Modeling the Dynamical Parallelism of Bio-Systems

Erzsébet Csuhaj-Varjú[1], Rudolf Freund[2], Dragoş Sburlan[3,4]

[1] Computer and Automation Institute
Hungarian Academy of Sciences
Kende utca 13–17, H-1111 Budapest, Hungary
csuhaj@sztaki.hu
[2] Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
rudi@emcc.at
[3] Department of Computer Science and Artificial Intelligence
University of Seville,
Av. Reina Mercedes, 41012, Seville, Spain
[4] Faculty of Mathematics and Informatics
Ovidius University of Constantza,
124 Mamaia Bd., Constantza, Romania
dsburlan@univ-ovidius.ro

Abstract. Among the many events that occur in the life of biological organisms are a multitude of specific chemical transformations, which provide the cell with usable energy and the molecules needed to form its structure and coordinate its activities. These biochemical reactions, as well as all other cellular processes, are governed by basic principles of chemistry and physics. A significant factor that determines whether or not reactions could take place is the entropy and it measures the randomness of the system. This measure depends on various factors like degrees of freedom (movement, vibration) for molecules, order in the solution, number of molecules, and so on. In an abstract framework, all these factors that describe the way molecules interact can be expressed by means of a computable multi-function that, depending on the current state of the system, it establishes the possible ways the system could evolve. Inspired by these facts, we introduce and study several bio-mimetic computational rewriting systems that use discrete components (i.e., finite alphabets, finite set(s) of rewriting rules, etc.) and which perform their computational steps in a non-deterministic manner and in a degree of rewriting parallelism that depends on the state of the system (both the non-determinism and the degree of the parallelism being specified by a given computable multi-function). Moreover, we are interested by systems that produce the same output, independently of the values taken by the considered functions.

1 Introduction

In nature, we often find biological systems (but not only) that are not necessarily homogeneous, consisting of many discrete, interacting entities that have a certain physical spatial distribution. This fact suggests that even if these entities interact in a parallel manner, they obey to some local conditions (concentration, for instance) and therefore the interaction parallelism cannot be considered as maximal or fixed, but as a variable that depends on the state of the system.

For example, at the cell level, bimolecular mechanisms are the result of many different chemical reactions that take place with a certain degree of mutual independence but such that they finally (and amazingly) exhibit an overall coordination. Traditionally these behaviors were modeled using the theory of partial derivative equations and nonlinear dynamical systems. However, this approach usually gave the general evolution and the dynamics of the system but not always giving the exact solution. From the discrete point of view, the interest was mainly in inferring the properties of the languages generated by such bio-inspired models. Although discrete models of complex phenomena may generate errors, the magnitude of the errors can be arbitrarily reduced by considering a better granularity of the phenomenon. This approach leads in general to a high computational effort, so a more efficient way of studying properties of bio-systems might be to consider discrete formal systems that have embedded in their formal description a certain degree of randomness, describing the way systems evolve.

One such property regards the measure of parallelism. From this point of view for instance, using biochemical reasoning one might predict that, given a particular state of a bio-system and the rules that make it evolves, an approximate next state is reached after a certain time. Basically, even if one does not know the exact number of times the rules are applied, one knows that after a particular time the reactions that had the potential to be applied, were actually accomplished in an approximate rate with respect to the state of the system.

Moreover, from the computer science point of view, in case we are trying to make use of bio-systems as computational devices we should be able to control their behavior no matter the rate of parallelism occurring within them. Therefore, we are interested in systems that one might call "parallel fault tolerant" meaning that they produce the same output no matter which is the "evolution" of the parallelism. This assumption might also have a biological counterpart, namely natural sub-systems are able to regulate themselves and replace, in case is needed, the functions of other sub-systems such that the overall system can perform the same task. From this point of view one can assume that a complex bio-system (like a cell or whatever organism) has the ability to reach a "desired" state, no matter how "local" decisions were made.

Here we will consider two bio-mimetic models, namely Lindenmayer systems, inspired by the development of multi-cellular organisms and P systems with promoters, motivated by enzyme activation/inactivation happening in the living cells.

We will extend the original definitions of these systems by considering computable multi-valued functions that control the derivation (in terms of specifying the rewriting parallelism and of nondeterminism).

2 Preliminaries

We assume the reader to be familiar with basic notions of formal languages, Lindenmayer systems, and P systems (for more details one can consult [4], [5], and [6]).

We start by briefly recalling some notions from multiset processing theory. Later on, we will introduce several new definitions needed for the purpose of this work.

A *multiset* over an arbitrary set X is a mapping $M : X \longrightarrow I\!N$. We denote by $M(x)$, $x \in X$, the multiplicity of x in the multiset M. If the set $X = \{x_1, \ldots, x_n\}$ is finite, then the multiset M can be explicitly given in the form $\{(x_1, M(x_1)), \ldots, (x_n, M(x_n))\}$. The support of a multiset M is the set $supp(M) = \{x \in X \mid M(x) \geq 1\}$. A multiset M is empty when its support is empty. Let $M_1, M_2 : X \longrightarrow I\!N$ be two multisets. We say that M_1 is included in M_2 (and we denote this by $M_1 \subseteq M_2$) if $M_1(x) \leq M_2(x)$, for all $x \in X$. The inclusion is strict if $M_1 \subseteq M_2$ and $M_1 \neq M_2$. The union (difference) of two multisets, $M_1 \cup M_2 : X \longrightarrow I\!N$ (respectively, $M_1 \setminus M_2 : X \longrightarrow I\!N$), is defined as $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$ (respectively, for $M_2 \subseteq M_1$, $(M_1 \setminus M_2)(x) = M_1(x) - M_2(x)$), for all $x \in X$. A multiset M of finite support, $\{(x_1, M(x_1)), \ldots, (x_n, M(x_n))\}$ can also be represented by the string: $w = x_1^{M(x_1)} x_2^{M(x_2)} \ldots x_n^{M(x_n)}$ and all the permutations of this string precisely identify the objects in the support of M and their multiplicities. Moreover, the Parikh image of w, $\Psi_X(w)$ is exactly the vector $(M(x_1), \ldots, M(x_n))$ of multiplicities. The cardinality of a multiset $w = x_1^{M(x_1)} x_2^{M(x_2)} \ldots x_n^{M(x_n)}$ is $card(w) = M(x_1) + M(x_2) + \ldots + M(x_n)$; the number of occurrences of x_i in w is denoted by $|w|_{x_i} = M(x_i)$, for $1 \leq i \leq n$.

Let $l \in I\!N$ and $w = x_1^{t_1} \ldots x_n^{t_n}$, $x_i \in X$, $t_i \in I\!N$, $1 \leq i \leq n$, a multiset over X. Then we can define the product $l * w = x_1^{l \cdot t_1} x_2^{l \cdot t_2} \ldots x_n^{l \cdot t_n}$.

Consider a finite set of symbols $V = \{a_1, a_2, \ldots, a_n\}$. An arbitrary multiset rewriting rule is a pair (u, v) with $u \in V^+$, $v \in V^*$ multisets over the set V; such a rule is typically written as $u \rightarrow v$. For a multiset rewriting rule $r : u \rightarrow v$, with $u, v \in V^*$ multisets over V, let $left(r) = u$ and $right(r) = v$.

In what follows we want to formally define the conditions required for a given multiset rewriting rule (or more multiset rewriting rules) to be applied (simultaneously applied, respectively) on a given multiset of symbols.

Let $w \in V^*$ be a multiset over V and let $R = \{r_1, r_2, \ldots, r_k\}$ be a set of multiset rewriting rules such that $r_i = u_i \rightarrow v_i$, with $u_i, v_i \in V^*$, $1 \leq i \leq k$. Denote by $R_w^{ap} \subseteq R$ the set of *applicable* multiset rewriting rules to w, that is, $R_w^{ap} = \{r \in R \mid left(r) \subseteq w\}$.

