Acta Cybernetica **15** (2000) 9-23.

P Systems with Communication Based on Concentration

Jürgen Dassow [*] Gheorghe Păun [†]

Abstract

We consider a variant of P systems where the communication of objects is controlled by the "concentration" of these objects: after each evolution step, the objects are redistributed among the regions of the system in such a way that each region contains the same number of copies of each object (plus/minus one, when the number of objects is not divisible by the number of regions). We show that P systems of this form, with only one flip-flop catalyst but without using other control ingredients, can generate the Parikh images of all matrix languages. When an unbounded number of catalysts is available, a characterization of recursively enumerable sets of vectors of natural numbers is obtained (by systems with only one membrane).

1 Introduction

The P systems are a class of distributed parallel computing devices of a biochemical type (so, they belong to the rather active area of Molecular Computing) which were recently introduced in [11]; an early survey can be found in [12].

In short, in the basic model one considers a *membrane structure* consisting of several cell-membranes which are hierarchically embedded in a main membrane, called the *skin* membrane. The membranes delimit *regions*, where we place *objects*, elements of a finite set (an alphabet). The objects evolve according to given *evolution rules*, which are associated with the regions. An object can evolve independently of the other objects in the same region of the membrane structure, or in cooperation with other objects. In particular, we can consider *catalysts*, objects which do not evolve alone, but only assist other objects to evolve. The evolution rules are given in the form of transition rules for multisets, can be the subject of a given priority relation, and in their right hand members contain symbols $(a, here), (a, out), (a, in_j)$, where a is an object. The meaning is that one

[*]Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, D-39016 Magdeburg, Germany, E-mail: dassow@iws.cs.uni-magdeburg.de

[†]Corresponding author. Research supported by Alexander von Humboldt Foundation. Institute of Mathematics of the Romanian Academy, PO Box 1 - 764, 70700 Bucureşti, Romania, E-mail: gpaun@imar.ro

occurrence of the symbol a is produced and remains in the same region, is sent out of the current region, or is sent to the region associated with membrane j (which should be reachable from the region where the rule is applied), respectively. The membranes can be ***dissolved***. When such an action takes place, all the objects of the dissolved membrane remain free in the membrane placed immediately outside, but the evolution rules of the dissolved membrane are lost. The skin membrane is never dissolved.

The application of evolution rules is done in a maximally parallel manner: at each step, all objects which can evolve should evolve.

Starting from an initial configuration and using the evolution rules, we get a *computation*. We consider a computation completed when it halts, no further rule can be applied. The multiplicity of objects present in a designated membrane in a halting configuration is the result of the computation. Thus, in this way we compute vectors of natural numbers.

Many variants are considered in [3], [4], [5], [10], [11], [13], [14], [16]. Most of them are computationally complete, equal in power to Turing machines. When an enhanced parallelism is provided, for instance, by allowing membrane division, then linear time solutions to NP-complete problems can be obtained, [7], [10], [14], [19].

One of the most non-realistic features of many of these variants is the use of the target indications *out* and in_j; expecially the last one is rather far from biochemistry. Attempts to get rid of indications of the form in_j were already done in [13] (where electrical charges are used instead of labels: each object is marked with $+, -$, or 0 and the same with the membranes; when an object is introduced with a mark $+$ or $-$, it will be passed to a membrane of the opposite sign, nondeterministically chosen among the neighboring membranes) and in [15] (indications of the form *in*, without any membrane specification, are used, associated with other ingredients which are used to control the communication).

Here we make one more step "towards reality": the main way of communicating chemical objects in biochemistry is based on differences of concentration (gradient) among regions of a cell, see [8], [9]. We consider here a variant of P systems where this idea alone governs all communications (that is, we remove all commands *out* and in_j, and we use only communication driven by the concentration difference). We use no other control ingredients (priority among evolution rules, actions controlling the thickness of membranes, such as the dissolving action, etc), but only catalysts (in the powerful form of bistable catalysts, able to change their state among two states, in a flip-flop manner). Using only one catalyst, we cover in this way at least the Parikh images of matrix languages. When an arbitrary number of bistable catalysts is used, we can characterize the recursively enumerable sets of vectors of natural numbers. Systems consisting of one membrane only are enough (hence in such a case only the communication to the outer region of the system is possible).

2 A Remark on Matrix Grammars

Before we consider P systems we briefly recall the definition of matrix grammars and their associated languages because we shall use those grammars in the study of the generative power of P systems. Moreover, we add a new normal form for matrix grammars which is useful in this paper and is of some interest in the theory of matrix grammars. For further elements of formal language theory we refer to [17]. (We only mention that V^* is the free monoid generated by V under the operation of concatenation; λ is the empty string, $|x|$ is the length of $x \in V^*$ and $\Psi_V(x)$ is the Parikh vector associated with $x \in V^*$.)

