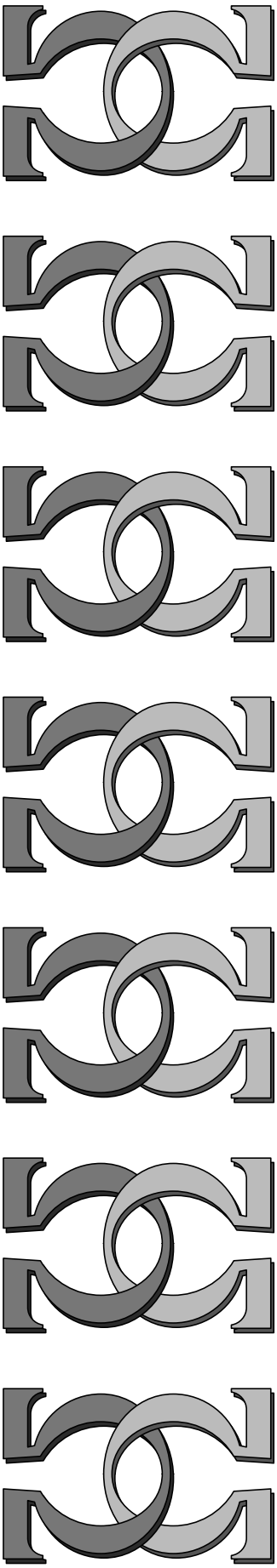

CDMTCS
Research
Report
Series

Computing with Membranes: A Variant

Gheorghe Păun
Institute of Mathematics of the Romanian Academy, Bucureşti, Romania

CDMTCS-098
March 1999

Centre for Discrete Mathematics and
Theoretical Computer Science

Computing with Membranes: A Variant

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro

Abstract. We propose here a variant of the *super-cell systems* (also called, for short, P systems), which looks less restrictive (more realistic) than the variants investigated so far. In P systems as introduced in [7] (see a survey in [8]), the use of objects evolution rules is regulated by a given priority relation; moreover, each membrane has a label and one can send objects to precise membranes, identified by their labels. We get rid of both there rather artificial (non-biochemical) features. Instead, we add to membranes and to objects an "electrical charge" and the objects are passed through membranes according to their charge. We prove that such systems are able to characterize the one-letter recursively enumerable languages (equivalently, the recursively enumerable sets of natural numbers), providing that an extra feature is considered: the membranes can be made thicker or thinner (also dissolved, as in [7]) and the communication through a membrane is possible only when its thickness is equal to 1. Several open problems are formulated.

KEYWORDS: Natural computing, P systems, Recursively enumerable languages

1 Introduction

The P systems are a class of distributed parallel computing devices of a biochemical inspiration (they can be considered as a possible branch of natural/molecular computing), borrowing ideas from Lindenmayer systems, grammar systems, the chemical abstract machine [2], multisets rewriting [1], etc.

In short, 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 obtained construct is called a *super-cell*. 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 evolve only together with other objects, but are not modified by the evolution (they just help other objects to evolve).

The evolution rules are given in the form of multisets transition rules, are subjects of a given priority relation, and in their right hand members contain pairs $(a, here), (a, out), (a, in_j)$, where a is an object. The meaning is that one occurrence of the symbol a is produced and remains in the same region, is sent out of the respective membrane, or is sent to membrane j (which should be reachable from the region where the rule is applied), respectively.

Note that the membranes are here both *separators* of sets (or multisets) of objects and evolution rules, and *communication channels*. Still more, 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 membranes are lost. The skin membrane is never dissolved.

The application of evolution rules is done in parallel.

Starting from an initial configuration (identified by the membrane structure, the objects – with multiplicities – and rules placed in its regions) and using the evolution rules, we get a *computation*. We consider a computation completed when it halts, no further rule can be applied. Two ways of assigning a result to a computation were considered: by designating an internal membrane as the output membrane, and by reading the result outside the system. We deal here with the latter variant, where a language is obtained in the natural way: we arrange the symbols leaving the system, in the order they are expelled from the skin membrane; when several objects exit at the same time, any permutation of them is accepted.

Many variants are considered in [7], [5], [10], [6].

We propose here one further variant, in an attempt to get a model as "realistic" as possible. Specifically, we get rid of two features of a formal language theory inspiration and which look far from biochemistry: the priority relation among evolution rules and the membrane labels. In order to control the communication of objects through membranes, we use "electrical charges" associated both to membranes and to objects. They can be "positive", "negative" (identified with $+, -$, respectively), or "neutral". An object marked with $+$ will enter any of the membranes marked with $-$ which are adjacent to the region where this object is produced; symmetrically, an object marked with $-$ will enter a membrane marked with $+$, nondeterministically chosen from the set of reachable membranes. The neutral objects are not introduced in an inner membrane (they can only be sent out of the current membrane).