Denote by $R_w^{sap} = r_1^{t_1} r_2^{t_2} \ldots r_k^{t_k}$, $t_i \in I\!N$, $1 \leq i \leq k$, a multiset over R of *simultaneously applicable* multiset rewriting rules to w. R_w^{sap} is any multiset

such that:

$$\bigcup_{1\leq i\leq k} t_i * left(r_i) \subseteq w. \tag{1}$$

Denote by R_w^{SAP} the set of all multisets of simultaneously applicable rules to w, i.e., $R_w^{SAP} = \{R_w^{sap} \text{ satisfying (1)} \mid R_w^{sap} \in R^*\}$.

Next, we want to define which is the "impact" of a multiset of rules when they are applied on a given multiset of symbols (i.e., how many distinct symbols are rewritten). Based on this, we further define the set containing the multisets of rules that produce the biggest "impact" on a given multiset of symbols.

For $x = r_1^{i_1} r_2^{i_2} \dots r_k^{i_k} \in R_w^{SAP}$ let

$$D_x = supp(\bigcup_{i=1}^{k} left(r_i)).$$

The set D_x indicates all *distinct* symbols from w that are rewritten by an application of x.

Denote by

$$R_w^{MSAP} = \{x = r_1^{i_1} r_2^{i_2} \dots r_k^{i_k} \in R_w^{SAP} \mid card(D_x) = \max_{y\in R_w^{SAP}}(card(D_y))\}$$

the set of multisets of simultaneously applicable rules to w, called the *maximal component* of R_w^{SAP}.

Remark 1. The maximal component of R_w^{SAP} contains all multisets of simultaneously applicable rules such that the rewriting of distinct symbols is maximal (in the sense of the processed objects).

Let Y be a set of multisets over R; we denote

$$Pr(Y) = \{r_1 r_2 \dots r_k \mid r_1^{t_1} r_2^{t_2} \dots r_k^{t_k} \in Y\}.$$

We will use the set of sets $W_w = \{X \subseteq R_w^{MSAP} \mid Pr(X) = Pr(R_w^{MSAP})\}$.

Let

$$R_w^{MAX} = \{x \in R_w^{SAP} \mid \text{there exists } y \in R_w^{SAP} \text{ such that } x \subseteq y \text{ implies } x = y\}$$

be *the set of multisets of all maximal simultaneously applicable rules* to w.

Remark 2. The concept of maximal parallelism of rewriting (used, for instance, in the P systems framework) is expressed using the set $R_w^{MAX} \subseteq R_w^{SAP}$; for example, considering a P system Π in a given configuration and a region of Π containing a set of rules R that acts on the multiset w, an element in R_w^{MAX} gives a possible ensemble of rules that can be applied on w in a maximal parallel manner.

In addition, one can remark that $R_w^{MSAP} \supseteq R_w^{MAX}$. Observe that for non-cooperative multiset rewriting rules $Pr(R_w^{MAX}) = Pr(R_w^{MSAP})$.

The notions presented above regarding multisets and rules can be extended to strings and productions in a straightforward manner as follows*. However, as opposed to the multiset case, here we have to pay more attention to the implicit order between the symbols in a string.

Considering a set of productions $R = \{r_1, r_2, \ldots, r_k\}$ over an alphabet V, then the set of *applicable* productions to a string w is $RS_w^{ap} = \{r \in R \mid w = \alpha_1 left(r)\alpha_2, \alpha_1, \alpha_2 \in V^*\}$. Let $U = \{\overline{a} \mid a \in V\}$ and let $o : V^* \to U^*$ be a morphism that maps symbols from V into their corresponding overlined symbols. Let $h_{r_i} : (V \cup U)^* \to (V \cup U)^*$ such that $h_{r_i}(\alpha_1 left(r_i)\alpha_2) = \alpha_1 o(left(r_i))\alpha_2$, $\alpha_1, \alpha_2 \in (V \cup U)^*$. Then we can define a multiset of *simultaneously applicable* productions to a string w as follows (recall that we are not interested which are the sites in w where the productions from R are applied, but if they can be simultaneously applied). $RS_w^{sap} = r_1^{t_1} \ldots r_k^{t_k}$ such that there exists $h_{r_1}^{t_1}(h_{r_2}^{t_2}(\ldots h_{r_k}^{t_k}(w)\ldots))$, i.e., there exists "enough" sites in w where the productions could be applied. The set of all multisets of simultaneously applicable productions to the string w is denoted by RS_w^{SAP}.

The set of all *distinct* symbols from the string w rewritten by an application of the multiset of productions $x = r_1^{i_1} r_2^{i_2} \ldots r_k^{i_k} \in RS_w^{SAP}$ is

$$DS_x = \{a \in V \mid (\exists)\ r_i, 1 \leq i \leq k, \text{ such that } left(r_i) = \alpha_1 a \alpha_2, \alpha_1, \alpha_2 \in V^*\}.$$

The *maximal component* of RS_w^{SAP} is defined as for multisets, i.e.,

$$RS_w^{MSAP} = \{x = r_1^{i_1} r_2^{i_2} \ldots r_k^{i_k} \in RS_w^{SAP} \mid card(DS_x) = \max_{y \in RS_w^{SAP}}(card(DS_y))\}.$$

Let Y be a set of multisets over R; we denote

$$Pr(Y) = \{r_1 r_2 \ldots r_k \mid r_1^{t_1} r_2^{t_2} \ldots r_k^{t_k} \in Y\}.$$

We will use the set of sets $WS_w = \{X \subseteq RS_w^{MSAP} \mid Pr(X) = Pr(RS_w^{MSAP})\}$.

Example 1. Let $V = \{a, b, c\}$. Consider the multiset $w = aaabbbbccc$ and the set of multiset rewriting rules $R = \{r_1 : abc \to \alpha, r_2 : bcc \to \beta, r_3 : aac \to \gamma, r_4 : acccc \to \theta\}$. Then we have:

- $R_w^{SAP} = \{\lambda, r_1, r_1^2, r_1^3, r_2, r_3, r_1 r_2, r_1 r_3, r_2 r_3\}$;
- $R_w^{MAX} = \{r^3, r_1 r_2, r_1 r_3, r_2 r_3\}$;
- $D_{r_1 r_3} = D_{r_1} = \{a, b, c\}$; $D_{r_3} = D_{r_4} = \{a, c\}$;
- $R_w^{MSAP} = \{r_1, r_1^2, r_1^3, r_1 r_2, r_1 r_3, r_2 r_3\}$.

Example 2. Let $V = \{a, b, c, d\}$. Consider the string $w = aabc$ and the set of rewriting productions $R = \{r_1 : a \to \alpha_1, r_2 : a \to \alpha_2, r_3 : b \to \beta, r_4 : c \to \gamma, r_5 : d \to \theta\}$. Then we have:

$$RS_w^{SAP} = \{\lambda, r_1, r_2, r_1^2, r_2^2, r_3, r_4, r_1 r_2, r_1 r_3, r_2 r_3, r_1 r_4, r_2 r_4, r_3 r_4\}$$

* We will use similar notations as for the multiset case, the difference being a capital letter S on the right hand side of each defined operator.

$$\begin{aligned} &\cup \{r_1^2r_3, r_2^2r_3, r_1^2r_4, r_2^2r_4, r_1r_2r_3, r_1r_3r_4, r_2r_3r_4\} \\ &\cup \{r_1^2r_3r_4, r_2^2r_3r_4, r_1r_2r_3r_4\}; \\ RS_w^{MAX} &= \{r_1^2r_3r_4, r_2^2r_3r_4, r_1r_2r_3r_4\}; \\ RS_w^{MSAP} &= \{r_1r_3r_4, r_2r_3r_4, r_1^2r_3r_4, r_2^2r_3r_4, r_1r_2r_3r_4\}. \end{aligned}$$

$$\begin{aligned} WS_w = \{&\{r_1r_3r_4, r_2^2r_3r_4, r_1r_2r_3r_4\}, \{r_1r_3r_4, r_2r_3r_4, r_1r_2r_3r_4\}, \\ &\{r_1^2r_3r_4, r_2^2r_3r_4, r_1r_2r_3r_4\}, \ldots\}. \end{aligned}$$

3 On the Dynamical Parallelism of L Systems

In this section we extend the classical definition of Lindenmayer system in order to fit a more general perspective. Such systems model biological developments in which parts of organism change simultaneously but not in the total parallel manner as in the classical Lindenmayer theory, but with respect to the current state of the organism. Several results regarding the computational power of these systems are also presented.