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

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \dots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied; we say that these rules are applied in the *appearance checking* mode.) If $F = \emptyset$, then the grammar is said to be without appearance checking (and F is no longer mentioned).

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When we use only grammars without appearance checking, then the obtained family is denoted by MAT. By $PsMAT$ we denote the Parikh sets associated with languages in MAT.

It is known that $MAT \subset MAT_{ac} = RE$ and that each one-letter language in the family MAT is regular, [6]. Further details about matrix grammars can be found in [2] and in [17].

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \dagger\}$, with these three sets being mutually disjoint, and the matrices in M are of one of the following forms:

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

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \dagger$ appearing in matrices of type 3; $\dagger$ is a trap symbol, once introduced, it is never removed. A matrix of type 4 is used only once, at the last step of a derivation.

Furthermore, for any symbol $X \in N_1$, there is a matrix whose left-hand side of the first rule is X.

According to Lemma 1.3.7 in [2], for each matrix grammar there is an equivalent matrix grammar in binary normal form.

We now introduce a stronger normal form.

A matrix grammar $G = (N, T, S, M, F)$ is said to be in *strong binary normal form*, if $N = N_1 \cup N_2 \cup \{S, \dagger\}$, with these three sets being mutually disjoint, and the matrices in M are of one of the following forms:

1. $(S \to XA)$ with $X \in N_1$, $A \in N_2$,
2. $(X \to Y,\ A \to \alpha)$ with $X, Y \in N_1$, $A \in N_2$, $\alpha \in N_2^2 \cup N_2 \cup T \cup \{\lambda\}$,
3. $(X \to Y,\ A \to \dagger)$ with $X, Y \in N_1$, $A \in N_2$,
4. $(X \to \lambda)$ with $X \in N_1$.

Moreover, there is only one matrix of type 1 (the start matrix) and only one of type 4 (the final matrix) and F consists exactly of all rules $A \to \dagger$ appearing in matrices of type 3, where $\dagger$ is the trap symbol, again. The matrix of type 4 is used only once, in the last step of a derivation such that after the application of this final matrix only terminal symbols and possibly some trap symbols are left. Finally, we may assume that each control symbol from N_1 appears at least once on the left-hand side of the first production of a matrix in M and that the final control symbol only appears on the left-hand side of the final production.

Lemma 1. *For any matrix grammar G with appearance checking there is a matrix grammar G' in strong binary normal form such that $L(G') = L(G)$.*

Proof. Let $G = (N, T, S, M, F)$ be a matrix grammar. Without loss of generality, we can assume that G is in binary normal form as given above. Thus $N = N_1 \cup N_2 \cup \{S, \dagger\}$. Let n be the cardinality of N_2 and $N_2 = \{A_k \mid 1 \le k \le n\}$. We immediately obtain the corresponding matrix grammar in strong binary normal form $G' = (N', T, S, M', F')$ in the following way:

- $N' = N \cup \{Z_k \mid 0 \le k \le n\}$, where the Z_k, $0 \le k \le n$, are new symbols not contained in N.
- $M' = (M \setminus \{(X \to \lambda,\ A \to \alpha) \mid (X \to \lambda,\ A \to \alpha) \in M\}) \cup M''$ with
  $$\begin{aligned} M'' = \ & \{(X \to Z_0, A \to \alpha) \mid (X \to \lambda, A \to \alpha) \in M\} \cup \\ & \{(Z_{k-1} \to Z_k,\ A_k \to \dagger) \mid 1 \le k \le n\} \cup \{(Z_n \to \lambda)\}, \end{aligned}$$
- $(Z_n \to \lambda)$ is the final matrix.

Obviously, instead of one of the final matrices $(X \to \lambda,\ A \to \alpha)$ of G we use $(X \to Z_0,\ A \to \alpha)$, check the presence of an element of N_2 by the rules $(Z_{k-1} \to Z_k,\ A_k \to \dagger)$, $1 \le k \le n$, and finally terminate by $(Z_n \to \lambda)$. □

3 P Systems: Some Variants

We first introduce the basic class of P systems, then we briefly define several variants. The definitions are not completely formal; for details, the reader is referred to the papers listed in the bibliography.

A *membrane structure* is a construct consisting of several *membranes* placed in a unique "skin" membrane; we identify a membrane structure with a string of correctly matching parentheses, placed in a unique pair of matching parentheses; each pair of matching parentheses corresponds to a membrane. Graphically, a membrane structure is represented by a Venn diagram.

Each membrane identifies a *region*, delimited by it and the membranes immediately inside it (if any). A membrane without any other membrane inside it is said to be *elementary*.