In this way we get a particular case of a usual P system which does not seem powerful enough in order to characterize the recursively enumerable languages. Thus, we supplement the model with a further feature: besides the action of dissolving a membrane (indicated by introducing the symbol δ), we also use the action of making a membrane thicker (this is indicated by the symbol τ). Initially, all membranes have the thickness 1. If a rule in a membrane of thickness 1 introduces the symbol τ, then the membrane becomes of thickness 2. A membrane of thickness 2 does not become thicker by using further rules which introduce the symbol τ, but no object can enter or exit it. If a rule which introduces the symbol δ is used in a membrane of thickness 1, then the membrane is dissolved; if the

membrane had thickness 2, then it returns to thickness 1. If at the same step one uses rules which introduce both δ and τ in the same membrane, then the membrane does not change its thickess.

This way of controlling the communication of objects (hence the use of rules) by means of the membrane thickness is enough in order to compensate for the loss in strenght due to removing the priority and the membrane labels: a characterization of recursively enumerable sets of natural numbers is obtained. In contrast to the case encountered in [7], the proof does not give a low bound on the number of necessary membranes (or on the depth of the membrane structure, as was proved in [11] for a certain variant of P systems). We *conjecture* that such a low bound is not possible. Of course, the hierarchy on the number of membranes collapses: the proof is constructive and it starts from a matrix grammar with appearance checking; if this grammar is associated (via known constructions, see [12], [4]) with a universal type-0 grammar, then our system will also be universal – with a fixed membrane structure and evolution rules and able to simulate any given system providing that the objects in the initial configuration provide information about the particular system and its input objects. However, starting from a universal type-0 grammar (for instance, as that given in [3]), by following the constructions in [12] or [4], and then the construction in the proof below, we get a huge number of membranes. Finding accurate lower or upper bounds of this number is an *open problem*.

2 P Systems

We do not give a general definition of a P system; the reader is referred to [7] and [8] for details (definitions and examples). Rather, we introduce the variant we consider here, in a manner not completely formal, followed by an example.

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. For instance, the membrane structure in Figure 2 corresponds to the string of parentheses $[_1[_2[_3[_4\ ]_4]_3]_2]_1$.

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*. The space outside the skin membrane is called the *outer region*.

If in the regions delimited by the membranes we place multisets of objects from a specified finite set V, then we obtain a *super-cell*.

A *super-cell system* (for short, a *P system*) is a super-cell provided with evolution rules for its objects and with a designated output region. Here we consider a variant we call *P systems with polarized membranes of a variable thickness*; we only investigate systems with catalysts and with an external output.

Such a 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, for reference only, with $1, 2, \ldots, m$); each membrane is marked with one of the symbols $+, -, 0$ (these markers are written as superscripts of the right square bracket representing the membrane, e.g., $[_1[_2\,]_2^+]_1^0$);

(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 $a \to v$ or $ca \to cv$, where a is an object from $V - C$ and $v = v'$ or $v = v'\delta$ or $v = v'\tau$, where v' is a string over

$$(V - C) \cup \{a^\alpha \mid a \in V - C, \alpha \in \{+, -, out\}\},$$

and δ, τ are special symbols not in V.

The membrane structure μ and the multisets represented by $w_1, \ldots, w_n$ constitute the *initial configuration* of the system. In the initial configuration, all membranes are considered *of thickness* 1. We can pass from a configuration to another one by using the evolution rules. This is done in parallel: all objects, from all membranes, which can be the subject of local evolution rules, should evolve simultaneously.

The use of a rule $ca \to cv$ (similarly for $a \to v$) in a region with a multiset represented by a string w means removing from w one copy of a and one of c, providing that such copies exist, then to follow the prescriptions of v: if an object appears in v unmarked with one of $+, -, out$, then it remains in the same region; if we have a^{out} and the membrane has thickness 1, then a copy of the object a will be introduced in the region placed immediately outside; if we have a^+ (or a^-), then a copy of a is introduced in one of the membranes marked with $-$ (respectively $+$) and adjacent to the region of the rule $ca \to cv$ (if no such a membrane exists, then the rule cannot be applied); if the special symbol δ appears in

v and the membrane where we work has thickness 1, then this membrane is dissolved; in this way, all the objects in this region become elements of the region placed immediately outside, while the rules of the dissolved membrane are removed.

The symbols δ, τ change the thickness of the membranes where they are produced as suggested by Figure 1.

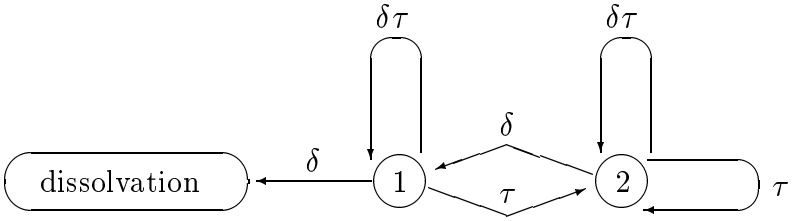

Figure 1. The effect of actions δ, τ.