Generally speaking, computing formal systems make usually use of rewriting rules to perform their computations. The semantics of such formal models provide the ways the rewriting rules are applied. Here, in order to capture the most general case, we will consider computable multi-valued functions that, depending on the current state of the system, control the applications of the rules. Intuitively, this assertion can be better understood if we express this mathematical formalism by means of a biological motivation. More specifically, for a given state of a bio-system (represented by a multiset/string w), one can predict that a certain rule, say $a \to \alpha$, is to be applied on w in a rate specified by a value in the interval $(x, y) \subseteq (0, 1)$. Therefore, in that computational step, the rule $a \to \alpha$ is applied a number of times i, such that $\left[x \cdot |w|_a\right] \leq i \leq \left[y \cdot |w|_a\right]$. However, when generalizing this concept, we might assume that there are more than one interval that control the applications of rules (generalizing even more, we have a set with no relation between its elements), and this brings us to the following formalism.

Definition 1. *An M-rate $0L$ system, denoted by $M0L$, is a quadruple $H = (V, R, \omega, f)$, where:*

- *$V = \{a_1, \ldots, a_m\}$ is a finite alphabet,*
- *R is a finite set of productions of the form $l : a \to \alpha$, with $a \in V$, $\alpha \in V^*$, and l is a label that precisely identifies[*] the production $a \to \alpha$. The set of productions R has to be complete, i.e., for each symbol $a \in V$ there must exist at least one production $l : a \to \alpha \in R$ with this letter a on its left side.*
- *f is a multi-valued computable function such that $f : V^* \to \mathscr{P}(R^*)$, $f(x) \in \mathscr{P}(RS_x^{SAP})$,*
- *$\omega \in V^*$ is the axiom.*

[*] We will drop off these labels in case their presence is unnecessary.

$M0L$ systems use M-rate parallel derivations, i.e., x directly derives y in a $M0L$ system $H = (V, R, \omega, f)$, with $x, y \in V^*$, written as $x \underset{f}{\Rightarrow} y$, if $x = x_1 x_2 \dots x_n$, $y = y_1 y_2 \dots y_n$, $x_i \in V$, $y_i \in V^*$, $1 \leq i \leq n$, and the following conditions hold:

- for every j, $1 \leq j \leq n$, either $y_j = x_j$ (the j-th symbol remains unchanged), or $r : x_j \to y_j \in P$ (some production of R is applied to the j-th symbol);
- the productions are applied according with $RS_x^{sap} = r_1^{t_1} r_2^{t_2} \dots r_k^{t_k} \in f(x)$ (i.e., the production r_i is applied t_i times, for $1 \leq i \leq k$).

This manner of derivation is called the *"weak mode"*. In the *"strong mode"* of derivation, we consider $f : V^* \to \mathscr{P}(R^*)$, $f(x) \in \{X \subseteq RS_x^{MSAP} \mid Pr(X) = Pr(RS_x^{MSAP})\}$.

The transitive and reflexive closure of $\overset{MOL}{\Longrightarrow}_H$ is denoted by $\overset{MOL}{\Longrightarrow}^*_H$. The generated language of the $M0L$ system H is

$$L(H) = \{u \in V^* \mid \omega \overset{MOL}{\Longrightarrow}^*_H u\}.$$

Remark 3. For both derivation modes, the *local degree of parallelism* for a given derivation step is defined by

$$\max_{y \in f(x)} (card(supp(y))).$$

For the weak mode of derivation, a degree of parallelism k, $1 \leq k \leq 2$, means that at most two distinct productions are applied simultaneously. The number of times each production is applied is given by the computable multi-valued function f.

Definition 2. *An M-rate $T0L$ system, denoted by $MTOL$, is a triplet $H = (V, T, \omega)$, where:*

- *V is a finite alphabet,*
- *$T = \{(T_1, f_1), \dots, (T_k, f_k)\}$ is a finite set of pairs, where each T_i, $1 \leq i \leq k$, is a complete set of context-free productions over V, and each f_i, $1 \leq i \leq k$, is a computable multi-valued function such that $f_i : V^* \to \mathscr{P}(T_i^*)$, $f_i(x) \in \mathscr{P}(T_i S_x^{SAP})$. This way of defining f_i, $1 \leq i \leq k$ stands for the weak mode of derivation. For the strong mode of derivation, multi-valued functions are defined as $f_i : V^* \to \mathscr{P}(T_i^*)$, $f_i(x) \in \{X \subseteq T_i S_x^{MSAP} \mid Pr(X) = Pr(T_i S_x^{MSAP})\}$, $1 \leq i \leq k$,*
- *$\omega \in V^*$ is the axiom.*

We say that x directly derives y in a $MT0L$ system $H = (V, T, \omega)$, with $x, y \in V^*$, written as $x \overset{MTOL}{\Longrightarrow}_H y$, if $x \overset{MOL}{\Longrightarrow}_{H_i} y$ for some i, $1 \leq i \leq k$, with the $M0L$ system $H_i = (V, T_i, \omega, f_i)$.

The transitive and reflexive closure of $\overset{MTOL}{\Longrightarrow}_H$ is denoted by $\overset{MTOL}{\Longrightarrow}^*_H$. The generated language of the $MT0L$ system H is

$$L(H) = \{u \in V^* \mid \omega \overset{MTOL}{\Longrightarrow}^*_H u\}.$$

Definition 3. *An M-rate ET0L system, denoted by $MET0L$, is a quadruple $H = (V, T, \omega, \Delta)$, where $\overline{H} = (V, T, \omega)$ is an $MT0L$ system, and $\Delta \subseteq V$, $\Delta \neq \emptyset$, is the terminal alphabet. In an $MET0L$ system $H = (V, T, \omega, \Delta)$, x directly derives y, with $x, y \in V^*$, written as $x \overset{METOL}{\Longrightarrow}_H y$, if $x \overset{MTOL}{\Longrightarrow}_{\overline{H}} y$.*

The transitive and reflexive closure of $\overset{METOL}{\Longrightarrow}_H$ is denoted by $\overset{METOL}{\Longrightarrow}^*_H$. The generated language of the $MET0L$ system H is $L(H) = \{w \in \Delta^* \mid \omega \overset{METOL}{\Longrightarrow}^*_H w\}$.

Definition 4. *An M0L (or MT0L, MET0L) system generating the same language independently of the functions associated with the set(s) of productions is called* parallel-free *M0L (or MT0L, MET0L, respectively) system.*

The families of languages generated by M0L (or MT0L, MET0L) systems working in the strong mode are denoted by $M0L^s$ (or $MT0L^s$, $MET0L^s$, respectively).

The families of languages generated by M0L (or MT0L, MET0L, respectively) systems working in the weak mode are denoted by $M0L^w$ (or $MT0L^w$, $MET0L^w$, respectively).

When we speak about above mentioned families of languages, we will denote the parallel-free property by adding the superscript pf.

We denote by $U = \{L \mid card(L) = 1\}$ the family of all singleton languages.

The following result characterize the computational power of extended, interactionless Lindenmayer systems when working in the weak mode and being parallel-free.

Theorem 1. $ME0L^{w,pf} = MET0L^{w,pf} = U \cup \{\emptyset\}$.

Proof. The assertion is trivial, because a system working in parallel-free mode means that whatever set/sets of multi-valued functions one can choose, the system produces the same output; in addition, because the system works in a weak mode then it means that certain productions might not be applied at all even if there are symbols (but not enough) that are within the scope of them. So, one can choose the multi-valued functions in such a manner that the productions cannot be applied at all. Therefore, such systems generate languages containing at most the systems axioms. □

Using a similar argument one can prove that:

Theorem 2. $MT0L^{w,pf} = M0L^{w,pf} = U$.

