Further Results on P Systems with Promoters/Inhibitors

Dragoş Sburlan

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
and
Faculty of Mathematics and Informatics
Ovidius University of Constanta, Romania
E-mail: dsburlan@univ-ovidius.ro

Summary. The paper gives several results regarding P systems with non-cooperative rules and promoters/inhibitors at the level of rules. For the class of P systems using inhibitors, generating families of sets of vectors of numbers, the equivalence with the family of Parikh sets of ET0L languages is presented. In case of P systems with non-cooperative promoted rules even if an upper bound was not given, the inclusion of the family $PsET0L$ was proved. Moreover, a characterization of such systems by means of a particular form of random context grammars, therefore a sequential formal device, is proposed.

1 Introduction

P systems with promoters/inhibitors represent a possible abstract mathematical model of several biological activities happening in a cell. For example, the mechanism of activating/deactivating enzymes by what is called phosphorization (the covalent attachment of phosphoryl to the enzyme) as well as the way the enzymes control different metabolic reaction chains, might be modelled by such systems.

From a formal point of view we are interested to find properties of the languages generated by such bio-inspired devices. For the general case, when "weak" cooperative rules are considered, i.e., involving only one catalyst, the computational completeness of P systems with promoters/inhibitors was proved in [2] and [4]. Also, when the weight of promoters/inhibitors can be larger than one, universality results are obtained (see [1]).

Here we deal with systems that use non-cooperative rules and promoters/inhibitors of weight one. Because only promoters/inhibitors cannot restrict the parallelism of a P systems with non-cooperative rules, as catalysts do for instance

in the case of catalytic P systems, the common accepted conjecture was that such systems are not computational complete. In this paper, we give an answer to the problem regarding the computational capability of the family of P systems with inhibited non-cooperative rules; the case when promoted non-cooperative rules are used is left open.

In Section 2 we present basic notions and results regarding Lindenmayer systems and random context grammars. In addition, for the random context grammar, we define a new restrictive form – random context grammars with limited checking – and we show that such grammars can generate strictly more than the family of languages generated by ET0L systems. We conjecture that such systems are not computationally complete.

In Section 3 we prove that any P system using inhibitors can finish its computation having all membranes empty except the output one. Moreover, we show that such P systems generate exactly the same family of Parikh sets as ET0L systems. This result has a certain importance not only from formal languages point of view, but also from a biological perspective because, for instance, the membership and reachability problems for ET0L systems are decidable.

In Section 4 we give a lower bound for the family of sets of vectors generated by P systems with non-cooperative promoted rules, namely the family of Parikh images of ET0L languages. In what concerns the upper bound, we give a characterization by means of random context grammars with limited checking. In this way, an open problem arises about the maximal parallel computing by means of a sequential machine.

Along the paper, several open problems and research proposals are also presented.

2 Preliminaries

Here we recall several notions from the classical theory of formal languages, namely Lindenmayer systems and random context grammars. For further details regarding these notions we refer to [6], [3], and [7].

2.1 Lindenmayer Systems

A 0L system is a triple $H = (V, P, \omega)$, where V is a finite alphabet, P is a set of context-free rules over V, and $\omega \in V^*$ is the axiom. The set of rules P has to be complete, i.e., for each symbol $a \in V$ there must be at least one rule $a \to \alpha \in P$ with this letter a on the left-hand side.

0L systems use parallel derivations, i.e., x directly derives y in a 0L system $H = (V, P, \omega)$, with $x, y \in V^*$, written as $x \overset{OL}{\Longrightarrow}_H y$, if $x = x_1 x_2 \dots x_n$, $y = y_1 y_2 \dots y_n$, where $x_i \in V$, $y_i \in V^*$, and $x_i \to y_i \in P$, $1 \leq i \leq n$.

A T0L system is a triple $H = (V, T, \omega)$, where V is a finite alphabet, $T = \{T_1, \dots, T_k\}$ is a finite set of tables over V, where each table T_i, $1 \leq i \leq k$ is a

complete set of context-free rules over V, and $\omega \in V^*$ is the axiom. We say that x directly derives y in a T0L system $H = (V, T, \omega)$, with $x, y \in V^*$, written as $x \overset{TOL}{\Longrightarrow}_H y$, if $x \overset{OL}{\Longrightarrow}_{H_i} y$ for some i, $1 \leq i \leq k$, with the $0L$ system $H_i = (V, T_i, \omega)$.

An ET0L system is a quadruple $H = (V, T, \omega, \Delta)$, where $\overline{H} = (V, T, \omega)$ is a T0L system, and $\Delta \in V$, $\Delta \neq \emptyset$, is the terminal alphabet. In an ET0L system $H = (V, T, \omega, \Delta)$, x directly derives y, with $x, y \in V^*$, written as $x \overset{ETOL}{\Longrightarrow}_H y$, if $x \overset{TOL}{\Longrightarrow}_{\overline{H}} y$. The transitive and reflexive closure of $\overset{ETOL}{\Longrightarrow}_H$ is denoted by $\overset{ETOL}{\Longrightarrow}^*_H$. The language generated by the system H (denoted by $L(H)$) is $L(H) = \{w \in \Delta^* \mid \omega \overset{ETOL}{\Longrightarrow}^*_H w\}$. Thus in an ET0L system only words over a distinguished sub-alphabet are in the generated language. A language is said to be an ET0L language if there is an ET0L system generating it.

