P Systems with Energy Accounting[1]

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

Yasuhiro SUZUKI, Hiroshi TANAKA
Medical Research Institute (Bio-Informatics)
Tokyo Medical and Dental University
1-5-45 Yushima Tokyo, 113-0034 Japan
E-mail: {suzuki.com@mri.tmd.ac.jp, tanaka@tmd.ac.jp}

Abstract. We consider P systems where each evolution rule "produces" or "consumes" some quantity of energy, in amounts which are expressed as integer numbers. In each moment and in each membrane the total energy involved in an evolution step should be positive, but if "too much" energy is present in a membrane, then the membrane will be destroyed (dissolved). We show that this feature is rather powerful. In the case of multisets of symbol-objects we find that systems with two membranes and arbitrary energy associated with rules, or with arbitrarily many membranes and a bounded energy associated with rules characterize the recursively enumerable sets of vectors of natural numbers (catalysts and priorities are used). In the case of string-objects we have only proved that the recursively enumerable languages can be generated by systems with arbitrarily many membranes and bounded energy; when bounding the number of membranes and leaving free the quantity of energy associated with each rule we have only generated all matrix languages. Several research topics are also pointed out.

KEY WORDS: Molecular Computing, Turing computability, P systems
C.R. Categories: F.4.3, F.1.1

1 Introduction

P systems are a class of distributed parallel computing models introduced in [11], inspired from the way the alive cells process chemical compounds, energy, and information. In short, in the *regions* delimited by a *membrane structure* (see Figure 1 for

[1]Work supported by Grant-in-Aid for Exploratory Research No 11837005, from the Ministry of Education, Science, Sports, and Culture of Japan.

an illustration of this notion), one places *multisets* of *objects*, which evolve according to *evolution rules* associated with the regions; a *computation* consists of transitions among system *configurations*; the *result* of a halting computation is the vector of the multiplicities of objects present in the final configuration in a specified *output membrane* or of objects which leave the external membrane of the system (the *skin* membrane) during a computation. That is, a P system of this form *computes* a set of vectors of natural numbers. Many variants characterize the family of recursively enumerable sets of vectors of natural numbers, which are exactly the Parikh sets associated with recursively enumerable languages. Details can be found in [11], [1], etc. When membrane division is allowed, NP-complete problems can be solved in linear time, [14], [7], [17].

The objects in a P system can also be described by strings over a given alphabet. These strings can be considered as elements of a language (hence a usual set), or even in the multiset sense, taking care of their multiplicities. In both cases, by making use of adequate string processing operations (rewriting, splicing, replication, etc), for various classes of P systems one gets characterizations of recursively enumerable languages; details can be found in [11], [2], [10].