Example 3. Let $H = (V, P, \omega, F)$ be an $M0L$ system working in the *strong* mode such that:

$$\begin{aligned} V &= \{a, b, c\}, \\ P &= \{r_1 : a \to aa,\ r_2 : b \to bb,\ r_3 : c \to cc\}, \\ \omega &= abc, \\ f(w) &= \{r_1 r_2 r_3\}, \text{ for } w \in V^*. \end{aligned}$$

Obviously, H generates the language $L(H) = \{a^n b^n c^n \mid n \geq 1\}$.

Here we will prove that extended, interactionless Lindenmayer systems can generate any language over a given alphabet V.

Theorem 3. $ME0L^w = MET0L^w = \mathscr{P}(V^*)$.

Proof. Let us prove that using such systems we can generate any language L (computable or not). For the sake of simplicity but without loosing the generality we will generate languages over one letter alphabet. For a given $L \subseteq \{a\}^+$, we construct the system $H = (V, P, \omega, \Delta, F)$ where $V = \{a, A\}$, $\omega = A$, $\Delta = \{a\}$, the set P contains the following productions:

$$r_1 : A \to aA,$$
$$r_2 : A \to \lambda,$$
$$r_3 : a \to a,$$

and f is defined as follows.

For a given string $w = a^n A$, $n \geq 0$, we have:

- $f(w) \subseteq \{r_1 r_3^i \mid 0 \leq i \leq n\}$ iff $a^n \notin L$,
- $f(w) \subseteq PS_w^{SAP}$ and there exists $z \in \{r_2 r_3^i \mid 0 \leq i \leq n\}$ such that $z \in f(w)$ iff $a^n \in L$.

Observe that any time $a^n \notin L$, the number of symbols a grows (the production $r_1 : A \to aA$ is executed since there exists $z \in \{r_1 r_3^i \mid 0 \leq i \leq n\}$, such that $z \in f(a^n A)$).

In case $a^n \in L$, then f indicates that the production $r_2 : A \to \lambda$ might be executed (because of the way the function f was defined, also the production $r_1 : A \to aA$ might be executed). Therefore, non-deterministically, the system H generates a string $w \in L$.

In this way, the constructed system can generate any subset of V^*. Hence, we have that $ME0L^w = MET0L^w = \mathscr{P}(V^*)$. □

Example 4. The language $L = \{a, a^3\}$ is not $MOL^{s,pf}$ (or $MT0L^{s,pf}$) language. This is proved by contradiction as follows. If there exists a $MOL^{s,pf}$ system $H = (V, P, \omega)$ such that $L(H) = \{a, a^3\}$ then, since obviously $V = \{a\}$, we have two cases: (i) $\omega = a$ and $a \Rightarrow a^3$, hence $a^3 \Rightarrow a^k$, $k \neq 1, 3$, therefore a contradiction; (ii) $w = a^3$, hence $a \Rightarrow a$ and $a \Rightarrow \lambda$. Thus $a^3 \Rightarrow a^2$ and so $a^2 \in L(G)$, therefore a contradiction.

Obviously, the following results stand:

Proposition 1. $MT0L^{s,pf} \subset RE$.

Proposition 2. $M0L^{s,pf} \subset RE$.

Now, we will prove that there exists a class of ET0L systems that are independent of the multi-valued functions associated with the sets of rules and which are able to generate the whole class of ET0L languages. Before we start recall that for each $L \in ET0L$ there exists an ET0L system H with two tables such that $L = L(H)$.

Theorem 4. $MET0L^{s,pf} = ET0L$.

Proof. We prove this result by double inclusion.

(1) $MET0L^{s,pf} \supseteq ET0L$.
Consider an ET0L system $\widetilde{H} = (\widetilde{V}, \widetilde{T}, \widetilde{\omega}, \widetilde{\Delta})$ such that $\widetilde{T} = \{\widetilde{T_1}, \widetilde{T_2}\}$.

Let $h_1 : \widetilde{V}^* \to \overline{V}^*$ be a morphism such that $h_1(a) = \overline{a}$, $a \in \widetilde{V}$. Also, let $h_2 : \widetilde{V}^* \to \overline{\overline{V}}^*$ be a morphism such that $h_2(a) = \overline{\overline{a}}$, $a \in \widetilde{V}$.

We will simulate the computation of the system $\widetilde{H}$ with an MET0L system $H = (V, T, \omega, \Delta)$ defined as follows.
• $V = \widetilde{V} \cup \{h_1(A), h_2(A) \mid A \in \widetilde{V}\} \cup \{t_1, t_2, t_3, t_4\} \cup \{\#\}$;
• $T = \{(T_1, f_1), (T_2, f_2), (T_3, f_3), (T_4, f_4)\}$, where

$$
\begin{aligned}
T_1 = {} & \{A \to h_1(\alpha)t_1\,, \ \text{for all } A \to \alpha \in \widetilde{T_1}\} \\
& \cup \{h_1(A) \to h_1(A)\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_2(A) \to \#\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{\# \to \#\,, t_1 \to \lambda\,, t_2 \to \#\,, t_3 \to \#\,, t_4 \to \#\}, \\
T_2 = {} & \{A \to A\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_1(A) \to At_2\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_2(A) \to \#\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{\# \to \#\,, t_1 \to \#\,, t_2 \to \lambda\,, t_3 \to \#\,, t_4 \to \#\}, \\
T_3 = {} & \{A \to h_2(\alpha)t_3\,, \ \text{for all } A \to \alpha \in \widetilde{T_2}\} \\
& \cup \{h_1(A) \to \#\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_2(A) \to h_2(A)\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{\# \to \#\,, t_1 \to \#\,, t_2 \to \#\,, t_3 \to \lambda\,, t_4 \to \#\}, \\
T_4 = {} & \{A \to A\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_2(A) \to At_4\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{h_1(A) \to \#\,, \ \text{for all } A \in \widetilde{V}\} \\
& \cup \{\# \to \#\,, t_1 \to \#\,, t_2 \to \#\,, t_3 \to \#\,, t_4 \to \lambda\};
\end{aligned}
$$

and $f_i : V^* \to \mathscr{P}(R^*)$, $f_i(x) \in \{X \subseteq RS_x^{MSAP} \mid Pr(X) = Pr(RS_x^{MSAP})\}$, $1 \leq i \leq 4$, arbitrarily.
• $\omega = \widetilde{\omega}$;
• $\Delta = \widetilde{\Delta}$.

Here is how the construction is done. We want to simulate the applications of $\widetilde{H}$ tables; to this aim let us assume, without loosing the generality, that $\widetilde{T_1}$ is simulated first. So, in H, at the beginning, one table is chosen nondeterministically and is applied on the initial sentential form ω; in case table T_2 or T_4 is chosen, then the current sentential form is left unchanged (only the productions of type $A \to A$, $A \in V$ are applied). If table T_1 (or T_3) is chosen then productions of type $A \to h_1(\alpha)t_1$ will be executed. However, because the parallelism is not

necessarily maximal but depends on the function f_1 (or f_3, respectively), then not necessarily all symbols A from the current sentential form will be rewritten. However, since in table T_1 there exists only one production having the symbol A on the left-hand side (i.e., $A \to h_1(\alpha)t_1$) and the system is working in the strong mode, then the symbol A will be rewritten at least once (remember that $f_i(x) \in \{X \subseteq RS_x^{MSAP} \mid Pr(X) = Pr(RS_x^{MSAP})\}$). Consequently, a symbol t_1 is produced. Assume now that there are still symbols A not rewritten by T_1 despite the existence of a production handling symbol A. If this is the case, observe that if any other table (except T_1) is chosen for a next application, then the symbol # is generated (henceforth the system will not be able anymore to generate a string over Δ because the production $\# \to \#$ is present in every table of H and will be always executed). Therefore, the only table that can be applied and that does not produce the symbol # is again T_1. Finally, all symbols $A \in V$ will be rewritten by applications of table T_1. In addition, table T_1 is also responsible for deleting all symbols t_1. After completing these tasks, the current string will have only images by morphism h_1 of symbols from V (let us call them "overlined" symbols). At that moment, if we choose to apply any other table except T_2 the symbol # is again produced. In case table T_2 is applied, then with a similar mechanism as presented before, the system checks whether or not all "overlined" symbols are rewritten into "regular" ones. Again, during this checking procedure if we choose to apply another table other than T_2, symbol # is produced. The simulation of the application of table $\widetilde{T_2}$ follows a similar pattern. In this way, the computation take place step by step, simulating the computation of $\widetilde{(H)}$ if the "right" tables are chosen for application, and always producing the trap symbol if a "wrong" table is chosen for application.