A 0L system is a TOL system that has only one table. Adding a number of tables yields an infinite hierarchy of subclasses of the class of T0L languages. In turn, every ET0L language can be generated by an ET0L system with only two tables.

We have that for any ET0L system $\overline{H}$ there exists an equivalent ET0L system $H = (V, T, \omega, \Delta)$ such that $V = V_N \cup \Delta$, $V_N \cap \Delta = \emptyset$, and each rule from $T_i \in T$, $1 \leq i \leq |T|$, is in one of the forms $A \to \alpha$, $A \in V_N$, $\alpha \in V^*$ or $a \to a$, $a \in \Delta$.

We will denote by $ET0L$ the family of languages generated by ET0L systems.

2.2 Random Context Grammars

A random context grammar is a quadruple $G = (N, T, P, S)$, where N, T, S are defined as in the context-free grammars and P is a finite set of random context rules, that is, triples of the form $(A \to \alpha, Q, R)$ where $A \to \alpha$ is a context-free rule with $A \in N$, $\alpha \in (N \cup T)^*$, and $Q, R \subseteq N$. For $x, y \in (N \cup T)^*$ we write $x \overset{rc}{\Longrightarrow} y$ iff $x = x_1 A x_2$, $y = x_1 \alpha x_2$ for some $x_1, x_2 \in (N \cup T)^*$, $(A \to \alpha, Q, R)$ is a triple in P, all symbols of Q appear and no symbol of R appears in $x_1 A x_2$. We will refer Q as the permitting context and R as the forbidding context of the rule $A \to \alpha$. If the forbidding context is empty for every rule, then we speak about a random context grammar without appearance checking. The language generated by a random-context grammar $G = (N, T, P, S)$ is $L(G) = \{v \in T^* \mid S \overset{rc}{\Longrightarrow}^* v\}$, where by $\overset{rc}{\Longrightarrow}^*$ we have denoted the transitive and reflexive closure of $\overset{rc}{\Longrightarrow}$.

We denote by RC^{λ}_{ac} the family of languages generated by random-context grammars with appearance checking and λ rules. The following result stands: $RC^{\lambda}_{ac} = RE$.

For any random context grammar with appearance checking and λ rules there exists an equivalent random context grammar with appearance checking and λ rules $G' = (N', T', P', S')$ such that for the rules $(A \to \alpha, Q, R) \in P'$ we have $|\alpha| \leq 2$, $|Q| \leq 1$, $|R| \leq 1$.

Definition 1. *A random context grammar with limited checking and λ rules is a random context grammar $G = (N, T, P, S)$ with rules of the form $(A \to \alpha, Q, R)$,*

where $A \to \alpha$ is a context-free rule with $A \in N$, $\alpha \in (N \cup T)^*$, and $Q, R \subseteq N$, $|Q| + |R| \leq 1$.

We denote by RC_{lc}^{λ} the family of languages generated by random context grammars with limited checking and λ rules.

A natural question arise:

Open Problem: what is the generative power of random context grammars with limited checking?

Conjecture. Random context grammars with limited checking are not universal, i.e., $RC_{lc}^{\lambda} \subset RE$. One can remark that this is a "winning" problem irrespective which is its answer, since if such grammars prove to be universal, then even if Theorem 5 become trivial, we will have a new normal form for random context and matrix grammars with appearing checking, but if they are not universal, then, as we will see, this implies that P systems with promoters are not universal. We also suggest an approach that might be useful for proving the non-universality of catalytic P systems with only one catalyst.

Anyway, random context grammars with limited checking are strictly more powerful than ET0L systems.

Theorem 1. *$RC_{lc}^{\lambda} \supset ET0L$.*

Proof. Consider an *ET0L* system $H = (V, T, \omega, \Delta)$ such that $V = V_N \cup \Delta$, $V_N \cap \Delta = \emptyset$, $T = \{T_1, T_2, \ldots, T_k\}$ and all rules from T_i, $1 \leq i \leq k$ are in one of the forms $A \to \alpha$, $A \in V_N$ $\alpha \in V^*$, or $a \to a$, $a \in \Delta$. We will simulate the computation of the system H with a random context grammars with limited checking $G = (N_G, T_G, P_G, S_G)$.

Since in H the rules $a \to a$, $a \in \Delta$, do not affect the derivation process, we can remove them from tables. Therefore, let $T_i = \{A_{(i,j)} \to \alpha_{(i,j)} \mid 1 \leq j \leq r_i\}$, $1 \leq i \leq k$, be the sets of rules representing the ET0L tables.

We denote:

$$N_G = (V \setminus \Delta) \cup \{t_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq 2 * r_i + 1\} \cup \{\overline{A} \mid A \in V \setminus \Delta\}$$
$$T_G = \Delta.$$

We define the set of rules P_G as follows.

$$(S \to t_{(i,1)}\omega,\ \emptyset, \emptyset),\ \text{with } 1 \leq i \leq k.$$

The starting symbol S is non-deterministically changed into one table selector ($t_{(i,1)}$, $1 \leq i \leq k$) and the string ω representing the axiom of the ET0L to be simulated.

Next we have the following rules that correspond to the rules of the tables:

$$(A_{(i,j)} \to \overline{\alpha_{(i,j)}},\ \{t_{(i,j)}\}, \emptyset),\ \text{with } 1 \leq i \leq k, 1 \leq j \leq r_i.$$

Basically, each rule produces "colored" (overlined) symbols in the presence of symbol $t_{(i,j)}$, $1 \leq i \leq k$, $1 \leq j \leq r_i$, representing the selected table.