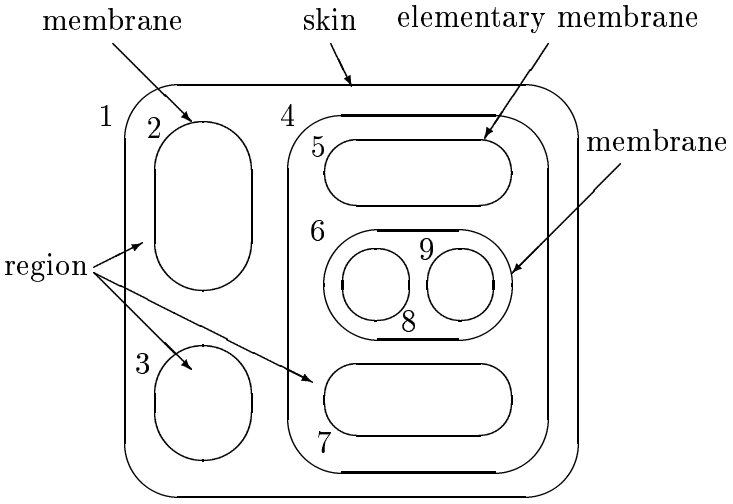

Figure 1: A membrane structure

In some sense, the P systems investigated up to now mainly correspond to the processing of chemical compounds in a cell, and not too much to the energy processing. Taking into account the energy was formulated as an explicit research topic in [15]. Here we try to fill in this gap, by considering also the energy involved in the reactions which develop in a cell. It is known that most of these reactions need energy, which is provided, for instance, by ATP (adenosine triphosphate). Thus, we may assume that some evolution rules produce energy while others consume energy. We work here with "energy quanta" and assume that all energy quantities are expressed by integer numbers. The energies of all rules used in a given step in a membrane are summed-up; if the total is positive, then the rules can be applied, otherwise the rules cannot be applied. That is, we may use only such a combination of rules which leads to a non-negative total of energy and which is maximal with this property (no further rule

can be applied to the remaining objects in the membrane such that the total energy is still positive). The energy passes from a step to the next one; initially, we assume that there is no energy in the system. If the energy accumulated at a given step in a membrane is larger than a given threshold associated with that membrane, then the membrane is dissolved and the respective energy is consumed.

We consider both symbol-objects and string-objects; because we have to count the total energy of rules used at each step, in both cases we have to take care of multiplicities of objects. In this framework, in both cases computational completeness is obtained: the recursively enumerable sets of vectors of natural numbers and the recursively enumerable languages are characterized, respectively. A trade-off between the number of membranes and the maximal (positive or negative) energy associated with a rule is found in the case of symbol-objects, either the number of membranes can be bounded, or the maximal absolute value of the energy associated with each rule can be bounded. We do not known whether or not both the number of membranes and the energy can be simultaneously bounded. Some other open problems are also formulated.

2 P Systems with Symbol-Objects

We refer to [18] for the elements of formal language theory we use here. We only specify that for a string $x \in V^*$ (V^* is the free monoid generated by the alphabet V under the operation of concatenation; the empty string is denoted by λ) and a symbol $a \in V$, we denote by $|x|$ the length of x and by $|x|_a$ the number of occurrences of the symbol a in the string x. The families of context-free, context-sensitive, and recursively enumerable languages are denoted by CF, CS, RE, respectively. For $w \in V^*, V = \{a_1, \ldots, a_n\}$, we denote by $\Psi_V(w)$ the Parikh vector of w, that is, $\Psi_V(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$; this is extended to languages in the natural way. For a family F of languages, we denote by PsF the family of Parikh sets of vectors associated with languages in F.

A membrane structure will be represented by a string of matching labeled parentheses. For instance, the membrane structure in Figure 1 is represented by

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

A multiset over an alphabet V is represented by a string over V and by all its permutations. Conversely, each string precisely identifies a multiset; the Parikh vector associated with the string indicates the multiplicities of each element of V in the corresponding multiset. Thus, when speaking of a "multiset" $w \in V^*$ we understand the multiset identified by w.

We now introduce the *P systems with energy accounting working with symbol-objects*. Such a system (of degree $m, m \geq 1$) is a construct

$$\Pi = (V, C, \mu, w_1, \ldots, w_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), d_1, \ldots, d_m),$$

where:

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

(ii) $C \subseteq V$ is the set of *catalysts*;

(iii) μ is a membrane structure consisting of m membranes, with the membranes and the regions labeled in a one-to-one manner with $1, 2, \ldots, m$; the skin membrane is labeled with 1;

(iv) $w_1, \ldots, w_m$, are multisets over V associated with the regions $1, 2, \ldots, m$ of μ;

(v) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* over V and ρ_i are partial order relations (*priorities*) over the sets R_i.

These rules are of the forms $a \to v\langle e \rangle$, $ca \to cv\langle e \rangle$, where $a \in V - C, c \in C$, v is a string over

$$(V - C) \times \{here, out, in\},$$

and e is an integer number which specifies the *energy* associated with the rule;

(v) $d_1, \ldots, d_m$ are natural numbers indicating the *dissolving thresholds* of each membrane.