Figure 1 represents a membrane structure, described by the string

$$\mu = [_1[_2[_3[_4\ ]_4]_3[_5\ ]_5]_2[_6[_7\ ]_7[_8\ ]_8]_6[_9\ ]_9]_1.$$

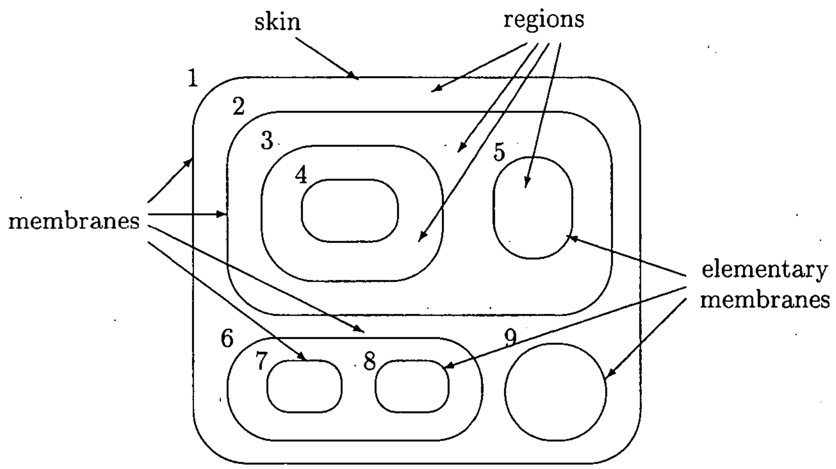

Figure 1. A membrane structure.

If in the regions delimited by the membranes we place multisets of objects from a specified finite set V, as well as evolution rules for these objects, then we obtain a *P system*.

More precisely, a P system (of degree $m, m \geq 1$) is a construct

$$\Pi = (V, T, C, \mu, w_1, \ldots, w_m, R_1, , \ldots, R_m),$$

where:

(i) V is an alphabet; its elements are called *objects*;

(ii) $T \subseteq V$ (the *output* alphabet);

(iii) $C \subseteq V, C \cap T = \emptyset$ (*catalysts*);

(iv) μ is a membrane structure consisting of m membranes (labeled with $1, 2, \ldots, m$);

(v) $w_i, 1 \leq i \leq m$, are strings over V associated with the regions $1, 2, \ldots, m$ of μ; they represent multisets of objects present in the regions of μ (the multiplicity of a symbol in a region is given by the number of occurrences of this symbol in the string corresponding to that region);

(vi) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* over V associated with the regions $1, 2, \ldots, m$ of μ.
The evolution rules are of the forms $a \rightarrow v$ or $ca \rightarrow cv$, where $a \in V - C$, $c \in C$, and v is a string over

$$(V \times \{here, out\}) \cup (V \times \{in_j \mid 1 \leq j \leq m\}).$$

When presenting the evolution rules, the indication "here" is often omitted.

The membrane structure and the multisets w_i, $1 \leq i \leq m$, in Π constitute the *initial configuration* of the system. We can pass from a configuration to another one by using the evolution rules. This is done in parallel: all objects, from all membranes, that can be the subject of local evolution rules, should evolve simultaneously.

The use of a rule $ca \rightarrow cv$ (similarly for rules $a \rightarrow v$) in a region with a multiset w means to subtract the multiset a from w, then to follow the prescriptions of v: if an object appears in v in the form $(b, here)$, then it remains in the same region; if we have (b, out), then a copy of the object b will be introduced in the membrane immediately outside the region of the rule $ca \rightarrow cv$ (if the rule is applied in the skin membrane, then the object leaves the system); if we have (b, in_i), then a copy of b is introduced in the membrane with the label i, providing that this membrane is adjacent to the region of the rule $ca \rightarrow cv$, otherwise the rule cannot be applied

A sequence of transitions between configurations of a given P system Π is called a *computation* with respect to Π. A computation is *successful* if and only if it halts, that is, there is no rule applicable to the objects present in the last configuration. The result of a successful computation consists of the vector of natural numbers which specify the number of copies of terminal objects which leave the skin membrane (e.g., if $T = \{a, b, c\}$ and no copy of a, four copies of b, and two copies of c leave the system, then the computed vector is $(0, 4, 2)$.)

Several further ingredients were considered in the literature: priority relations among evolution rules, the action of dissolving a membrane, an opposite action to dissolving, which makes thicker the membranes, inhibiting the communication, "electrical charges" associated with objects and membranes, used for controlling the communication [13], bistable catalysts [16]. We introduce here only the last of these ingredients, because it will be also used in our variant of P systems.