The next task needed for a correct simulation is to check whether or not all symbols in the current sentential form were actually transformed into overlined ones. In order to accomplish this task we have the rules:

$$(t_{(i,j)} \rightarrow t_{(i,j+1)},\ \emptyset, \{A_{i,j}\}),\ \text{with } 1 \leq i \leq k, 1 \leq j \leq r_i.$$

In fact, there are needed even a smaller number of rules of the above type. This is due to the fact that our system might contain more rules with the same symbol on the left-hand side. However, the reason for adopting such numbers was just for having a simpler notation.

In case there are still remaining not overlined symbols, a symbol $t_{(i,r_i+1)}$, $1 \leq i \leq k$, $1 \leq j \leq r_i$, is not produced.

We also have the rules:

$$(\overline{A_{(i,j)}} \rightarrow A_{(i,j)},\ \{t_{(i,r_i+1)}\}, \emptyset),\ 1 \leq i \leq k, 1 \leq j \leq r_i.$$

Their role is to change back the overlined symbols into regular ones. Now, in order to simulate the selection of a new table and to iterate the process, we have to be sure that all overlined symbols have been changed back into regular ones.

The procedure is accomplished, in a similar way, by the rules:

$$(t_{(i,j)} \rightarrow t_{(i,j+1)},\ \emptyset, \{\overline{A_{(i,j-r_i)}}\}),\ 1 \leq i \leq k, r_i + 1 \leq j \leq 2 * r_i.$$

Finally, if everything worked as we expected (recall that the simulation is non-deterministic and if the derivation went in a "wrong" way, then it is blocked), then we have produced a symbol $t_{(i,r_i+1)}$, $1 \leq i \leq k$. Next, we can use this symbol to repeat the table selection mechanism or to stop non-deterministically the generative core. These actions are achieved by making use of the rules:

$$(t_{(i,2*r_i+1)} \rightarrow t_{(h,1)},\ \emptyset, \emptyset),\ \text{with } 1 \leq i, h \leq k,$$
$$(t_{(1,2*r_i+1)} \rightarrow \lambda,\ \emptyset, \emptyset).$$

Because we consider in the language only terminal strings, we have $L(H) = L(G)$, hence the inclusion $RC_{lc}^{\lambda} \supseteq ET0L$ is proved.

In order to prove that the inclusion is strict we have to construct a language that cannot be generated by an ET0L system. To this aim, we construct $G = (V, T, P, S)$ that generates the language $\{(a^n b)^m \mid m \geq n \geq 1\}$ defined as follows:

- $V = \{S, A, B, C, D, X, Y, N, N', P, P', P'', R, Q, Z, a, b\}$,
- $T = \{a, b\}$,
- P contains the following rules:

$(S \to AX,\ \emptyset, \emptyset)$
$(A \to BB,\ \{X\}, \emptyset)$
$(X \to YN,\ \emptyset, \{A\})$
$(B \to A,\ \{Y\}, \emptyset)$
$(Y \to X,\ \emptyset, \{B\})$
$(X \to P,\ \emptyset, \emptyset)$
$(A \to aC,\ \{P\}, \emptyset)$
$(P \to P',\ \{N'\}, \emptyset)$
$(N \to N',\ \emptyset, \emptyset)$
$(N' \to \lambda,\ \emptyset, \emptyset)$
$(P' \to P'',\ \emptyset, \{A\})$
$(P'' \to Q,\ \emptyset, \{N'\})$
$(C \to bD,\ \{Q\}, \emptyset)$
$(Q \to R,\ \emptyset, \{C\})$
$(D \to C,\ \{R\}, \emptyset)$
$(R \to Q,\ \emptyset, \{D\})$
$(Q \to Z,\ \emptyset, \{C\})$
$(D \to \lambda,\ \{Z\}, \emptyset)$
$(Z \to \lambda,\ \emptyset, \emptyset)$

The details concerning the way this grammar works are left to the reader. Consequently, we have $RC_{lc}^{\lambda} \supset ET0L$. □

3 Non-cooperative P Systems with Inhibitors

In this section we consider P systems with symbol objects, non-cooperative rules, inhibitors at the level of rules, and generating sets of vectors of numbers. The family of sets of vectors generated by systems with at most $m \geq 1$ membranes is denoted by $PsP_m(ncoo, inhR)$. In order to prove the main result of the section, that is $PsP_m(ncoo, inhR) = PsET0L$, we will accomplish the following plan:

- show the equivalence between P systems with non-cooperative inhibited rules using m membranes, and P systems with non-cooperative inhibited rules and only one membrane;
- show that any P system with non-cooperative inhibited rules is equivalent with a P system with non-cooperative inhibited rules having the alphabet made out of two disjoint sets, the set of terminals and of nonterminals; in addition, all rules have a non-terminal on their left-hand side; moreover, the set is complete, i.e., for each nonterminal there exists at least one rule having it on the left-hand side;
- for a given set of inhibited rules, define saturated classes of rules, i.e., find the sets containing rules that does not mutually forbids each other;
- show that $PsP_m(ncoo, inhR) = PsET0L$ by double inclusion.