Observe that the strong working mode feature is essential because if at a certain moment a wrong table is chosen for application, then we have to be sure that at least one symbol # is produced, hence a production has to be applied at least once if it can be applied.

Concluding, we have that the constructed system H generates the same language as the arbitrarily considered ET0L system $\widetilde{H}$. Consequently, we have that $MET0L^{s,pf} \supseteq ET0L$.

(2) $ET0L \supseteq MET0L^{s,pf}$.
In order to prove this inclusion, we will simulate the computation of an arbitrary MET0L^{s,pf} system $\overline{H} = (\overline{V}, \overline{T}, \overline{\omega}, \overline{\Delta})$ with an ETOL system $H = (V, T, \omega, \Delta)$ we construct. First, remark that because the system $\overline{H}$ is parallel-free, then, no matter how the multi-valued functions associated with the sets of productions are chosen, the result of the computation is the same. In particular, one can associate with all the sets of productions of the system the multi-valued functions such that, during the computation, the productions are applied in a total parallel manner (as for the L systems).

Therefore, H is defined as follows:
- $V = \overline{V}$;
- $T = \{T_1, T_2, \ldots, T_k\}$ providing that $\overline{T} = \{\overline{T_1}, \overline{T_2}, \ldots, \overline{T_k}\}$;
- $\omega = \overline{\omega}$;

• $T_i = \{A \to \alpha \mid A \to \alpha \in \overline{T_i}, 1 \leq i \leq k\}$;
• $\Delta = \overline{\Delta}$.

Observe that in an ET0L system, when a certain table is applied, if a production can be applied then it will be applied (of course, respecting the nondeterminism if exists). This corresponds to the strong mode of derivation for MET0L systems.

Consequently, we have that $ET0L \supseteq MET0L^{s,pf}$ and therefore we can conclude that $MET0L^{s,pf} = ET0L$. □

4 P Systems with Promoters/Inhibitors

A P system with promoters/inhibitors is formally defined as follows.

Definition 5. *A P system with promoters/inhibitors of degree $m \geq 1$, with symbol–objects is a construct*

$$\Pi = (V, \Sigma, C, P, I, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0),$$

where:

- *V is the alphabet of Π; its elements are called objects;*
- *Σ is the output alphabet, $\Sigma \subseteq V$;*
- *$C \subseteq V$ is the set of catalysts;*
- *P is the set of promoters, $P \subseteq V$;*
- *I is the set of inhibitors, $I \subseteq V$;*
- *μ is a membrane structure consisting of m membranes labeled $1, 2, \ldots, m$;*
- *w_i, $1 \leq i \leq m$, are strings that represent the multisets over V associated with the regions $1, 2, \ldots, m$ of μ;*
- *R_i, $1 \leq i \leq m$, are finite sets of* evolution rules *associated with the regions i of μ; an evolution rule can be non-cooperative, $a \to v$, promoted (inhibited) non-cooperative $a \to v|_p$ (or $a \to v|_{\neg i}$, respectively), catalytic $ca \to cv$, or promoted (inhibited) catalytic $ca \to cv|_p$ (or $ca \to cv|_{\neg i}$, respectively), where $a \in (V \setminus C)$, $c \in C$, $p \in P$, $i \in I$, and v is a string over V_{tar}, with $V_{tar} = (V \setminus C) \times TAR$, for $TAR = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$; the target is indicated as index of an object and if no target is specified that is intended to be* here*; also, the subscript j is omitted in in_j if there is only one choice to go for an object having this target indication.*
- *$i_0 \in \{1, \ldots, m\}$ specifies the output region of Π.*

The *membrane structure* is a hierarchical arrangement of membranes, embedded in a *skin membrane*, the one which separates the system from the environment. Each membrane defines a *region*, i.e., the space in-between the membrane and the membranes directly included in it (if any). Membranes are labeled in order to identify the regions they delimit.

The *configuration* of Π is an instantaneous description of it, including its membrane structure and the contents of all the membranes. The *initial configuration* is composed by the membrane structure μ and the objects initially present in the regions of the system, as described by $w_1, \ldots, w_m$.

The result of applying a rule $u \to v$ (or $u \to v|_p$ or $u \to v|_{\neg i}$), where $u, v \in V^*$ is determined by v: if the object a appears in v in the form a_{here} then it will remain in the same region, else if it is in the form a_{out} (or a_{in}) then it will exit to the upper region (or to one inner region, respectively).

A promoted rule $u \to v|_p$ can be applied only in the presence of promoter p. An inhibited rule $u \to v|_{\neg i}$ can be applied only in the absence of inhibitor i. Promoters and inhibitors can evolve according to some rules.

For two configurations $C_1 = (w'_1, w'_2, \ldots, w'_m)$ and $C_2 = (w''_1, w''_2, \ldots, w''_m)$ of Π, we define a *transition* from C_1 to C_2 if we can pass from C_1 to C_2 by applying the rules from $R_1, \ldots, R_m$, in the corresponding regions $1, \ldots, m$, in a maximally parallel manner and with competition on the objects* .

A *computation* of Π is a sequence of transitions between configurations, starting from the initial one. Π makes a *successful computation* iff it halts, i.e., there is no rule applicable to the objects present in the halting configuration.

The *result* of a successful computation is the number (or the vector of numbers) of objects from Σ present in the output region i_0 in the halting configuration. Collecting all the numbers (or vectors of numbers), for any possible halting configuration, we get the set $N(\Pi)$ (or $Ps(\Pi)$, respectively) – the set of numbers (vectors of numbers) generated by Π.

The family of sets of numbers (or vectors of numbers) generated by P systems with promoted ($\beta = pro$)/inhibited ($\beta = inh$) non-cooperative ($\alpha = ncoo$) and catalytic ($\alpha = cat_k$) object rewriting rules, using at most k catalysts, and having at most m membranes, is denoted by $NOP_m(\alpha, \beta)$ (or $PsOP_m(\alpha, \beta)$, respectively).

There are known the following results regarding the computational power of P systems with promoters/inhibitors.

- $PsOP_2(cat_1, pro) = PsOP_2(cat_1, inh) = PsRE$ (see [1],[3]).
- $PsOP_1(ncoo, inh) = PsET0L$ (see [8]).

However, given the definition of the model, in what follows we will prove that the model of P systems with non-cooperative promoted rules equals in computational power the model of P systems with non-cooperative inhibited rules.

Our first step is to show that the family of sets of vectors generated by P systems with promoted non-cooperative rules and an arbitrary membranes structure equals the family of sets of vectors generated by P systems with the same features but with only one membrane.

Lemma 1. *$PsOP_m(ncoo, pro) = PsOP_1(ncoo, pro)$, for $m \geq 2$.*

Proof. The inclusion $PsOP_m(ncoo, pro) \supseteq PsOP_1(ncoo, pro)$ is trivial. For the proof of the inclusion $PsOP_m(ncoo, pro) \subseteq PsOP_1(ncoo, pro)$, we construct a P system $\Pi_1 = (V, \Sigma, C, P, I, \mu, w, R, i_0)$ that simulates the computation of the P system $\overline{\Pi_m} = (\overline{V}, \overline{\Sigma}, \overline{C}, \overline{P}, \overline{I}, \overline{\mu}, \overline{w_1}, \ldots, \overline{w_m}, \overline{R_1}, \ldots, \overline{R_m}, \overline{i_0})$ in the following way.