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

The membrane structure and the multisets in Π constitute the *initial configuration* of the system.

The application of a rule $ca \to cv\langle e \rangle$ (the case of rules $a \to v\langle e \rangle$ is a particular one) in a region containing a multiset w means to remove a copy of the object a from w, making sure that also a copy of c is present, and to add the objects specified by v, following the prescriptions given by v: If an object appears in v in the form $(b, here)$, then it remains in the same region; if it appears in the form (b, out), then a copy of the object b will be introduced in the region which surrounds the region of the rule $cu \to cv\langle e \rangle$; if it appears in the form (b, in), then a copy of b is introduced in one of the regions of the membranes placed directly inside the region of the rule $ca \to cv\langle e \rangle$, nondeterministically chosen, if such a region exists, otherwise the rule cannot be applied.

In each step and in each membrane, the rules are used in the maximally parallel manner (all symbols which can evolve should evolve), taking into account the priority relation (a rule $r \in R_i$ is used only if no rule $r' \in R_i$ with $r' > r$ according to the relation ρ_i can be used at the same step), and with the following restriction about the total energy: in each membrane the energies of the rules used in a given step are totalized, and also added to the energy present in the membrane at the beginning of the current step; the rules can be applied only if the total energy is greater than or equal to zero. That is, we have to use only such a combination of rules with a non-negative total of energy and such that no rule can be added in such a way that the total energy is still non-negative. Of course, if the total energy is positive, this total can be arbitrarily large. There is no energy in the system at the beginning of a computation (in the initial configuration).

However, if the total energy accumulated in a given step in a given membrane i is larger than or equal to d_i, then the membrane is dissolved. The energy of the dissolved membrane is lost (it is consumed by the dissolving action). As usual in P systems, when

a membrane is dissolved its objects are left free in the surrounding membrane and its rules are lost. The skin membrane is never dissolved, hence for the skin membrane the bound d_1 acts as a restriction on the used rules, their total energy cannot exceed d_1.

Note that the catalysts never evolve, they just assist other objects to evolve. In particular, the catalysts can pass from a membrane to another one only by the membrane dissolvation.

By using the rules of Π in the way described above, we can pass from a configuration of the system to another configuration. We leave to the reader the task of formally defining a transition between two configurations. In order to make clearer the previous discussion, we have chosen to examine an **example**.

Consider the system (of degree 2, without catalysts and without priorities)

$$\Pi = (\{a, b, c\}, \emptyset, [_1[_2\ ]_2]_1, \lambda, abc, (R_1, \emptyset), (R_2, \emptyset), 1, 4),$$

with the following rules

$$\begin{aligned} R_1 &= \{a \to a(out)\langle 0\rangle\}, \\ R_2 &= \{a \to aa\langle -2\rangle,\ a \to aaa\langle -4\rangle, \\ &\quad b \to bb\langle 2\rangle,\ b \to bbb\langle 2\rangle, \\ &\quad c \to cc\langle 1\rangle,\ c \to ccc\langle 3\rangle\}. \end{aligned}$$

At the first step we can use any combination of rules from membrane 2, with two exceptions, $\{a \to aaa\langle -4\rangle, b \to bb\langle 2\rangle, c \to cc\langle 1\rangle\}$ and $\{a \to aaa\langle -4\rangle, b \to bbb\langle 2\rangle, c \to cc\langle 1\rangle\}$: in both cases, the total energy is $-4+2+1 = -1$. For the set $\{a \to aa\langle -2\rangle, b \to bb\langle 2\rangle, c \to cc\langle 1\rangle\}$, with the total energy $-2 + 2 + 1 = 1$, we have a positive total, the combination is allowed, the membrane survives, and one unit of energy remains. Thus, we are not allowed to use only a b-rule or only a c-rule, the combination is not maximal in such a case.