Here is how we proceed:

Theorem 2. *$PsP_m(ncoo, inhR) = PsP_1(ncoo, inhR)$, for $m \geq 2$.*

Proof. The inclusion $PsP_m(ncoo, inhR) \supseteq PsP_1(ncoo, inhR)$ is trivial. For the proof of the inclusion $PsP_m(ncoo, inhR) \subseteq PsP_1(ncoo, inhR)$, we construct a P system $\Pi_1 = (V, C, \mu, w, R, \vartheta)$ that simulates the computation of P system $\overline{\Pi_m} = (\overline{V}, \overline{C}, \overline{\mu}, \overline{w_1}, \ldots, \overline{w_m}, \overline{R_1}, \ldots, \overline{R_m}, \overline{\vartheta})$ in the following way.

First, denote by $\mathcal{L} = \{1, 2, \ldots, m\}$ the set of labels of the regions in $\overline{\Pi_m}$. Then, we define:

• $V = \{a_i \mid a \in \overline{V}, i \in \mathcal{L}\}$.

• $C = \overline{C} = \emptyset$;
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) $h(\lambda, j) = \lambda$, for all $j \in \mathcal{L}$;
3) $h(x_1x_2, j) = h(x_1, j)h(x_2, j)$, $x_1, x_2 \in \overline{V}^*$, $j \in \mathcal{L}$.

• denote by $w = h(\overline{w_1})h(\overline{w_2}) \dots h(\overline{w_m})$, 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 $a \to \alpha|_{\neg b} \in \overline{R_i}$, $a, b \in \overline{V}$, 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 $h(a, i) \to \alpha'|_{\neg h(b,i)}$ 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.
• $\vartheta = 1$;

In other words, for the P system with a single region that simulates a P system with m regions, we have encoded the regions 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 $\overline{\vartheta}$). Therefore, we have to delete the unnecessary objects that remain in the output region of Π_1 in a halting configuration since we want to show that Π_1 and $\overline{\Pi_m}$ generate exactly the same set of vectors of numbers. We will modify the rules presented above in the following manner.

We add to the vocabulary V a new symbol D (the object D stands for the "deletion command") and we replace each rule $a_i \to \alpha'|_{\neg b_i} \in R$ by

$$a_i \to \alpha' D|_{\neg b_i} \in R,$$

and each rule $a_i \to \alpha' \in R$ by

$$a_i \to \alpha' D \in R,$$

respectively.

In addition, we add the following rules

$$D \to \lambda,$$
$$a_i \to \lambda|_{\neg D},$$

for all $a_i \in V$, $i \neq \overline{\vartheta}$
One can remark that in this way we produce at each computational step at least

one object D and also, in the same time, we delete the already existing object(s) D. If there exist rules that can be executed (i.e., there will be objects D) rules of type $a_i \to \lambda|_{\neg D}$ cannot be applied. When the computation halts, objects D are not produced anymore, and so, the deletion rules can start and erase the remaining unnecessary objects. Consequently we have shown that both systems generate the same family of vectors of natural numbers, hence we have $PsP_m(ncoo, inhR) \subseteq PsP_1(ncoo, inhR)$. □

As a consequence of the proof above we have the following corollary:

Corollary 1. *For any P system Π with inhibited non-cooperative rules there exists an equivalent P system Π' with inhibited non-cooperative rules such that, for any halting configuration of Π', all regions of Π', excepting the output one, are empty.*

Remark 1. Later we will show that P systems with inhibited non-cooperative rules generate exactly $PsET0L$. Therefore, one can prove the above result also by using the fact that the family $ET0L$ is closed under arbitrary morphisms (so we can delete the unnecessary objects in a halting configuration).

Now, let us define the *non-excluding inhibiting relation* and *saturated sets* with respect to this relation.

First, for a given alphabet V, let $R = \{r_1, r_2, \ldots, r_k\}$ be a set of productions of the form $r_i : (A_i \to \alpha_i|_{\neg B_i})$, $A_i, B_i \in V$, $A_i \neq B_i$, $\alpha_i \in V^*$, $1 \leq i \leq k$.

For a rule $r : (A \to \alpha|_{\neg B}) \in R$ we define $left(r) = A$ and $inh(r) = B$.

Two rules $r_i, r_j \in R$ are said to be in the *non-excluding inhibiting relation*, and we denote this by $r_i \equiv_{nei} r_j$, iff $left(r_i) \neq inh(r_j)$ and $left(r_j) \neq inh(r_i)$.

Remark 2. We have the following properties:

- $\equiv_{nei}$ is reflexive (obvious),
- $\equiv_{nei}$ is symmetric (obvious),
- $\equiv_{nei}$ is not transitive.

For example, take $R = \{r_1 : (A \to \alpha|_{\neg B}), r_2 : (D \to \beta|_{\neg C}), r_3 : (B \to \gamma|_{\neg A})\}$. Observe that $r_1 \equiv_{nei} r_2$, $r_2 \equiv_{nei} r_3$, but $r_1 \not\equiv_{nei} r_3$.

Definition 2. *A subset $W \subseteq R$ is said to be* saturated *(or* complete*) with respect to non-excluding inhibiting relation $\equiv_{nei}$ iff $(\forall)\ r_i, r_j \in W$, $r_i \equiv_{nei} r_j$, and $(\forall)\ r_i \in R \setminus W$, $(\exists)\ r_j \in W$ such that $r_i \not\equiv_{nei} r_j$.*

Remark 3. The set R may contain more saturated subsets, say $W_1, W_2, \ldots, W_k$, with respect to the non-excluding inhibiting relation $\equiv_{nei}$. We have $W_1 \cup W_2 \cup \ldots \cup W_k = R$ and $W_i \cap W_j$, $1 \leq i, j \leq k$, not necessarily empty.