That is, δ decreases the thickness, τ increases it; when both these symbols are introduced in the same region, the corresponding membrane preserves its thickness. The thickness of a membrane of thickness 2 is not further increased; once dissolved, a membrane is no longer recreated.

The communication of objects has priority on the actions δ, τ, in the sense that if at the same step one both sends a symbol through a membrane and one changes the thickness of that membrane, then one first transmits the symbol and after that one changes the thickness.

The rules are applied in parallel, an object introduced by a rule cannot evolve at the same step by means of another rule.

Note that the catalysts can pass from a region to another only by membrane dissolving actions.

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 objects from T ejected from the skin membrane, in the order they are ejected. Using these objects, we form a string. When several objects are ejected at the same time, any permutation of them is considered. In this way, a string or a set of strings is associated with each computation, that is, a language is associated with the system. (Note that the objects which remain in the system, as well as the objects which exit the system but are not elements of T are ignored.)

We denote by $L(\Pi)$ the language computed by Π in the way described above and by $LP^{\pm}(Cat, \delta, \tau)$ the family of languages generated by P systems as above, using catalysts and both actions indicated by δ, τ; when one of the features $\alpha \in \{Cat, \delta, \tau\}$ is not used, we write $n\alpha$ instead of α. If only systems with at most m membranes are used, then we add the subscript m to LP. (The superscript $\pm$ is used in order to distinguish the present variant,

with polarized membranes, from those in [7], [10], [5], where the objects are communicated by indicating the label of the target membrane.)

The following example illustrates this definition. Consider the P system of degree 4

$$\Pi = (V, T, C, \mu, w_1, w_2, w_3, w_4, R_1, R_2, R_3, R_4),$$

with

$$\begin{aligned}
&V = \{a, a', b, b', c, d, \dagger\},\\
&T = \{a\},\\
&C = \{c\},\\
&\mu = [_1[_2[_3\ [_4\ ]_4^0]_3^0]_2^0]_1^0,\\
&w_1 = \lambda,\ R_1 = \{a \to a^{out}\},\\
&w_2 = \lambda,\ R_2 = \{a \to a',\ a' \to a^{out},\ b \to \dagger,\ d \to \tau,\ \dagger \to \dagger\},\\
&w_3 = \lambda,\ R_3 = \{a \to a^{out}a,\ cb \to cb',\ cd \to cd\delta\},\\
&w_4 = a^n cd,\ R_4 = \{a \to ab\delta\}.
\end{aligned}$$

The initial configuration of the system is presented in Figure 2. With each label of a membrane, we present its polarity; here, all membranes are neutral.

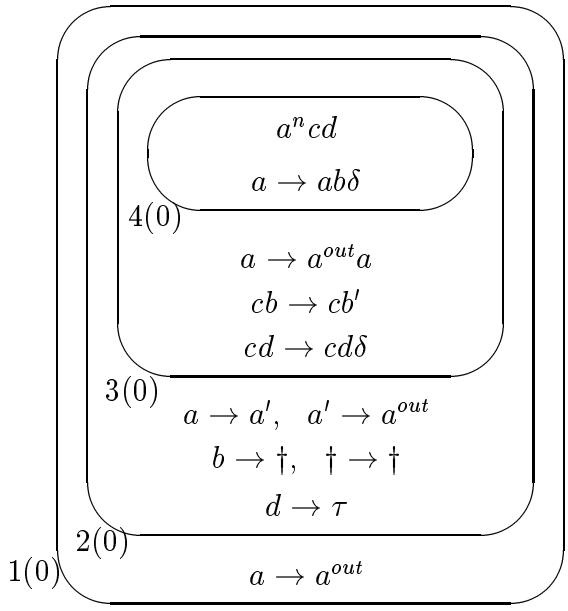

Figure 2. An example of a P system.

The system works as follows. Initially, we have objects only in membrane 4, where n copies of a and one copy of each of c and d are present. The rule $a \to ab\delta$ should be applied to each of these n copies of a; as a result we get n copies of b and the membrane is dissolved. The used rule is "lost" and the objects (n copies of a, n copies of b, one copy of c and one of d) are left free in the next region, that associated with membrane 3. If the rule $cd \to cd\delta$ is used in a moment when copies of b are still present, then such a symbol will reach region 2, where the rule $b \to \dagger$ will be used. The symbol $\dagger$ can evove indefinitely by using the rule $\dagger \to \dagger$, that is, the computation never halts (hence we have no output). In order to avoid the use of this trap-rule, we have to make sure that we dissolve membrane 3 after transforming all symbols b in b'. This means that we have to use n times (the number of occurrences of b) the rule $cb \to cb'$; in parallel, we have to use n times the rule $a \to a^{out}a$ for all copies of a which are present. Therefore, at each step we send out of membrane 3 n copies of a.