The bistable catalysts are catalysts able to change their state among two possibilities, c and $\bar{c}$. Specifically, we allow rules of the forms $ca \rightarrow \bar{c}v$ and $\bar{c}a \rightarrow cv$, but not also of the forms $ca \rightarrow cv$ and $\bar{c}a \rightarrow \bar{c}v$; that is why we also call these catalysts *flip-flop catalysts.*

For the power of P systems which use certain combinations of these ingredients we refer to the papers mentioned at the end of the present work. In particular, several characterizations of the recursively enumerable sets of vectors of natural numbers can be found in [11], [13], [15].

4 Definition of P Systems with Communication Based on Concentration

The indication in_i looks rather unrealistic from a biochemical point of view, so it is of interest to try to avoid its use. Actually, we will remove all indications *here, out*, in_i, by controlling the communication of objects only by means of their concentration in the regions of the system.

At the first sight, this can be done in a very simple way: in the set V of objects we consider a subset, $V_c \subseteq V$, and whenever such objects are introduced, they are immediately redistributed among all regions of the system in such a way that all regions will have the same number of occurrences of each symbol from V_c; a difference of one occurrence is allowed when the total number of occurrences is not divisible by the number of regions. Note that a system with m membranes defines $m+1$ regions, m regions inside the system, plus the outer region.

Note also that always we take into consideration each object in V_c separately and we do not count these objects (and then redistribute them) as being indistinguishable.

A problem appears when the number of copies of one object to be redistributed is not a multiple of $m+1$, where m is the number of membranes in the system. In such a case, the "remainder objects" are redistributed according to the following "efficiency principle": we move objects at the lowest distance (crossing the smallest number of membranes). For instance, if we have $m+2$ objects in a region, then all regions (including the outer region) will get one object, while the region where the objects were produced will remain with two copies. If we have started with $m+3$ objects, then each region will get one object, one further object will remain in the region where the objects were produced, and one further object will be sent to one of the neighboring regions, randomly choosen.

Because in the proofs in the following sections we will always introduce objects to be communicated in only one region at a time, the above mentioned principle will be enough to govern the communication, so we ignore the more complex situations which can appear.

In this framework, no indication *here, in, out* is necessary. The objects in V_c will go in or out, according to their concentration, the objects in $V - V_c$ remain in the region where they are introduced.

Of course, no catalyst can appear in V_c and all terminal symbols are in this set (we have to read the result of a computation outside the system, hence we have to send out of the system the terminal symbols).

How the redistribution of objects takes place in a practical circumstance is not considered here. We assume that the redistribution is done instantaneously, always correctly, and at the level of the whole system in a step, by a magical mechanism which is not explicit in our model.

We denote by $Ps(\Pi)$ the set of vectors of natural numbers computed by a P system Π and by $PsLP_m(con, 2Cat_k)$ the family of sets $Ps(\Pi)$ of this form generated by P systems which communicate by concentration, use at most k bistable catalysts, and have at most m membranes; when m or k are not specified, we replace them with $*$. (The notation reminds the notion of a *Parikh set*, $\Psi_V(L)$, associated with a language $L \subseteq V^*$.) We also denote by $PsRE$ the Parikh images of recursively enumerable languages (the family of such languages is denoted by RE), which is nothing else than the family of recursively enumerable sets of vectors of natural numbers.

5 The Power of P Systems with Concentration-Based Communication

First we show that systems with one pair of catalysts are sufficient to generate all matrix languages (without appearance checking).

Theorem 1. *$PsMAT \subset PsLP_*(con, 2Cat_1)$.*

Proof. We first prove the inclusion.

Let us consider a matrix grammar without appearance checking, $G = (N, T, S, M)$, in the binary normal form (that is, $N = N_1 \cup N_2 \cup \{S\}$). Assume that it contains n two-rule matrices, labeled in a one-to-one manner with $m_1, m_2, \dots, m_n$.

We construct the P system (of degree $n+1$)

$$\Pi = (V, V_c, T, C, \mu, w_0, w_1, \dots, w_n, R_0, R_1, \dots, R_n),$$

with

$$\begin{aligned}
V &= N \cup T \cup V_c \cup C \cup \{D, E, E', E'', \bar{E}, \dagger, \#\} \cup \{X' \mid X \in N_1\},\\
V_c &= \{X_i, A_i \mid m_i = (X \to \alpha, A \to \beta) \in M, 1 \le i \le n\}\\
&\cup \{\bar{X} \mid X \in N_1\} \cup T \cup \{\bar{a} \mid a \in T\},\\
C &= \{c, \bar{c}\},\\
\mu &= [_0 [_1\ ]_1 [_2\ ]_2 \cdots [_n\ ]_n]_0,\\
w_0 &= XADEc, \text{ for } (S \to XA) \text{ being the initial matrix of } G,\\
w_i &= Ec, \text{ for all } i = 1, 2, \dots, n,
\end{aligned}$$

and with the following sets of evolution rules:

1. The set R_0 contains the following rules:
   (a) $X \to X_i^{n+2}$,
   $cA \to \bar{c}A_i^{n+2}$, for all $1 \le i \le n$ such that $m_i : (X \to \alpha, A \to \beta) \in M$,
   (b) $cD \to \bar{c}\dagger$,
   $\dagger \to \dagger$,
   (c) $\bar{c}\bar{X} \to cX$, for all $X \in N_1$,
   (d) $\bar{c}E'' \to c\bar{E}$,
   (e) $E \to E'$,
   $E' \to E''$,
   $E'' \to E$,
   (f) $\bar{a} \to a^{n+2}$,
   $a \to \#$, for all $a \in T$,
   (g) $X_i \to \#$, for all $X \in N_1, i = 1, 2, \ldots, n$,
   $A_i \to \#$, for all $A \in N_2, i = 1, 2, \ldots, n$.
2. For each $i = 1, 2, \ldots, n$ such that $m_i : (X \to \alpha, A \to \beta)$, the set R_i contains the following rules:
   (a) $cX_i \to \bar{c}\bar{\alpha}^2$
   (of course, if $\alpha = \lambda$, then the rule is $cX_i \to \bar{c}$),
   (b) $\bar{c}A_i \to c\bar{\beta}^2$,
   where $\bar{\beta}$ is $\bar{a}$ for $\beta = a, a \in T$ and $\bar{\beta} = \beta$ if $\beta \in N_2^*$,
   (c) $cA_i \to \bar{c}\dagger$,
   $\bar{c}E \to c\dagger$,
   $\dagger \to \dagger$,
   (d) $Y_j \to \#$, for all $Y \in N_1, 1 \le j \le n, j \ne i$,
   $B_j \to \#$, for all $B \in N_2, 1 \le j \le n, j \ne i$,
   (e) $\bar{Y} \to \#$, for all $Y \in N_1$,
   $B \to \#$, for all $B \in N_2$,
   $a \to \#$,
   $\bar{a} \to \#$, for all $a \in T$.

The system works as follows.

The object $\dagger$ is a trap object, once introduced it can evolve forever, thus the computation will never finish. The object # is a dummy one, non-active.

Each matrix from M is simulated in Π in three steps, as follows.

Assume that in the skin membrane we have one symbol from N_1, some symbols from N_2, the symbols D, E, and the catalyst c (plus occurrences of the dummy symbol, which we will ignore from now on); initially, we have here the multiset represented by $XADEc$, where $(S \to XA)$ is the initial matrix of M. Assume that at the same time in each membrane $1, 2, \ldots, n$ we have the objects E and c, as in the initial stage.

In the skin membrane, the symbol $X \in N_2$ will introduce $n+2$ copies of X_i, for some $1 \leq i \leq n$, and, at the same time, the catalyst plus a symbol $A \in N_2$ will introduce $n+2$ symbols A_j, for some $1 \leq j \leq n$, while the catalyst becomes $\bar{c}$; moreover, E will be replaced by E'. The symbols X_i, A_j should be redistributed. Because we have exactly $n+2$ copies of each of them, exactly one copy of each of them will be placed in each region of the system (one symbol leaves the system). Note that if the catalyst is not used by a rule $cA \rightarrow \bar{c}A_j^{n+2}$, then it must be used by the rule $cD \rightarrow \bar{c}\dagger$ and the computation never stops.

We distinguish eight cases, according to the symbols present in a membrane $i, 1 \leq i \leq n$, after one evolution step (we denote by z_i the string representing the multiset of these symbols):

1. $z_i = X_iA_iEc$. Using the rule $cX_i \rightarrow \bar{c}\bar{\alpha}^2$ we pass to (the multiset represented by) $z' = \bar{\alpha}^2A_iE\bar{c}$. One copy of $\bar{\alpha}$ is sent to the skin membrane, one remains here. Using the rules $\bar{c}A_i \rightarrow c\bar{\beta}^2$ and $\bar{\alpha} \rightarrow \#$, we pass to $z'' = \#\bar{\beta}^2Ec$. One copy of each symbol from $\bar{\beta}$ is sent to the skin membrane, one remains in membrane i and at the next step is replaced by $\#$.

   Note that if we do not use the rule $cX_i \rightarrow \bar{c}\bar{\alpha}^2$ at the first step, then we have to use the rule $cA_i \rightarrow \bar{c}\dagger$; similarly, if we do not use the rule $\bar{c}A_i \rightarrow c\bar{\beta}^2$ at the second step, then we have to use the rule $\bar{c}E \rightarrow c\dagger$. In both cases, the computation will never halt.