* In Section 5 we consider another mode of applying the rules by employing some computable multi-valued functions that will govern the applications of the rules.

First, denote by $\mathcal{L} = \{1, 2, \ldots, m\}$ the set of labels of the regions of $\overline{\Pi_m}$. Then, we define:
• $V = \{a_i \mid a \in \overline{V}, i \in \mathcal{L}\}$;
• $\Sigma = \{a_i \mid a \in \overline{\Sigma}, i = \overline{i_0} \in \mathcal{L}\}$;
• $C = \overline{C} = I = \overline{I} = \emptyset$;
• $P \subseteq V$;
• $\mu = [\ ]_1$;
Let $h : \overline{V}^* \times \mathcal{L} \to V^*$ be a mapping such that
1) $h(a, i) = a_i, a \in \overline{V}, i \in \mathcal{L}$;
2) $h(\lambda, i) = \lambda$, for all $i \in \mathcal{L}$;
3) $h(x_1x_2, i) = h(x_1, i)h(x_2, i)$, $x_1, x_2 \in \overline{V}^*$, $i \in \mathcal{L}$.

• denote by $w = h(\overline{w_1})h(\overline{w_2}) \ldots h(\overline{w_m})$, where $\overline{w_i}$ is the multiset present in region $i \in \mathcal{L}$ of $\overline{\Pi_m}$ at the beginning of the computation;
• R is defined as follows. For each rule $a \to \alpha|_b \in \overline{R_i}$, $a, b \in \overline{V}$, α is a string over $\{c, c_{out}, c_{in} \mid c \in \overline{V}\}$, $i \in \mathcal{L}$, we add to R the rule $h(a, i) \to \alpha'|_{h(b,i)}$ where α' is the corresponding string over $\{h(c, i), h(c, j), h(c, k) \mid c \in \overline{V}, i, j, k \in \mathcal{L}\}$, j being the label of the outer region of i, and k being the label of the inner region of i;
• $i_0 = 1$.

In other words, for the P system with a single region that simulates a P system with m regions, we have encoded the region labels into objects (the subscript associated to an object indicates the region where the corresponding object belongs) and we have expressed the rules of regions by the corresponding encoded objects. In this way we ensured that, when simulating $\overline{\Pi_m}$ with Π_1, both the parallelism at the level of regions and at the level of whole system $\overline{\Pi_m}$ is respected. In addition, one can remark that whenever $\overline{\Pi_m}$ halts, Π_1 halts as well. Moreover, when Π_1 halts, we will have in the output region of Π_1 all the objects corresponding to the multisets present in all regions of $\overline{\Pi_m}$. However, in the output multiset w_{Π_1} of Π_1 we can distinguish the output multiset $w_{\overline{\Pi_m}}$ of $\overline{\Pi_m}$ because we know which are the objects corresponding to the output region of $\overline{\Pi_m}$ (they are the objects that have as index $\overline{i_0}$).

Therefore, we have that $PsOP_m(ncoo, pro) \subseteq PsOP_1(ncoo, pro)$. Consequently, the theorem holds. □

Now, one can prove that the class of sets of vectors of numbers generated by P systems with non-cooperative promoted rules includes at least the class of Parikh images of languages generated by ET0L systems.

Lemma 2. $PsOP_1(ncoo, pro) \supseteq PsET0L$.

Proof. We will simulate the computation performed by an arbitrary ET0L system $H = (V, T, \omega, \Delta)$ with $T = \{T_1, T_2\}$, using a P system Π_1 defined as follows.

$$\Pi_1 = (V_\pi, \Sigma, C, P, I, \mu, w, R, i_0), \text{ where:}$$

- $V_\pi = V \cup \{t, t_1, t_2, K, K_1\}$;
- $\Sigma = \Delta$;
- $C = I \ = \ \emptyset$;
- $P \subseteq V$;
- $w = \omega t$;
- $\mu = [\ ]_1$;
- $i_0 = 1$.

The set of rules R_π is defined as:

$$
\begin{aligned}
& t \to t_1, \\
& t \to t_2, \\
& A \to \alpha K|_{t_1}\ , \text{ for all rules } A \to \alpha \in T_1, \\
& A \to \alpha K|_{t_2}\ , \text{ for all rules } A \to \alpha \in T_2, \\
& K \to K_1, \\
& t_1 \to t|_A\ , \text{ for all } A \in V \setminus \Delta, \\
& t_2 \to t|_A\ , \text{ for all } A \in V \setminus \Delta, \\
& t_1 \to \lambda|_{K_1}, \\
& t_1 \to t|_{K_1}, \\
& t_2 \to \lambda|_{K_1}, \\
& t_2 \to t|_{K_1}, \\
& K_1 \to \lambda.
\end{aligned}
$$

At the beginning of simulation we have inside the region of the P system the input multiset, consisting of string ω (which corresponds to the axiom of the ET0L system H), and object t (which represents the starting trigger for the simulation of the nondeterministic table selection mechanism). Nondeterministically, object t is transformed into t_1 or t_2. Once object t_1 (or object t_2) is produced, the simulation of the corresponding ET0L table application starts. All rules $A \to \alpha K|_{t_1}$ (or $A \to \alpha K|_{t_2}$, respectively) corresponding to ET0L rules $A \to \alpha \in T_1$ (or $A \to \alpha \in T_2$, respectively) are applied in the maximally parallel manner. One can notice that if we applied at least once such a rule, we have produced at least one object K. In this moment we can distinguish two cases: 1) the current configuration is represented by a multiset that contains objects corresponding to ET0L nonterminals; 2) the current configuration is represented by a multiset that contains only objects corresponding to ET0L terminals.

In the first case one of the rules $t_1 \to t|_A$ or $t_2 \to t|_A$ will be executed, as well as the rule $K \to K_1$. Since an object t is produced, the simulation of applying a table in ET0L is iterated (recall that we do not have a terminal string, therefore we do not have to stop).

In the second case, the rules $t_1 \to t|_A$ or $t_2 \to t|_A$ cannot be executed because we assumed that the current configuration is represented by a multiset

that contains only objects corresponding to ET0L terminals. Therefore, the rule $K \to K_1$ is executed and afterward one of the rules $t_1 \to \lambda|_{K_1}$ (or $t_2 \to \lambda|_{K_1}$, respectively) and $t_1 \to t|_{K_1}$. Depending on which rule is chosen to be applied we have again two cases – we stop the simulation having in the output region a terminal string or we continue. In both cases, as a last step of the iteration, rule $K_1 \to \lambda$ is applied.

The above construction proves that $PsOP_1(ncoo, pro) \supseteq PsET0L$. □

For the converse inclusion we have to prove that any P system with non-cooperative promoted rules can be simulated by a P system with non-cooperative inhibited rules. Therefore, since $PsOP_1(ncoo, inh) = PsET0L$, the following result is true.

Lemma 3. $PsOP_1(ncoo, pro) \subseteq PsET0L$.

Proof. Let us consider a P system with non-cooperative promoted rules $\widetilde{\Pi} = (\widetilde{V}, \widetilde{\Sigma}, \widetilde{C}, \widetilde{P}, \widetilde{I}, \widetilde{\mu}, \widetilde{w}, \widetilde{R}, \widetilde{i_0})$ where $\widetilde{V} = \{A_i \mid 1 \leq i \leq k\}$, $\widetilde{\Sigma} \subseteq \widetilde{V}$, $\widetilde{P} \subseteq \widetilde{V}$, $\widetilde{C} = \widetilde{I} = \emptyset$, and $\widetilde{\mu} = [\ ]_1$.