In membrane 2, each a becomes a' and, at the next step, is sent out, to membrane 1. From membrane 1, each object a leaves the system, hence participates to the generated string.

When d arrives in membrane 2, it must evolve by the rule $d \to \tau$. In this way, membrane 2 becomes thicker and from now on no symbol can leave it. This means that, after using the rule $d \to \tau$, the rule $a' \to a^{out}$ is no longer applicable. Thus, although from membrane 3 we send $n^2 + n$ copies of a to membrane 2 (n^2 during transforming b into b' and n at the step when we use the rule $cd \to cd\delta$), from membrane 2 to membrane 1 we send only n^2 copies of a (in the step when using $d \to \tau$ we also use $a \to a'$ and $a' \to a^{out}$, but at the subsequent step this latter rule will not be applied).

Consequently, the output consists of n^2 copies of the symbol a. We can say that the previous P system computes the mapping $f(n) = n^2$.

It is easy to modify the previous system in such a way to generate the (non-context-free) language $\{a^{n^2} \mid n \geq 1\}$: consider one further membrane inside membrane 4, with the rules $f \to af, f \to afcd\delta$, where f is a new object; when this membrane is dissolved, n copies of a, for some $n \geq 1$, are left free in membrane 4 (together with the catalyst c and the auxiliary symbol d; of course, no object there initially is in membrane 4).

3 The power of P systems with polarized membranes of a variable thickness

We do not start here a systematic study of the generative power of the different variants of P systems as introduced in the previous section (with or without catalysts, with or without using the actions δ and τ), but we provide only a basic result, stating that they are computationally complete. Actually, we do not characterize here the recursively enumerable languages, but only the one-letter recursively enumerable languages, which are equivalent with the recursively enumerable sets of natural numbers. Whether or not the general result,

about languages over arbitrary alphabets, is true remains as an *open* problem.

We denote by RE_1 the family of one-letter recursively enumerable languages. For the few elements of formal language theory we use here we refer to [12], [4].

Theorem. (The computational completeness theorem) $RE_1 = LP^{\pm}(Cat, \delta, \tau)$.

Proof. The inclusion $\supseteq$ follows from the Church-Turing thesis (or can be proved by a direct, straightforward but complex, construction). We prove here only the opposite inclusion.

Consider a language $L \in RE_1, L \subseteq \{a\}^*$. Take a matrix grammar with appearance checking, $G = (N, \{a\}, S, M, F)$ generating this language (one knows that each recursively enumerable language can be generated by such a grammar).

According to Lemma 1.3.7 in [4], without loss of generality we may assume that $N = N_1 \cup N_2 \cup \{S, \dagger\}$, with these three sets mutually disjoint, and that 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 $X, Y \in N_1, A \in N_2, x \in (N_2 \cup \{a\})^*$,
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 $x \in \{a\}^*$.

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. The symbols in N_1 are used to control the use of rules of the form $A \to x$ with $A \in N_2$, while $\dagger$ 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 (clearly, matrices of forms 2 and 3 cannot be used at the last step of a derivation).

We modify the grammar G by replacing each rule $X \to \lambda$ in matrices of type 4 by $X \to b$, where b is a new symbol. We denote by $G' = (N, \{a, b\}, S, M', F)$ the obtained grammar. Clearly, $L(G') = \{b\}L(G)$.

We label the matrices in M' in a one-to-one manner; assume that matrices $m_1, \ldots, m_k$ are of type 3 (with rules $A \to \dagger$ appearing in F) and matrices $m_{k+1}, \ldots, m_{k+n}$ are of types 2 and 4, for some $k \geq 0$ and $n \geq 1$.

We construct the system (of degree $2k + 3n - 1$)

$$\begin{aligned}\Pi = (V, T, C, \mu,\ & w_1, w_{1'}, w_2, w_{2'}, \ldots, w_k, w_{k'}, w_{(k+1)}, w_{(k+1)'}, w_{(k+1)''}, \\ & w_{(k+2)}, w_{(k+2)'}, w_{(k+2)''}, \ldots, w_{(k+n-1)}, w_{(k+n-1)'}, w_{(k+n-1)''}, w_{(k+n)'}, w_{(k+n)''}, \\ & R_1, R_{1'}, R_2, R_{2'}, \ldots, R_k, R_{k'}, R_{(k+1)}, R_{(k+1)'}, R_{(k+1)''}, \\ & R_{(k+2)}, R_{(k+2)'}, R_{(k+2)''}, \ldots, R_{(k+n-1)}, R_{(k+n-1)'}, R_{(k+n-1)''}, R_{(k+n)'}, R_{(k+n)''}),\end{aligned}$$