   Simultaneously, in the skin membrane we proceed as follows. After the first step we here have a multiset $z_1 = X_iA_iDE'\bar{c}u$, where u consists of all symbols from N_2 which remain here (plus dummy symbols). At the same time, we receive from membrane i a symbol $\bar{Y}$, for some $Y \in N_1$ (or nothing, if the matrix m_i was a terminal one, $m_i : (X \rightarrow \lambda, A \rightarrow x)$), and E' is replaced by E''. At the next step, the symbols X_i, A_i are replaced by $\#$, while $\bar{c}\bar{Y}$ are replaced by cY and E'' is replaced by E. The multiset still contains the symbols E and $\bar{c}$. At the same time, we receive from membrane i either symbols from N_2 or a symbol $\bar{a}, a \in T$. The multiset contains again D, E, c, a symbol from N_1 and some symbols from N_2, hence we return to a contents as that from the beginning. At the next step, for the symbol $\bar{a}, a \in T$, we produce $n+2$ copies of a; exactly one of these copies enters each of the regions. In all membranes of the system, a is immediately replaced by $\#$. The copy which leaves the system contributes to the result of the current computation.

   Therefore, after three steps in membrane i and three steps in the skin membrane, we have accomplished the simulation of the matrix m_i. The procedure can be iterated.

2. $z_i = X_iA_jEc$, for some $i \neq j$. At the first step we can again use the rule $cX_i \rightarrow \bar{c}\bar{\alpha}^2$, simultaneously with $A_j \rightarrow \#$, but at the next step we have to introduce the trap-object $\dagger$, because we have to use the rule $\bar{c}E \rightarrow c\dagger$. The computation never stops.

3. $z_i = X_iEc$; exactly as in case 2, without using the rule $A_j \rightarrow \#$.

4. $z_i = X_jA_iEc$, for some $j \neq i$. At the first step we have to use the rule $cA_i \rightarrow \bar{c}\dagger$ (simultaneously with $X_j \rightarrow \#$), so the computation never stops.

5. $z_i = A_iEc$; exactly as in case 4 (without using the rule $X_j \rightarrow \#$).

6. $z_i = X_jA_kEc$, for some $j \neq i, k \neq i$. We replace X_j, A_k by $\#$ and, from the point of view of membrane i, the computation can continue.

7. $z_i = X_jEc$, for some $j \neq i$; exactly as in case 6.

8. $z_i = A_jEc$, for some $j \neq i$; exactly as in case 6.

Therefore, if a symbol X_i or a symbol A_i is introduced, then the computation can correctly continue and halt if and only if both symbols X_i and A_i were introduced. At the first sight, cases 6, 7, 8 do not correctly control the computation, but this is still done: when a symbol X_j or A_k is introduced, $n+2$ copies of it are introduced; one of these copies will reach membrane j, respectively k, and these membranes check whether or not both these symbols are present and whether or not $j = k$.

The derivation in the grammar G stops by using a matrix of the form $m_i = (X \rightarrow \lambda,\ A \rightarrow \alpha)$, for some $\alpha \in T \cup \{\lambda\}$. When this matrix is simulated in Π, no symbol $\bar{Y}$ is sent from membrane i to the skin membrane. Instead of the rule $\bar{c}\bar{Y} \rightarrow cY$, we can use the rule $\bar{c}E'' \rightarrow c\bar{E}$. If no symbol from N_2 is present, then the computation stops.

As long as any nonterminal is present in the skin membrane, the computation must continue. If we have only symbols from N_1 or only symbols from N_2, then the computation will never stop (see again cases 3, 5, 7, 8 discussed above). Consequently, a computation stops if and only if it correctly simulates a terminal derivation with respect to G.

From the previous construction, it is now clear that $\Psi_T(L(G)) = Ps(\Pi)$, that is, $PsMAT \subseteq PsLP_*(con, 2Cat_1)$.

In order to prove the strictness of the inclusion we consider the P system

$$\Pi = (\{A, B, D, X, X', X'', Y, c, a, \dagger\}, \{a\}, \{a\}, \{c\}, [_1\ ]_1, cAB, R_1)$$

with

$$\begin{aligned} R_1 \ = \ & \{cA \rightarrow \bar{c}AB,\ \bar{c}A \rightarrow cA,\ A \rightarrow \dagger,\ \bar{c}A \rightarrow cXY,\ B \rightarrow Baa, \\ & a \rightarrow D,\ cB \rightarrow \bar{c}D,\ \bar{c}X' \rightarrow cX,\ X \rightarrow X', \\ & X' \rightarrow \dagger,\ \bar{c}X' \rightarrow cX'',\ cY \rightarrow \bar{c}\dagger,\ \dagger \rightarrow \dagger\}. \end{aligned}$$