Assume that we use the combination $\{a \to aa\langle -2\rangle, b \to bb\langle 2\rangle, c \to cc\langle 1\rangle\}$ and pass to the next step with the multiset $aabbcc$ in membrane 2, plus one energy unit. We can use twice the rule $a \to aaa\langle -4\rangle$ (-8 energy units), but we have to provide at most further 7 units (one unit is available from the previous step). If we use twice the rule $c \to ccc\langle 3\rangle$, then we obtain in total at least 11 energy units (both objects b must evolve, and this provides 4 further energy units) and the membrane is dissolved.

We arrive in membrane 1 with six copies of a, a number of copies of b, and c and no energy unit. The copies of a will be sent out of the system, those of b and c will remain in membrane 1. The evolution of the system stops here.

A sequence of such transitions between configurations is called a *computation* of 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 is $\Psi_V(w)$, where w describes the multiset of objects from V which have left the skin membrane during the computation; we say that this vector is *generated* by Π. We denote by $N(\Pi)$ the set of all vectors generated by Π.

The family of all sets $N(\Pi)$ generated by systems of degree at most m, $m \geq 1$, using catalysts and priorities among rules, and with the maximal absolute value of energy

associated with a rule at most ε is denoted by $NEP_m(Cat, Pri, \varepsilon)$; when we do not use catalysts, we replace Cat by $nCat$, and when we do not use priorities, we replace Pri by $nPri$; if no bound on the number of membranes or on the energy associated with each rule is imposed, then we replace the corresponding parameters by $*$.

In the proofs from the next sections we need the notion of a *matrix grammar with appearance checking*.

Such a grammar 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 $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, 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.)

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}. If $F = \emptyset$, then G is said to be without appearance checking. The family of languages generated by such grammars is denoted by MAT.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$. Further details about matrix grammars can be found in [3] and in [18].

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets 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 $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
3. $(X \to Y, A \to \#)$, 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 $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ 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.

According to Lemma 1.3.7 in [3], for each matrix grammar G there is an equivalent matrix grammar G' in the binary normal form; if G is without appearance checking, then also G' is without appearance checking.

3 The Power of Systems with Symbol-Objects

We first prove that P systems with energy accounting, as defined above, with a small number of membranes are able to simulate all Turing machines. Because we use pri-

orities, this result is not new (see [11]), but its proof is much simpler than that for the case when no energy accounting is involved.

Theorem 1. *$PsRE = NEP_m(Cat, Pri, *)$, for all $m \geq 2$.*

Proof. We only have to prove the inclusion $PsRE \subseteq NEP_2(Cat, Pri, *)$. To this aim, we make use of the equality $RE = MAT_{ac}$, more exactly, of the equality $PsRE = PsMAT_{ac}$.

Let $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and matrices of the four forms mentioned above. Assume that we have s matrices of the form $(X \to \alpha, A \to x)$, with $X \in N_1, \alpha \in N_1 \cup \{\lambda\}, x \in (N_2 \cup T)^*$, and t matrices of the form $(X \to Y, A \to \#)$, $X, Y \in N_1, A \in N_2$. Consider a new symbol, f and replace the matrices $(X \to \lambda, A \to x)$ with $(X \to f, A \to x)$. We continue to denote by $(X \to \alpha, A \to x)$ the obtained matrices and by G the obtained grammar. We label by $m_i, 1 \leq i \leq s$, the matrices $(X \to \alpha, A \to x)$ and by $m_{s+j}, 1 \leq j \leq t$, the matrices of the form $(X \to Y, A \to \#)$.