with the following components:

$$
\begin{aligned}
V &= \{X, X', X'', X''', X^{iv}, \bar{X} \mid X \in N_1\} \\
&\cup \{A, A', A'', A''', A^{iv}, \bar{A}, A^{(1)}, A^{(2)} \mid A \in N_2\} \\
&\cup \{D, D', D'', \bar{D}, a, b, c, \dagger\}, \\
T &= \{a\}, \\
C &= \{c\}, \\
\mu &= {[}_1{[}_{1'}\ {]}^{-}_{1'}{[}_2{[}_{2'}\ {]}^{-}_{2'} \cdots {[}_k{[}_{k'}\ {]}^{-}_{k'}{[}_{(k+1)}{[}_{(k+1)'}{[}_{(k+1)''}\ {]}^{-}_{(k+1)''}{]}^{-}_{(k+1)'} \cdots \\
&\quad {[}_{(k+n-1)}{[}_{(k+n-1)'}{[}_{(k+n-1)''}\ {]}^{-}_{(k+n-1)''}{]}^{-}_{(k+n-1)'} \\
&\quad {[}_{(k+n)'}{[}_{(k+n)''}\ {]}^{-}_{(k+n)''}{]}^{+}_{(k+n)'}{]}^{+}_{(k+n-1)} \cdots {]}^{+}_{(k+1)}{]}^{+}_{k} \cdots {]}^{+}_{2}{]}^{0}_{1}, \\
w_1 &= XA, \text{ for } (S \to XA) \text{ the initial matrix of } G, \\
w_{i'} &= \lambda,\ 1 \leq i \leq k, \\
w_i &= \lambda,\ 2 \leq i \leq k+n-1, \\
w_{(k+i)'} &= c,\ 1 \leq i \leq n, \\
w_{(k+i)''} &= \lambda,\ 1 \leq i \leq n, \\
R_1 &= \{X \to X^+,\ X \to X^-,\ \bar{X} \to X \mid X \in N_1\} \\
&\cup \{A \to A^+,\ A \to A^-,\ \bar{A} \to A,\ A'' \to \dagger,\ A^{(2)} \to \dagger \mid A \in N_2\} \\
&\cup \{\bar{a} \to a^{out},\ \dagger \to \dagger\}, \\
R_i &= \{X \to X^+,\ X \to X^-,\ \bar{X} \to \bar{X}^{out} \mid X \in N_1\} \\
&\cup \{A \to A^+,\ A \to A^-,\ \bar{A} \to \bar{A}^{out},\ A'' \to \dagger,\ A^{(2)} \to \dagger \mid A \in N_2\} \\
&\cup \{\bar{a} \to \bar{a}^{out},\ \dagger \to \dagger\}, \\
&\quad \text{for } i = 2, \ldots, k+n-1, \\
R_{i'} &= \{X \to Y'\tau,\ Y' \to Y'',\ Y'' \to \bar{Y}^{out},\ A \to \dagger,\ \dagger \to \dagger\} \\
&\cup \{Z \to \dagger \mid Z \in N_1, Z \neq X\} \\
&\cup \{B \to B^{(1)},\ B^{(1)} \to B^{(2)}\delta,\ B^{(2)} \to \bar{B}^{out} \mid B \in N_2, B \neq A\}, \\
&\quad \text{for } i = 1, 2, \ldots, k, \text{ with } m_i = (X \to Y, A \to \dagger), \\
R_{j'} &= \{X \to Y'D^+\tau,\ Y' \to Y'',\ Y'' \to Y''',\ Y''' \to Y^{iv}, Y^{iv} \to \bar{Y}^{(out)}, \\
&\quad Y^{iv} \to \dagger,\ \dagger \to \dagger,\ cA \to cA',\ A' \to A''\delta,\ A'' \to A''', \\
&\quad A''' \to A^{iv},\ A^{iv} \to h_{out}(x)\} \\
&\cup \{B \to B^+,\ \bar{B} \to \bar{B}^{out},\ B^{(2)} \to \dagger \mid B \in N_2\} \\
&\cup \{Z \to \dagger \mid Z \in N_1, Z \neq X\}, \\
&\quad \text{where } h_{out} \text{ is the morphism defined by } h_{out}(\alpha) = \bar{\alpha}^{out}, \\
&\quad \text{for each } \alpha \in N_2 \cup \{a\}, \\
R_{j''} &= \{D \to D'\tau\}
\end{aligned}
$$

$$\cup \{B \to B^{(1)},\ B^{(1)} \to B^{(2)}\delta,\ B^{(2)} \to \bar{B}^{out} \mid B \in N_2\},$$
for $j = k+1, \ldots, k+n,$ with $m_j = (X \to Y, A \to x)$
(note that $Y \in N_1 \cup \{b\}$).