We start by having one copy of the object B. The first phase of a computation consists in using $n-1, n \geq 1$ times the couple of rules $cA \rightarrow \bar{c}AB$, $\bar{c}A \rightarrow cA$, which introduces n copies of B. When using the rule $cA \rightarrow \bar{c}AB$, in parallel, we also use the rule $B \rightarrow Baa$, while in parallel with using the rule $\bar{c}A \rightarrow cA$, we use both rules $B \rightarrow Baa$ and $a \rightarrow D$. This means that for each B, one of the two copies of a

introduced by the rule $B \to Baa$ is sent out of the system and one remains inside; the copy which remains in the system is then replaced by the "dummy" object D. After these $2(n-1)$ steps, we use the rule $cA \to \bar{c}AB, \bar{c}A \to cXY$ – in parallel with $B \to Baa$ and $a \to D$. In this way, we send out of the system 1, 2, 2, 3, 3,..., $n, n, n+1$ copies of a, which is, in total $(n+1)(n+1) - 1$, for some $n \geq 1$.

Note that during this phase, we cannot use the catalyst together with another object, because of the rule $A \to \dagger$: as long as A is present, it has to evolve by one of the rules $cA \to \bar{c}AB$, $\bar{c}A \to cA$, otherwise the computation will never terminate.

After removing the object A (and introducing the objects X, Y), we have to use the rule $cB \to \bar{c}B$; one copy of B is replaced by D, the others introduce two copies of a, while X becomes X'. The rule $cB \to \bar{c}D$ must me used, otherwise we have to use $cY \to \bar{c}\dagger$. For each B, one copy of a is sent out, the other one will remain inside the system and it will be replaced by D at the next step. The catalyst returns to the non-barred version by using the rule $\bar{c}X' \to cX$. The process is iterated, and it is somewhat symmetric to the first phase: the number of copies of B is decreased step by step and, in parallel, copies of a are sent out of the system. Again, we have pairs of steps, hence we send out $n+1, n, n, \ldots, 2, 2, 1, 1$, which means in total $(n+1)(n+1)$.

When exactly one copy of B is present, we can use the rule $\bar{c}X' \to cX''$, which can be followed only by $cB \to \bar{c}D$, and then no rule can be used.

Therefore

$$Ps(\Pi) = \{2n^2 + 4n + 1 \mid n \geq 1\}.$$

By the above mentioned fact that all languages $L \in MAT$ where $L \subseteq \{a\}^*$ for some letter a are regular, we obtain $Ps(\Pi) \notin PsMAT$. □

In the proof of the strictness of the inclusion we have used a P system with one region and one catalyst. If we keep the restriction concerning the number of regions but delete that concerning the number of catalysts, we are able to generate all recursively enumerable languages.

Theorem 2. $PsRE = PsLP_1(con, 2Cat_*)$.

Proof. Because $PsMAT_{ac} = PsRE$ and because the inclusion $PsLP_1(con, 2Cat_*) \subseteq PsRE$ is straightforward, we only have to prove that $PsMAT_{ac} \subseteq PsLP_1(con, 2Cat_*)$.

Starting from a matrix grammar $G = (N, T, S, M, F)$ in strong binary normal form with $N = N_1 \cup N_2 \cup \{S, \dagger\}$ and n matrices of the form $m_i : (X \to Y,\ A \to \alpha)$, $1 \leq i \leq n$, and k matrices of the form $m_{n+j} : (X \to Y,\ A \to \dagger)$, $1 \leq j \leq k$, we construct the corresponding P system

$$\Pi = (V, T, C, [_1]_1, w_1, R_1)$$

where

$$\begin{aligned} V &= N \cup T \cup C \cup \{E, \dagger\} \cup \{X', X'' \mid X \in N_1\}, \\ C &= \{c_i, \bar{c_i} \mid 1 \leq i \leq n\} \cup \{d_j, \bar{d_j} \mid 1 \leq j \leq k\}, \\ w_1 &= XAEc_1c_2 \ldots c_n d_1 d_2 \ldots d_k, \end{aligned}$$

where $(S \to XA)$ is the start matrix in M, and the set R_1 contains the following rules:

1. For each matrix $m_i : \; (X \to Y, \; A \to \alpha)$, $1 \leq i \leq n$, we consider the rules:
$$c_iX \to \overline{c_i}Y', \; Y' \to Y, \; \overline{c_i}E \to c_i\dagger, \; \overline{c_i}A \to \overline{\alpha},$$
where $\overline{\alpha} = \alpha^2$ for $\alpha \in T$, and $\overline{\alpha} = \alpha$ otherwise.
2. For each matrix $m_{n+j} : \; (X \to Y, \; A \to \dagger)$, $1 \leq j \leq k$, we consider the rules:
$$d_jX \to \overline{d_j}Y', \; Y' \to Y'', \; \overline{d_j}A \to d_j\dagger, \; \overline{d_j}Y'' \to d_jY\,.$$
3. For the final matrix $(Z \to \lambda)$ we take the rule $Z \to \lambda$, and we also use the rule $\dagger \to \dagger$ to ensure that the computation never stops if the trap symbol $\dagger$ has been introduced.