Example 1. Consider the following set of rules

$$R = \{r_1 : (A \to \alpha_1|_{\neg B}), r_2 : (C \to \alpha_2|_{\neg D}), r_3 : (B \to \alpha_3|_{\neg D}),$$
$$r_4 : (A \to \alpha_3|_{\neg C}), r_5 : (D \to \alpha_4|_{\neg C})\}.$$

Then we have:

W_1:	W_2:	W_3:	W_4:
$r_1 : (A \to \alpha_1 \vert_{\neg B})$	$r_3 : (B \to \alpha_3 \vert_{\neg D})$	$r_1 : (A \to \alpha_1 \vert_{\neg B})$	$r_2 : (C \to \alpha_2 \vert_{\neg D})$
$r_2 : (C \to \alpha_2 \vert_{\neg D})$	$r_4 : (A \to \alpha_3 \vert_{\neg C})$	$r_4 : (A \to \alpha_3 \vert_{\neg C})$	$r_3 : (B \to \alpha_3 \vert_{\neg D})$
		$r_5 : (D \to \alpha_4 \vert_{\neg C})$	

Lemma 1. *For any P system* $\overline{\Pi_1} = (\overline{V}, \overline{C}, \overline{\mu}, \overline{w}, \overline{R}, \overline{\vartheta})$ *with non-cooperative rules and inhibitors there exists an equivalent P system* $\Pi_1 = (V, C, \mu, w, R, \vartheta)$ *with non-cooperative rules and inhibitors such that:*

$$
\begin{aligned}
&V = V_N \cup V_T,\ V_N \cap V_T = \emptyset,\\
&C = \overline{C} = \emptyset,\\
&\mu = \overline{\mu} = [\,]_1,\\
&\quad w \subseteq V_N,\\
&R = \{r_1, \ldots, r_k\},\ \textit{is a set of rules of the form } A \to \alpha \vert_{\neg B}\\
&\quad \textit{or } A \to \alpha,\ \textit{where } A, B \in V_N, \alpha \in V^*,\\
&\quad \vartheta = 1.
\end{aligned}
$$

Proof. (Sketch) We define the set $V_N = \{A \mid a \in \overline{V}\} \cup \{D\}$ and $V_T = \overline{V}$. The set of rules is defined as:

$$
\begin{aligned}
R = &\{A \to \Phi D \mid a \to \phi \in R, A, D \in V_N, \Phi \in V_N^*\}\\
&\cup \{A \to \Phi D \vert_{\neg B} \mid a \to \phi \vert_{\neg b} \in R, A, B, D \in V_N, \Phi \in V_N^*\}\\
&\cup \{D \to \lambda\}\\
&\cup \{A \to a \vert_{\neg D} \mid A, D \in V_N, a \in V_T\}
\end{aligned}
$$

where by Φ we understand the image of ϕ through a morphism that maps all regular letters from $\overline{V}$ to corresponding capital letters. Basically we detect when the system halts and only at that time we change the nonterminals into terminals. □

Remark 4. A similar result stands also for P systems with non-cooperative rules and promoters. Of course, both results can be extended to P systems having m membranes and not with only one membrane as we have considered.

Lemma 2. *For any P system* $\overline{\Pi_1} = (\overline{V}, \overline{C}, \overline{\mu}, \overline{w}, \overline{R}, \overline{\vartheta})$ *with non-cooperative rules and inhibitors given in the form specified by Lemma 1 there exists an equivalent P system* $\Pi_1 = (V, C, \mu, w, R, \vartheta)$ *with non-cooperative rules and inhibitors such that the set of rules R is complete (i.e., for each* $A \in V$ *there exists at least one rule of type* $A \to \alpha \in R$ *or* $A \to \alpha \vert_{\neg B} \in R$*).*

Proof. (Sketch) Assume that in $\overline{R}$ there is no rule with object B on its left-hand side. (i.e., there are no rules of type $B \to \beta$ or $B \to \beta \vert_{\neg C}$). In addition, assume that object B is produced by rules of type $A \to \alpha B$ or $A \to \alpha B \vert_{\neg C}$. Also, suppose there is at least one rule of type $A \to \alpha \vert_{\neg B}$.

We construct the P system Π_1, having the set of rules R complete, that simulates the moves of $\overline{\Pi_1}$ in the following manner. First, let

- $V = \overline{V} \cup \{D, b\}$.

The set of rules R is defined as follows:

- for any rule of type $A \to \alpha \in \overline{R}$ we add to R the rules:

$$A \to \alpha D,$$
$$D \to \lambda;$$

- for any rule of type $A \to \alpha|_{\neg C} \in \overline{R}$ we add to R the rules:

$$A \to \alpha D|_{\neg C},$$
$$D \to \lambda;$$

- for any rule of type $A \to \alpha B \in \overline{R}$ or $A \to \alpha B|_{\neg C} \in \overline{R}$ (therefore rules that produce at least one object B) we add to R the rules:

$$B \to B,$$
$$B \to b|_{\neg D}.$$

The explanation of the simulation is rather easy: at each computational step Π_1 produces the same multiset of objects as $\overline{\Pi_1}$ and, in addition, several copies of object D (according to the number of rules applied in $\overline{\Pi_1}$). The object D will be used in Π_1 to trigger a signal that the corresponding computation in $\overline{\Pi_1}$ halts (the absence of object D from the current multiset means that no rules from R, that correspond to rules in $\overline{R}$, can be applied anymore).