As in the example considered in the previous section, once the symbol † is introduced, the computation is lost, because it will never halts.

The shape of the initial configuration – for the case $k = 2, n = 4$ – is indicated in Figure 3.

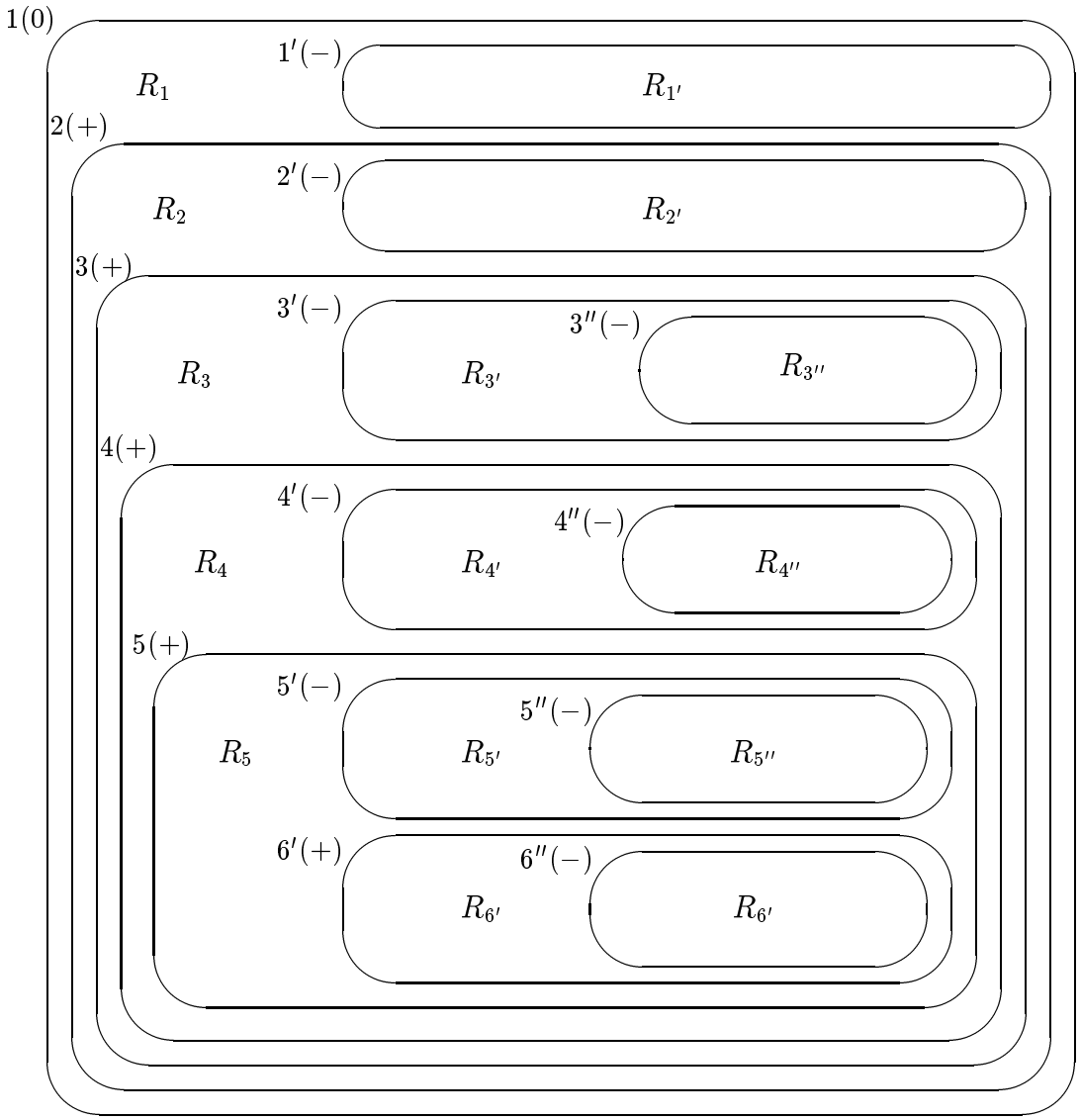

Figure 3. The shape of the system in the proof of the theorem.

The system Π works as follows.

In the initial configuration we can apply a rule only in membrane 1. Consider an arbitrary step, when in membrane 1 we have one occurrence of a symbol $X \in N_1$ and several occurrences of symbols from N_2 (initially, we have here only one object from N_2). In other regions we have at most symbols $\bar{b}, D'$, and c (initially, only occurrences of the catalyst c).

All symbols in membrane 1 get an "electrical charge", in a nondeterministic manner, by means of the rules of the form $X \to X^+, X \to X^-$ and $A \to A^+, A \to A^-$. This happens also in each of the membranes $2, 3, \ldots, k+n-1$. The symbols marked in this way have to be communicated to a membrane of an opposite polarity. Always we have only two membranes where the symbols can be moved, of opposite polarities (note that membrane $(k+n-1)'$ is marked with $-$ and membrane $(k+n)'$ with $+$).