Without any loss of generality, a non-cooperative rule $A \to \alpha$ is equivalent from computational point of view with the promoted non-cooperative rule $A \to \alpha|_A$. Hence, assume that $\widetilde{R} = \widetilde{R_1} \cup \widetilde{R_1} \cup \ldots \cup \widetilde{R_k}$, with $k = card(\widetilde{V})$, where

$$\begin{aligned}\widetilde{R_i} = \{&A_i \to \alpha_{(1,i)}|_{p_{(1,i)}},\\ &A_i \to \alpha_{(2,i)}|_{p_{(2,i)}},\\ &\ldots\ldots\ldots\ldots\\ &A_i \to \alpha_{(t_i,i)}|_{p_{(t_i,i)}}\}\end{aligned}$$

The set $\widetilde{R_i}$, $1 \leq i \leq card(\widetilde{V})$, contains all the rules from $\widetilde{R}$ having the symbol A_i in their left-hand side. Remark that all the rules are promoted by certain objects from $\overline{V}$; in particular all non-cooperative rules were written as the equivalent promoted ones using the above convention.

We construct a P system with non-cooperative inhibited rules Π that simulates the computation of $\widetilde{\Pi}$ as follows.

$$\Pi = (V, \Sigma, C, P, I, \mu, w, R, i_0), \text{ where:}$$

- $V = \widetilde{V} \cup \{r_{j,i} \mid A_i \to \alpha_{(j,i)} \in \overline{R}\} \cup \overline{V} \cup \overline{\overline{V}} \cup \dot{V} \cup \ddot{V} \cup \{X_0, X_1, X, Y_0, Y_1, Y\}$;
- $\Sigma = \widetilde{\Sigma}$;
- $C = P = \emptyset$;
- $I \subseteq V$;
- $\mu = [\ ]_1$;
- $i_0 = 1$;

and the set of rules R is defined as follows:

- for each $\widetilde{R_i} \subseteq \widetilde{R}$ defined before we add to R the rules*:

Step	Rule
1	$r_{(j,i)} \to \dot{r}_{(j,i)}\|_{\neg p_{(j,i)}}$, $1 \leq j \leq t_i$
1?	$A_i \to \overline{A_i} X_0$
2?	$X_0 \to X_1 X$
2?	$\overline{A_i} \to \overline{\overline{\alpha_{(j,i)}}} Y_0\|_{\neg \dot{r}_{(j,i)}}$
3?	$\dot{r}_{(j,i)} \to \ddot{r}_{(j,i)}\|_{\neg X_0}$
3?, 4?	$X \to \lambda$
3?	$X_1 \to X$
3?	$Y_0 \to Y_1 Y$
3?	$\overline{\overline{A_i}} \to \dot{A}_i$
3?	$\overline{A_i} \to \dot{A}_i\|_{\neg X_0}$
4?	$\ddot{r}_{(j,i)} \to \lambda\|_{\neg Y}$
4?,5?	$Y \to \lambda$
4?	$Y_1 \to Y$
4?	$\dot{A}_i \to \ddot{A}_i\|_{\neg Y}$
5?	$\ddot{r}_{(j,i)} \to r_{(j,i)}\|_{\neg X}$
5?	$\dot{A}_i \to A_i\|_{\neg X}$

Here is how the simulation is done. First of all recall that the classical definition of P systems assumes an universal clock that regulates the computation; we will use this feature to synchronize different branches of computation that run in parallel. The basic idea of the simulation is the following: we try to simulate the execution of the rules $A_i \to \alpha_{(j,i)}|_{p_{(j,i)}} \in \overline{R}$ by checking in different branches of computation (that require some renaming of objects) the simultaneous existence of A_i and of $p_{(j,i)}$. In case they are simultaneously available then we simulate the rewriting of A_i by $\alpha_{(j,i)}$, otherwise we reestablish the initial configuration (since we have used the renaming of objects). We stop the simulation when we detect that no rules can be further applied in the simulated P system.

In more details, let us consider that with each rule $A_i \to \alpha_{(j,i)}|_{p_{(j,i)}} \in \overline{R}$ is associated an object $r_{(j,i)} \in V$. Its purpose will be, by means of the rule $r_{(j,i)} \to \dot{r}_{(j,i)}|_{\neg p_{(j,i)}}$, to "check" whether or not the object $p_{(j,i)}$ is in the current multiset (if $p_{(j,i)}$ exists then we will have in the current multiset the object $\dot{r}_{(j,i)}$). Simultaneously, all existing objects A_i present will be rewritten by the rule $A_i \to \overline{A_i} X_0$. The object X_0 is the root of another branch of computation that is required to "collect" the not rewritten symbols $\overline{A_i}$ by the rule $\overline{A_i} \to \overline{\overline{\alpha_{(j,i)}}}$

* On the left side of each rule we have specified the step of a given iteration, in which the corresponding rule might be applied. The question marks indicate that the corresponding rules might not be applicable in that step.

(situation that occurs when the rules of type $A_i \to \alpha_{(j,i)}|_{p_{(j,i)}}$ were not executed because the objects $p_{(j,i)}$ were missing and there were no non-cooperative rules of type $A_i \to \alpha \in \overline{R}$). These objects are rewritten by the rule $\overline{A_i} \to \dot{A}_i$. Later, objects $\dot{A}_i$ will be rewritten into A_i if there exists at least one rule of $\overline{\Pi}$ that was simulated and the process is repeated, otherwise we stop the the computation by deleting all objects $\ddot{r}_{(j,i)}$ (by the rule $\ddot{r}_{(j,i)} \to \lambda|_{\neg Y}$) and transforming the objects $\dot{A}_i$ into $\ddot{A}_i$ (by the rule $\dot{A}_i \to \ddot{A}_i|_{\neg Y}$). It is worth to mention that the objects Y_0 (and their descendants, the objects Y_1, Y) are used as witness to the simulation of rules $A_i \to \alpha_{(j,i)}$. If at least one object Y appeared, it means that the simulation should continue since at least one rule was simulated. □

By Lemma 1, Lemma 2, and Lemma 3 we conclude that:

Theorem 5. *$PsOP_m(ncoo, pro) = PsOP_m(ncoo, inh) = PsET0L$, for $m \geq 1$.*

5 On Dynamical Parallelism of P Systems with Non-cooperative Promoted Rules

In this section we will extend the original definition of P systems with symbol rewriting rules but we will focus mainly on the definition of P systems using non-cooperative promoted rules. However, the notions presented can be easily extended to other models of P systems.

For instance considering an alphabet of objects V, a multiset of objects $w \in V^*$, and a set of multiset rewriting rules $R = \{r_i : u_i \to v_i|_{p_i} \mid u_i, v_i, p_i \in V^*, 1 \leq i \leq k\}$ we can define $R_w^{sap} = r_1^{t_1} r_2^{t_2} \dots r_k^{t_k}$, $t_i \in I\!N$, $1 \leq i \leq k$, a multiset over $R = \{r_i : u_i \to v_i|_{p_i} \mid 1 \leq i \leq k\}$ of simultaneously applicable multiset rewriting promoted rules to w. R_w^{sap} is any multiset such that:

$$\bigcup_{1 \leq i \leq k} t_i * left(r_i) \subseteq w \text{ and } p_i \subseteq w \text{ if } t_i > 0.$$

Based on this we can define as before the notions R_w^{SAP}, R_w^{MSAP}, D_x, R_w^{MAX}, W_w. Hence we can consider M-strong rate and M-weak rate modes of derivation in a similar fashion as we did for M0L systems.

Consider a P system $\Pi = (V, \Sigma, C, P, I, \mu, w_1, \dots, w_n, R_1, \dots, R_n, i_0)$ defined using the standard notation. Let $f_i : V^* \to \mathscr{P}(R^*)$, $f_i(x) \in \mathscr{P}(R_x^{SAP})$, $1 \leq i \leq n$, be computable multi-valued functions.

Consider now the system

$$\Pi(f_1, \dots, f_n) = (V, C, \Sigma, P, I, \mu, w_1, \dots, w_n, R_1, \dots, R_n, i_0, f_1, \dots, f_n)$$

and let us describe how the computations are performed.