Now, suppose the object B is produced by a rule at a certain moment. Then, since in any non-halting configuration there exists object(s) D, the only rule handling object B that can be executed is $B \to B$. Finally, if the computation stops in $\overline{\Pi_1}$, then it means that in Π_1 the rules that can be further executed (in a non-deterministic manner) are $B \to B$ and $B \to b|_{\neg D}$. In case rule $B \to b|_{\neg D}$ is executed for all existing objects B, then the computation stops, Π_1 producing the same multiset of objects as $\overline{\Pi_1}$. □

Remark 5. A similar result stands also for P systems with non-cooperative rules and promoters. Of course, both results can be extended to P systems having any number $m \geq 1$ membranes.

Now we can state the following result:

Theorem 3. $PsP_m(ncoo, inhR) = PsET0L$.

Proof. We prove this result by double inclusion.

Case 1. $PsP_m(ncoo, inhR) \supseteq PsET0L$.

For an ET0L system $H = (V, T = \{T_1, T_2\}, \omega, \Delta)$ we can construct a P system $\Pi = (V_\pi, C, \mu, w, R, \vartheta)$ that simulates any of its derivations in the following way.

$$
\begin{aligned}
V_\pi &= V \cup \{t_1, t_2\} \cup \{S\};\\
C &= \emptyset;\\
\mu &= [\ ]_1;\\
w &= S\omega;\\
R &= \{S \to St_1, S \to St_2\}\\
&\quad \cup \{S \to \lambda, t_1 \to \lambda, t_2 \to \lambda\}\\
&\quad \cup \{a \to \alpha|_{\neg t_1} \mid a \to \alpha \in T_1\}\\
&\quad \cup \{a \to \alpha|_{\neg t_2} \mid a \to \alpha \in T_2\}\\
&\quad \cup \{A \to A|_{\neg S} \mid A \in V \setminus \Delta\};\\
\vartheta &= 1.
\end{aligned}
$$

Here is how the system works. The initial multiset consists of the string ω which corresponds to the ET0L axiom and a symbol S. The rules of types $S \to St_1$ (or $S \to St_2$) and $S \to \lambda$ represent the "core engine" that selects non-deterministically the ET0L table to be simulated; at any moment, in a non-deterministic manner, the rule $S \to \lambda$ can be applied and the computation stops.

A rule of type $S \to St_1$ (or $S \to St_2$) forbids the applications of all rules $a \to \alpha|_{\neg t_1}$ (or $a \to \alpha|_{\neg t_2}$, respectively), but allows the execution of rules corresponding to the ET0L table T_2 (or the execution of rules corresponding to the ET0L table T_1, respectively). In this way, we correctly have simulated the application of the table t_2 (or t_1). Also, in the same computational step, either object t_1 or object t_2 is deleted from the current multiset by one of the rules $t_1 \to \lambda$ or $t_2 \to \lambda$, and the whole process can be iterated.

If the "core engine" stops (because the rule $S \to \lambda$ was applied), then we may have two cases.

(i) the current multiset is formed only by symbols from Δ and then the whole computation will stop and the system generates exactly the same vector of numbers as the Parikh vector of a corresponding successful computation of the ET0L system.

(ii) the current multiset contains symbols from $V \setminus \Delta$ and then the computation will not stop, because the rules of type $A \to A|_{\neg S}$ will cycle forever.

In conclusion we have shown that $PsP_m(ncoo, inhR) \supseteq PsET0L$.

Case 2. $PsET0L \supseteq PsP_m(ncoo, inhR)$.

Given a P system Π_m with $m \geq 1$ membranes we construct an equivalent P system $\Pi_1 = (V_\pi, C, \mu, w, R, \vartheta)$ (see Theorem 2). Without a loss of generality we consider the system Π_1 as being in the form given by Lemma 1 and Lemma 2, i.e., with a complete set of context-free rules over the disjoint sets of nonterminals and terminals.

We construct the ET0L system $H = (V, T = \{T_1, T_2, \ldots, T_k\}, \omega, \Delta)$ that simulates Π_1, as follows:

- $V = V_\pi \cup \{\#\}$;

• Let $\overline{R}$ be the set of all rules of type $a \to \alpha|_{\neg b} \in R$, $b \neq \lambda$, from R. Also, let $\overline{T_i}$, $1 \leq i \leq k$, be all saturated subsets (with respect to non-excluding inhibiting relation $\equiv_{nei}$) of $\overline{R}$. Then, we define

$$\begin{aligned} T_i = &\{a \to \alpha \mid a \to \alpha|_{\neg b} \in \overline{T_i}\} \\ &\cup \{b \to \# \mid a \to \alpha|_{\neg b} \in \overline{T_i}\} \\ &\cup \{a \to \alpha \mid a \to \alpha \in R \setminus \overline{R} \text{ and } a \neq b \ (\forall)\ c \to \alpha|_{\neg b} \in \overline{T_i}\} \\ &\cup \{\# \to \#\}, \quad 1 \leq i \leq k; \end{aligned}$$

