing*, (Gh. Păun, G. Rozenberg, A.Salomaa, C.Zandron eds.), LNCS 2597 (2003), Springer-Verlag, Berlin, 1–19.

[2] P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg, Membrane Systems with Promoters/Inhibitors, *Acta Informatica* 38, 10 (2002), 695–720.

[3] M. Cavaliere, Evolution–Communication P Systems, *Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597, Springer-Verlag, Berlin, 2003, 134–145.

[4] G. Ciobanu, R. Desai, A. Kumar, Membrane Systems and Distributed Computing, *Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597, Springer-Verlag, Berlin, 2003, 187–202.

[5] M. Ionescu, C. Martín-Vide, Gh. Păun, P systems with Symport/Antiport Rules: the Traces of Objects, *Grammars*, 5, 2(2002), 65–79.

[6] S.N. Krishna, A. Păun, Three Universality Results on P Systems, *Brainstorming Week on Membrane Computing, Tech. Rep. No. 26* (M. Cavaliere, C. Martn-Vide, Gh. Păun eds.), Rovira i Virgili University, Tarragona (2003), 198–206.

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

[8] J. Pelmont, *Catalyseurs du monde vivant*, Press Universitaires de Grenoble, Grenoble, 1995.

[9] L.B. Ray, E.M. Adler, N.R. Gough, L.D. Chong, Focus Issue: Computational Approaches in Signal Transduction, *Science's STKE*, eg3, 2004. The paper can be found at *www.stke.org*.

[10] G. Rozenberg, A. Salomaa eds., *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.

[11] L. Wang, Y.P. Sun, W.L. Chen, J.H. Li, C.C. Zhang, *FEMS Microbiol. Lett.*, 213 (2002), 155-165.

[12] D. White *The Physiology and Biochemistry of Prokaryotes*, Oxford University Press, 2nd edition, 2000.