Starting from the initial configuration given by the n-tuple $(w_1, \dots, w_n)$, where w_i, $1 \leq i \leq n$, are multisets over V, the system evolves according to the rules and objects present in the membranes, in a non-deterministic parallel manner. The selection of the rules as well as the number of applications of the

selected rules on the multiset w_i, $1 \leq i \leq n$, are given by a multi-valued function f_i, $1 \leq i \leq n$ (we have that $f_i(w_i) \in \mathscr{P}((R_i)_{w_i}^{SAP})$, in case of weak mode derivation, or $f_i(w_i) \in \{X \subseteq R_{w_i}^{MSAP} \mid Pr(X) = Pr(R_{w_i}^{MSAP})\}$ for the strong mode derivation). As usually, the system computes according with a universal clock.

For a given configuration $(x_1, x_2, \ldots, x_n)$ with x_i, $1 \leq i \leq n$, multisets over V, the rules are applied according with $(R_i)_{x_i}^{sap} = r_{(i,1)}^{t_1} r_{(i,2)}^{t_2} \ldots r_{(i,k)}^{t_k} \in f_i(x_i)$, $1 \leq i \leq n$ (i.e., given a multiset x_i, the rule $r_{(i,j)}$ is applied t_j times $1 \leq j \leq k$).

For two configurations $C_1 = w'$ and $C_2 = w''$ of Π, we can define the transition from C_1 to C_2 if we can pass from C_1 to C_2 by using the evolution rules from R_i, $1 \leq i \leq n$, applied according with the functions f_i, $1 \leq i \leq n$.

As usual, a *computation* of a P system Π is a sequence of transitions between configurations. The system will make a *successful computation* if and only if it halts. In case of generative P systems, the *result* of a successful computation is the number (or the vector of numbers) of objects from Σ present in the membrane i_0, in a halting configuration of Π. If the computation never halts, then we will have no output.

A P system Π is *parallel-free* if and only if for any set of functions f_i, $1 \leq i \leq n$, the system Π produces the same set of vectors of natural numbers.

For such P systems, when speaking about the generated families of sets of (vectors of) natural numbers we will specify the mode of derivation (weak or strong), as well as the parallel-free property by superscripts associated to the classical notation given in Section 4. For instance, $PsOP_m^{s,pf}(ncoo, pro)$ represents the family of sets of numbers generated by P systems using non-cooperative promoted rules, working in the strong mode and having the parallel-free property.

Let us examine the following example:

Example 5. Consider the P system $\Pi = (V, \Sigma, C, P, I, \mu, w, R, \vartheta, f)$, defined as follows:

$$
\begin{aligned}
V &= \{a, p\};\\
\Sigma &= \{a\};\\
C &= \emptyset;\\
P &= \{p\};\\
I &= \emptyset;\\
\mu &= [\ ]_1;\\
w &= a^2 p;\\
R &= \{r_1 : a \to aaa|_p, r_2 : p \to p, r_3 : p \to \lambda\};\\
\vartheta &= 1;
\end{aligned}
$$

$f(x) = \{r_1^{[0.5*|x|_a]+1} r_2^i r_3^j \mid i, j \in \{0, 1\}, i + j = 1\}$, $x \in V^*$.

Let us see in more details how this system performs the computation. First, observe that the execution of rule r_1 is controlled by the promoter p, while the

number of applications of r_1 on the current multiset is given by the function f. Assume that the rule $r_1 : a \to aaa|_p$ is executed k times in k derivation steps. Then we have:

$$\begin{aligned} |w|_a = |w_1|_a &= 2; \\ |w_2|_a &= 6; \\ |w_3|_a &= 14; \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ |w_k|_a &= ([|w_{k-1}|_a * 0.5] + 1) * card(right(r_1)) \\ &+ |w_{k-1}|_a - ([|w_{k-1}| * 0.5] + 1) \\ &= 2 * (1 + [|w_{k-1}| * 0.5]) + |w_{k-1}|_a \end{aligned}$$

where by w_i we denoted the multiset of objects present in the region of Π, at the step i of computation.

Observe that if $|w_1|_a \vdots 2 \Rightarrow |w_k|_a \vdots 2$, then this means that $[|w_i|_a * 0.5] = |w_i|_a * 0.5$, $1 \leq i \leq k$. Then, since $|w_1|_a = 2$, we can drop off the integer part function and we have the following recurrent formulas:

$$\begin{array}{rcl|l} |w_k|_a &=& 2 * |w_{k-1}|_a + 2 & *2^0 \\ |w_{k-1}|_a &=& 2 * |w_{k-2}|_a + 2 & *2^1 \\ \cdots\cdots & & \cdots\cdots\cdots\cdots & \\ |w_2|_a &=& 2 * |w_1|_a + 2 & *2^{k-2} \end{array}$$

In order to obtain the general term $|w_k|_a$ we will multiply each recurrent formula by a corresponding constant (as shown above) and we will sum the results. It follows that:

$|w_k|_a = 2^{k-1} * |w_1|_a + 2^1 + 2^2 + \cdots + 2^{k-1} = 2 + \cdots + 2^k = \frac{2*(2^k-1)}{2-1} = 2^{k+1} - 2$.

This means that Π generates the set $\{2^{k+1} - 2 \mid k \geq 2\}$.

In Section 3 we have shown that $MET0L^{s,pf} = ET0L$. Since we have proved in a constructive manner that $PsOP_m(ncoo, pro) = PsOP_m(ncoo, inh)$, for $m \geq 1$, and in [8] it is proved also constructively that $PsOP_m(ncoo, inh) = PsET0L$, then the following result stands.

P systems with $m \geq 1$ membranes, using non-cooperative promoted symbol objects rewriting rules, working in the strong mode, and having the parallel-free property, generates the set of all Parikh images of languages generated by ET0L systems. More formally, we have:

Theorem 6. *$PsOP_m^{s,pf}(ncoo, pro) = PsET0L$, for $m \geq 1$.*

6 Conclusions

In this paper we have considered a more general perspective of modeling biological systems. Here, chemical reactions happening in real organisms are modeled

by the rewriting of objects but not in a total parallel manner as in Lindenmayer systems or maximally parallel manner as in the P systems framework. Multi-valued functions govern the execution of the rules.

We define such type of parallelism with the help of multi-valued functions that for a given configuration establish the number of times rules are applied. Regarding this matter, we propose two ways of performing the derivation, the *weak* and the *strong* modes, respectively. The *weak mode* describes the case when no restriction is imposed on the way the rewriting is done; the system is parallel (not necessarily maximal) with competition on objects if it is the case. The *strong mode* of derivation assumes that rules are applied in a manner described by the multi-valued functions, but, in addition one can remark that each distinct symbol from the sentential form is rewritten at least once (of course, if there are rules that can handle it).

In this respect, we have relaxed the original P system definition by showing for the classes of systems we study that they contain proper subclasses able to generate/accept the same families of sets of vectors/numbers as the corresponding upper classes, regardless the rewriting parallelism in a given configuration. Systems having such property were called *parallel-free* P systems.

Several results regarding these topics are obtained. In addition, we studied "more classical" problems. In Section 4 we have solved an open problem in [4], regarding the computational power of P systems with non-cooperative promoted rules.

Acknowledgments.
The work of the third author was possible due to a doctoral FPU grant from Ministry of Education, Spain.

References

1. P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg, Membrane Systems with Promoters/Inhibitors. *Acta Informatica*, 38, 10 (2002), 695–720.
2. M. Cavaliere, D. Sburlan, Time Independent P Systems, *Lecture Notes in Computer Science*, 3365 (2005), 239–258.
3. M. Ionescu, D. Sburlan, On P Systems with Promoters/Inhibitors, *International Journal of Universal Computer Science*, 10, 5 (2004), 581–599.
4. Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.
5. G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.
6. G. Rozenberg, A. Salomaa, Eds., *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.
7. D. Sburlan, *Promoting and Inhibiting Contexts in Membrane Systems*, doctoral Thesis, University of Seville, Spain, 2006.
8. D. Sburlan, Further Results on P Systems with Promoters/Inhibitors, *International Journal of Foundations of Computer Science*, 17, 1 (2006), 205–222.