• ω=w;
• $\Delta = V_T$, where $V_T \subset V$ is the set of terminals;

Here is how the ET0L system H simulates the computation of Π_1. First remark that, from the way we defined the saturated subsets, the conditions on the rules can be omitted (observe that two rules $r_1 : (a_1 \to \alpha_1|_{\neg b_1}$ and $r_2 : (a_2 \to \alpha_2|_{\neg b_2})$ can simultaneously rewrite symbols a_1 and a_2 iff $b_1 \neq a_2$ and $a_1 \neq b_2$) in case we divide them in different tables. In addition, we have added to each table all context-free rules of the P system that does not violate the saturation relation considered for the table. We also add rules of type $b \to \#$ if rules $\{a \to \alpha \mid a \to \alpha|_{\neg b} \in \overline{T_i}\} \in T_i$; in this way we assure that if we choose the "wrong" table, the computation will never stop since the # is produced (and therefore $\# \to \#$ will always be executed no matter which table is chosen).

In this way, if the computation stops, then the ET0L generates a language whose Parikh image is the same with the set of vectors of numbers generated by Π_1, and hence by Π_m.

In conclusion we have shown that $PsET0L = PsP_m(ncoo, inhR)$. □

4 Non-cooperative P Systems with Promoters

In this section we prove that P systems with symbol objects and non-cooperative promoted rules can generate at least the same family of vectors sets as $PsET0L$. In what concerns the upper bound we show how these P systems can be simulated by random context grammars with limited checking.

Theorem 4. $PsP_1(ncoo, proR) \supseteq PsET0L$.

Proof. We will simulate the computation of an ET0L system $H = (V, T = \{T_1, T_2\}, \omega, \Delta)$ by using a P system

$$\Pi_1 = (V_\pi, \emptyset, [\,]_1, \omega t, R, 1)$$

with

$$V_\pi = V \cup \{t, t_1, t_2, K, K_1\},$$

and the set R containing the following rules:

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

At the beginning of the simulation we have inside the region of the P system the input multiset, consisting of string ω (which is the axiom of the ET0L system H), and object t (which represents the starting trigger for the simulation of the non-deterministic 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 maximal parallel manner. One can notice that if we applied at least once such 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 it will be executed one of the rules $t_1 \to t|_A$ or $t_2 \to t|_A$, as well as rule $K \to K_1$. Since an object t is produced, then 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, 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, 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 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.

In conclusion, the above construction proves that $PsP_1(ncoo, proR) \supseteq PsET0L$. □

Lemma 3. *For any P system Π with promoted non-cooperative rules there exists an equivalent P system Π' with promoted non-cooperative rules such that, for any halting configuration of Π', all regions of Π', excepting the output one, are empty.*

Proof. (Sketch) We will proceed as in Theorem 2 by first simulating a P system with promoted non-cooperative rules and m membranes $\overline{\Pi_m}$ with a P system with non-cooperative promoted rules and one membrane $\Pi_1 = (V, C, \mu, w, R, \vartheta)$. Then, using the same coding technique as in the mentioned theorem, the problem is reduced to the ability to apply a morphism that deletes all not necessary objects. Without entering into details we present here just the set of rules R' of a P system $\Pi_1' = (V \cup \{E, E_1, E_2, D, D_1\}, C, \mu, wE, R', \vartheta)$. By $\Delta \subset V$ we denote the set of objects to be deleted in a halting configuration of Π_1.

$$
\begin{aligned}
&A \to \overline{A}|_E, \text{ for } A \in V,\\
&\overline{A} \to \overline{\overline{\alpha}} D|_B, \text{ for all rules } A \to \alpha|_B \in R,\\
&E \to E_1|_D,\\
&E_1 \to E,\\
&D \to D_1,\\
&\overline{\overline{A}} \to A|_{E_1},\\
&E \to E_2|_{D_1},\\
&D_1 \to \lambda,\\
&A \to \lambda|_{E_2}, \text{ for all symbols } A \in \Delta,\\
&E_2 \to \lambda.
\end{aligned}
$$

Observe that the technique is to use object D as a "witness", indicating that rules of type $A \to \alpha$ were applied. Fundamental for the construction is the ability of promoters (and of inhibitors as well) to react in the same time with the rules they promote (or inhibit, respectively). Also, the synchronization based on the external clock that regulates the computation is important because it allows us to check whether or not a rule was applied. □

Theorem 5. $PsRC_{lc}^{\lambda} \supseteq PsP_m(ncoo, proR)$.

Proof. Here we will prove that using random context grammars with limited checking we can simulate a P system with promoted non-cooperative rules and only one membrane. For the sake of simplicity and because of the arguments exposed in Lemma 3. we will take the system $\Pi_1 = (V, C, \mu, w, R, \vartheta)$ and we will construct a random context grammar with limited checking and λ rules $G = (N, T, P, S)$ that simulates the moves of Π_1.