At any moment, except after the last step of a computation, where the final rule $Z \to \lambda$ is applied, exactly one symbol from $N_1 \cup N_1' \cup N_1''$ is present. If a matrix $m_i : \; (X \to Y, \; A \to \alpha)$, $1 \leq i \leq n$, is simulated, then first the first production $X \to Y$ is evolved together with the catalyst c_i yielding Y from X in two steps, and in the second step the corresponding second rule $A \to \alpha$ must be simulated together with the catalyst $\overline{c_i}$, otherwise the trap symbol $\dagger$ is introduced. For $\alpha \in T$ we generate a second copy, which can leave the system.

For simulating a matrix $m_{n+j} : \; (X \to Y, \; A \to \dagger)$, $1 \leq j \leq k$, three computation steps in Π are necessary: if the symbol A is present, in the second step catalyst $\overline{d_j}$ must react together with A thus generating the trap symbol $\dagger$; otherwise, $\overline{d_j}$ can wait for being evolved to d_j again together with Y''.

A computation in Π now stops if and only if the final matrix has been simulated and no trap symbol $\dagger$ is present, i.e., exactly if and only if a successful derivation in G has been simulated correctly in Π.

Consequently, we obtain $\Psi_T(L(G)) = \Psi_T(L(\Pi))$, which concludes the proof. □

We close this paper with the observation that the concentration of symbols was already used in [1] as a control mechanism of L systems. We can combine the idea of [1] with that proposed in this paper and we can consider a sort of "bi-cameral D0L systems", as a pair of D0L systems (morphisms) which work separately on multisets of symbols and, after each rewriting step, redistribute the symbols in the same way as in a P system with a concentration-controlled communication. As the output of such a device we consider the union of the two multisets (or the Parikh image of this union). The power and the properties of these cooperative D0L systems remain to be investigated. Anyway, it is clear that they can induce growth functions and length sets which are not growth functions or length sets of usual D0L systems: take two different unary D0L systems; by redistribution we will get a length set which does not consist of the powers of a given natural number, as the length set of a unary D0L language is.

Acknowkedgement. The authors are very grateful to the referee for her/his detailed comments and proposals for an improvement. Especially, Lemma 1 and its use in the proof of Theorem 2 belong to the referee.

References

[1] J. Dassow, Concentration dependent 0L systems, *Intern. J. Computer Math.*, 7 (1979), 187–206.

[2] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

[3] J. Dassow, Gh. Păun, On the power of membrane computing, *J. of Universal Computer Sci.*, 5, 2 (1999), 33–49 (www.iicm.edu/jucs).

[4] R. Freund, Generalized P systems, *Fundamentals of Computation Theory, FCT'99*, Iaşi, 1999, (G. Ciobanu, Gh. Păun, eds.), *LNCS* 1684, Springer, 1999, 281–292.

[5] R. Freund, F. Freund, Molecular computing with generalized homogeneous P systems, *Proc. Conf. DNA6* (A. Condon, G. Rozenberg, eds.), Leiden, 2000, 113–125.

[6] D. Hauschildt, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Inform.*, 31 (1994), 719–728.

[7] S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.

[8] W. R. Loewenstein, *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*, Oxford Univ. Press, New York, Oxford, 1999.

[9] S. S. Mader, *Biology* (Fifth Edition), McGraw-Hill, Boston, 1996 (Chapter 6: *Membrane structure and function*, 84–102).

[10] C. Martin-Vide, V. Mitrana, P systems with valuations, in vol. *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 154–166.

[11] Gh. Păun, Computing with membranes, *J. Computer and System Sci.*, 61, 1 (2000), 108–143 (see also *TUCS Research Report* No. 208, November 1998, http://www.tucs.fi).

[12] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, 67 (Febr. 1999), 139–152.

[13] Gh. Păun, Computing with membranes. A variant, *Intern. J. of Foundations of Computer Science*, 11, 1 (2000), 167–182.

[14] Gh. Păun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages, and Combinatorics*, 6, 1 (2001), 75–90.

[15] Gh. Păun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, submitted, 1999.

[16] Gh. Păun, S. Yu, On synchronization in P systems, *Fundamenta Informaticae*, 38, 4 (1999), 397–410.

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

[18] Y. Suzuki, H. Tanaka, On a LISP implementation of a class of P systems, *Romanian J. of Information Science and Technology*, 3, 2 (2000), 173–186.

[19] Cl. Zandron, Cl. Ferretti, G. Mauri, Solving NP complete problems using P systems with active membranes, in vol. *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 289–301.

Received August, 2000