Assume that some symbols are moved in this way until reaching a membrane i', for some $1 \leq i \leq k$, associated with a matrix $m_i = (X \to Y, A \to \dagger)$ from the grammar G'. We distinguish several cases:

1. If an occurrence of A is present, then the rule $A \to \dagger$ from $R_{i'}$ must be used, hence the computation never halts.

2. If a symbol $Z \in N_1$ different from X is present, then the rule $Z \to \dagger$ must be used and the computation never halts.

3. If X is not present, but some symbols $B \in N_2 - \{A\}$ are present, then we have to use the rules $B \to B^{(1)}, B^{(1)} \to B^{(2)}\delta$. The membrane is dissolved and the symbol $B^{(2)}$ is left free in region i. The rule $B^{(2)} \to \dagger$ is present here and it has to be used, hence the computation never halts.

4. Assume now that both X and A are present in region i'. At the first step, X is replaced with Y', one introduces the symbol τ, and all symbols $B \in N_2 - \{A\}$ are replaced by $B^{(1)}$. The membrane gets thickness 2. At the next step, we use the rules $Y' \to Y''$ and $B^{(1)} \to B^{(2)}\delta$, for all $B \in N_2 - \{A\}$ which are present in the membrane. (When simulating a derivation with respect to the grammar G', in the system there is at least one symbol B as above, because the matrices of this form are not used at the last step of a derivation. We shall see immediately that the nonterminals must travel through membranes together, otherwise the computation will never halts.) In this way, the membrane returns to thickness one. At the next step, all symbols Y and $B \in N_2 - \{A\}$ are sent out, in a barred form (and this is possible, because the membrane has the thickness 1). In this way, the matrix m_i has been correctly simulated: X is replaced with Y, providing that A is not present. We will see below that A is not only absent from membrane i', but is was absent also from membrane 1 at the beginning of the simulation of the matrix m_i.

Let us now consider the case when some symbols enter a membrane j' for some $j = k+1, \ldots, k+n$, associated with a matrix $m_j = (X \to Y, A \to x)$, with $Y \in N_1 \cup \{b\}$ and $x \in (N_2 \cup \{a\})^*$. We again distinguish several cases:

1. If a symbol $Z \in N_1$ different from X is present in membrane j', then the rule $Z \to \dagger$ must be used and the computation never halts.

2. If the right symbol X is present, but A is not present, then in the first step we use the rule $X \to Y'D^+\tau$ (hence membrane j' becomes thicker and the object D is sent to membrane j'') and $B \to B^+$ for all symbols $B \in N_2$ which are present. At the next step, we use the rule $Y' \to Y''$ in membrane j' and $D \to D'\tau$, $B \to B^{(1)}$ in membrane j'' (which becomes thicker). We continue by using the rule $Y'' \to Y'''$ in membrane j' and $B^{(1)} \to B^{(2)}\delta$ in membrane j'' (which returns to thickness 1). At the next step we use $Y''' \to Y^{iv}$ in membrane j' and $B^{(2)} \to \bar{B}^{out}$ in membrane j''. We continue by using $Y^{iv} \to \dagger$ in membrane j' (the rule $Y^{iv} \to \bar{Y}^{out}$ cannot be used, because membrane j' is of thickness 2.) The computation will never stop.

3. If both X and A are present, then the rules we use at the next steps are the following:

step 1: $X \to Y'D^+\tau,\ cA \to cA',\ B \to B^+$,
for $B \in N_2$, in membrane j' (which becomes thicker);

step 2: $Y' \to Y'',\ A' \to A''\delta$,
in membrane j' (which returns to thickness 1)
$D \to D'\tau,\ B \to B^{(1)}$,
for $B \in N_2$, in membrane j'' (which becomes of thickness 2),

step 3: $Y'' \to Y''',\ A'' \to A'''$, in membrane j',
$B^{(1)} \to B^{(2)}\delta$, for $B \in N_2$, in membrane j'' (which returns to thickness 1),

step 4: $Y'' \to Y^{iv},\ A''' \to A^{iv}$, in membrane j',
$B^{(2)} \to \bar{B}^{out}$, for $B \in N_2$, in membrane j'',

step 5: $Y^{iv} \to \bar{Y}^{out}, A^{iv} \to h_{out}(x),\ \bar{B} \to \bar{B}^{out}$,
for $B \in N_2$, in membrane j'.

In this way, the matrix m_j is correctly simulated.