Before we start, without loosing the generality, let us distinguish a terminal alphabet $\Delta \subseteq V$ and let us suppose that R contains only rules of type $A \to \alpha|_B$, $A, B \in V \setminus \Delta$, $\alpha \in V^*$. Then we have:

$$
\begin{aligned}
N = {} & (V \setminus \Delta) \cup \{S, t, t_{ok}\} \cup \{\overline{A}, \overline{\overline{A}}, \widetilde{A}\} \cup \{t_i \mid 1 \le i \le 4 * |V| + 1\}\\
& \cup \{t_i \mid 4 * |V| + 2 \le i \le 4 * |V| + 2 + k,\ k \text{ is the number of rules } A \to \alpha|_B \in R\\
& \quad \text{such that there is no } A \to \beta \in R\},\\
T = {} & \Delta
\end{aligned}
$$

The set of rules P is constructed in the following way.
First, we add to the set P the rule

$$(S \to wt,\ \emptyset, \emptyset),\ \text{where } w \text{ is the input string of } \Pi_1.$$

Its role is to set up a sentential form corresponding to the input multiset of Π_1; the symbol t will be used as a signal indicating that the simulation of the parallel applications of rules in Π_1 is about to start. As in Theorem 1, the trick is to "paint" all objects respecting the conditions imposed for the rules in the P system definition, and then non-deterministically check whether or not all rules that correspond to the ones in P system definition were actually applied. In this way we are able to simulate the maximal parallelism feature of the P systems. More formally, we add to the set P the rules:

$$\begin{aligned}
&(A \to \overline{A},\ \{t\}, \emptyset),\ \text{for } A \in V,\\
&(t_i \to t_{i+1},\ \emptyset, \{A_i\}),\ \text{for } A_i \in V, 1 \le i \le |V|,\\
&(\overline{A} \to \overline{\overline{\alpha}}\widetilde{A},\ \{B\}, \emptyset),\ \text{for all rules } A \to \alpha|_B,\\
&(\overline{A} \to \overline{\overline{\alpha}}\widetilde{A},\ \{\widetilde{B}\}, \emptyset),\ \text{for all rules } A \to \alpha|_B,\\
&(t_{|V|+i} \to t_{|V|+i+1},\ \emptyset, \{\overline{A_i}\}),\ \text{for } A_i \in V, 1 \le i \le |V|,\\
&(\widetilde{A} \to \lambda,\ \{t_{2*|V|+1}\}, \emptyset),\\
&(t_{2*|V|+i} \to t_{2*|V|+i+1},\ \emptyset, \{\widetilde{A_i}\}),\ \text{for } A_i \in V, 1 \le i \le |V|,\\
&(\overline{\overline{A_i}} \to A_i,\ \{t_{3*|V|+1}\}, \emptyset),\ \text{for } A_i \in V, 1 \le i \le |V|,\\
&(t_{3*|V|+i} \to t_{3*|V|+i+1},\ \emptyset, \{\overline{\overline{A_i}}\}),\ \text{for } \overline{\overline{A_i}}, 1 \le i \le |V|.
\end{aligned}$$

Now, if symbol $t_{4*|V|+1}$ is obtained, then we know that the parallelism of the P system was correctly simulated. However, we do not know whether or not the corresponding configuration of the P system admits a new transition (recall that in the P system we have objects, therefore there is no difference between terminals and nonterminals). So, in our grammar we have to be sure when the simulation core stops (i.e., symbol $t_{4*|V|+1}$ appears), and only afterward we have to transform (in case it was a correct simulation) the nonterminals into terminals. Therefore, non-deterministically we have to check if we can apply at least one rule of type $A \to \alpha|_B$. This task can be achieved by sequences of "linked" rules that check the presence of objects A and B:

$$\begin{aligned}
&(t_{4*|V|+1} \to t_{4*|V|+2},\ \{A\}, \emptyset),\\
&(t_{4*|V|+2} \to t_{ok},\ \{B\}, \emptyset).
\end{aligned}$$

Next, if object t_{ok} appears in the sentential form then it means that conditions for applying other rules are fulfilled and so we can restart the whole process or we can non-deterministically stop. We have also the rules:

$$(t_{ok} \to t, \emptyset, \emptyset),\quad (t_{ok} \to \lambda, \emptyset, \emptyset).$$

In this way we have shown that random context grammars with limited checking are able to simulate P systems with promoted non-cooperative rules. □

5 Conclusions

We have defined a particular form of random context grammars, namely random context grammars with limited checking. We have proved that such grammars can generate strictly more than the family $ET0L$. Even if it was not proved whether or not the random context grammars with limited checking are as powerful as Turing machines, they are still useful when characterizing P systems with non-cooperative inhibited rules. Moreover, we conjecture that such grammars can be used to give an upper bound for the family of sets of vectors generated by P systems with non-cooperative rules and one catalyst. We have shown that P systems with inhibited non-cooperative rules generate the same Parikh sets as ET0L systems, therefore we have for "free" all the proved properties of this classical formalism. In addition, we have proved that for each P system with non-cooperative promoted (or inhibited) rules there exists an equivalent P system that halts having all the regions empty, except the output one. Several open problems are mentioned along this work.

Acknowledgments

This work was done while the author was funded by AECI, Spanish Ministry of Foreign Affairs. The author is very thankful to all colleagues from the Research Group on Natural Computing, University of Seville for providing a very good scientific atmosphere.

References

1. A. Alhazov, D. Sburlan: Ultimately confluent rewriting systems. Parallel multiset–rewriting with contexts. Submitted 2004.
2. P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg: Membrane systems with promoters/inhibitors. *Acta Informatica*, 38, 10 (2002), 695–720.
3. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989.
4. M. Ionescu, D. Sburlan: On P systems with promoters/inhibitors. *JUCS*, 10, 5 (2004), 581–599.
5. Gh. Păun: *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
6. G. Rozenberg, A. Salomaa: *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.
7. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.