In conclusion, we either correctly simulate the matrix m_j, or we introduce the trap-symbol $\dagger$. It is important to note that the symbol $\dagger$ is introduced also when a symbol $B \in N_2$ enters this membrane without also having here the symbol X; this symbol B can be any one from N_2 (equal or not to A). When simulating a matrix ($X \to \lambda$, $A \to x$) at the last step of a derivation in G, in G' we use the matrix ($X \to b$, $A \to x$), hence $Y = b$.

If the rule $cA \to cA'$ is not used at the first step, then the computation never stops: at step 5 we cannot use the rule $Y^{iv} \to \bar{Y}^{out}$, because membrane j' remains of thickness 2; this implies that the rule $Y^{iv} \to \dagger$ must be used and the derivation will never stop.

4. If no symbol from N_1 is present, but some symbols from N_2 are present in membrane j', then we have two subcases.

If also A is present and we use the rule $cA \to cA'$ in membrane j', then, at the next step we have to use the rule $A' \to A''\delta$, the membrane is dissolved, and A'' is left free in membrane j. Here, the rule $A'' \to \dagger$ must be used.

If we do not use the rule $cA \to cA'$, then all symbols are moved to membrane j'' by means of rules $B \to B^+$. In membrane j'' we use the rules $B \to B^{(1)}, B^{(1)} \to B^{(2)}\delta$ and this membrane is dissolved. The symbols $B^{(2)}$ are left free in membrane j', where the rules $B^{(2)} \to \dagger$ must be used.

Consequently, we either correctly simulate the matrix m_j, or the trap symbol $\dagger$ is introduced and the computation never ends.

Note that the barred symbols different from $\bar{b}$ are always sent out of membranes $2, 3, \ldots, k+n-1$, until reaching membrane 1. In this membrane, the nonterminals lose the bars (while $\bar{a}$ sends out of the system a copy of a). In this way, the process can be iterated.

We emphasize again that if some symbols from N_2 arrive in a membrane i', for any i, and the corresponding symbol X is not present, then the computation cannot stop. This means that always the symbols must travel together, namely together with the symol from N_1 present in the current configuration. This ensures that the simulation of matrices $(X \to Y, A \to \dagger)$ is correctly done: we are sure that A is not present, because all symbols from N_2 are present in the same membrane.

Thus, the equality $L(\Pi) = L(G)$ follows (the symbol b does not exit the system). □

Note that the system above has a number of membranes which depends on the number of matrices in the starting matrix grammar. The same assertion is true with respect to the depth of this system (the maximal number of embedded membranes). For the case of P systems as in [7] which use priorities and dissolving actions, but not catalysts, it is shown in [11] that a normal form theorem is true: systems of depth two suffice. Is an *open problem* whether or not such a result is also true in the present case.

4 Final remarks

We believe that the variant of P systems we have introduced here deserves a special attention, because of their "naturalness" in what concerns the way of moving objects from a membrane to another one (and the absence of priorities). One of the directions of research is to consider all families of the form $LP^{\pm}(\alpha, \beta, \gamma)$, for $\alpha \in \{Cat, nCat\}, \beta \in \{\delta, n\delta\}, \gamma \in \{\tau, n\tau\}$. Another fruitful idea is to work with string objects, not with symbol objects. This is done also in [7]. The strings are moved as a whole from a region to another one and the evolution rules are string transformations, for instance, based on (context-free) rewriting rules or on splicing rules (see [9] for a comprehensive presentation of the splicing operation and of other operations on strings suggested by DNA biochemistry). In the case of splicing we get a characterization of recursively enumerable languages: in [7] it is proved that systems of degree three suffice even if we do not use priorities; three membranes means one skin membrane and two inner membranes which can be labeled with + and −. We *conjecture* that a characterization of recursively enumerable languages can be obtained also in the case when using context-free rewriting rules.

References

[1] J. P. Banâtre, A. Coutant, D. Le Metayer, A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems*, 4 (1988), 133–144.

[2] G. Berry, G. Boudol, The chemical abstract machine, *Theoretical Computer Sci.*, 96 (1992), 217–248.

[3] C. Calude, Gh. Păun, Global syntax and semantics for recursively enumerable languages, *Fundamenta Informaticae*, 4, 2 (1981), 245–254.

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

[5] J. Dassow, Gh. Păun, On the power of membrane computing, *J. Univ. Computer Sci.*, 5, 2 (1999), 33–49 (see also *TUCS Research Report* No. 217, December 1998, http://www.tucs.fi).

[6] V. Mitrana, P systems with dynamic membrane structures, submitted, 1999.

[7] Gh. Păun, Computing with membranes, submitted, 1998 (see also *TUCS Research Report* No. 208, November 1998, http://www.tucs.fi).

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

[9] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Berlin, 1998.

[10] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, submitted, 1999 (see also *TUCS Research Report* No. 218, December 1998, http://www.tucs.fi).

[11] I. Petre, A normal form for P systems, *Bulletin of the EATCS*, 67 (Febr. 1999), 165–172.

[12] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.