We construct the P system (of degree 2)

$$\Pi = (V, \{c\}, [_1[_2\ ]_2]_1, \lambda, w_1, (R_1, \emptyset), (R_2, \rho_2), 1, 2),$$

with

$$V = N \cup T \cup \{c, f, f', Z, \} \cup \{Z_i \mid s+1 \leq i \leq s+t\},$$
$$w_2 = XAcZZ_{s+1} \ldots Z_{s+t}, \text{ for } (S \to XA) \text{ being the initial matrix of } G,$$

and the sets of rules constructed in the following way:

1. For each matrix $m_i : (X \to \alpha, A \to x) \in M, 1 \leq i \leq s$, with $\alpha \in N_1 \cup \{f\}$, we introduce in R_2 the rules

   $$X \to \alpha\langle 3i+1\rangle,$$
   $$cA \to cx\langle -3i-1\rangle.$$

2. For each matrix $m_i : (X \to Y, A \to \#) \in M, s+1 \leq i \leq s+t$, we introduce in R_2 the rules

   $$X \to Y\langle -3i\rangle,$$
   $$r_i : cA \to cA\langle 3i+2\rangle,$$
   $$r_i' : cZ_i \to cZ_i\langle 3i\rangle,$$

   with $r_i > r_i'$.

3. We also introduce the rules

   $$f \to f'\langle 3\rangle,$$
   $$Z \to Z\langle 0\rangle.$$

   in R_2.

4. The set R_1 contains the following rules:

$$\alpha \to \alpha\langle 0\rangle, \text{ for all } \alpha \in N_1 \cup N_2 \cup \{f\},$$
$$a \to (a, out)\langle 0\rangle, \text{ for all } a \in T.$$

(Note that the communication command *in* is not used in the previous construction.)

Assume that at some moment we have in membrane 2 a multiset $XwcZZ_{s+1}\dots Z_{s+t}$; initially, $w = A$, for $(S \to SA)$ the initial matrix of G.

As long as membrane 2 will exist, the rule $Z \to Z\langle 0\rangle$ can be used, hence the computation cannot stop. If we dissolve membrane 2 and still nonterminal symbols B of G are present, then the computation cannot stop because of rules $B \to B\langle 0\rangle$ from membrane 1. Therefore, before dissolving membrane 2, we have to get a multiset without any nonterminal; the occurrence of the nonterminal from N_1 is removed only when introducing the symbol f and this should be the last step of a derivation in G.

If we use a rule $X \to \alpha\langle 3i+1\rangle$ associated with a matrix $m_i : (X \to \alpha, A \to x)$, for some $1 \le i \le s$, then at the same step we have to use also the rule $cA \to cx\langle -3i-1\rangle$. Indeed, if no other rule (different from $Z \to Z\langle 0\rangle$) is used, then membrane 2 is dissolved and either a symbol from N_1 or the symbol f arrives in membrane 1 and the computation never stops. A rule $cB \to cy\langle -3j-1\rangle$ with $j < i$ will also lead to the dissolvation of membrane 2 (more than two energy units are produced); such a rule with $j > i$ cannot be used, because the total energy is negative. If we use a rule $cB \to cB\langle 3j+2\rangle$ or $cZ_j \to cZ_j\langle 3j\rangle$, for some $j \ge s+1$, then again we dissolve membrane 2 with nonterminals inside and the computation never stops. The only possibility is to corectly simulate the matrix m_i, by using simultaneously with $X \to \alpha\langle 3i+1\rangle$ the rule $cA \to cx\langle -3i-1\rangle$. We pass to the next step without any remaining energy.

In the same way, in order to use a rule $X \to Y\langle -3i\rangle$ associated with a matrix $m_i : (X \to Y, A \to \#), s+1 \le i \le s+t$, we have to use at the same step the corresponding rule $cZ_i \to cZ_i\langle 3i\rangle$, which means that A is not present in the current multiset. If A is present, because of the priority relation, the rule $cA \to cA\langle 3i+2\rangle$ must be used, and the membrane is dissolved with a nonterminal inside. A rule of the form $cA \to cx\langle -3j-1\rangle$ for some $j \le s$, cannot be used because the total energy is negative; the same assertion holds if we use a rule $cB \to cB\langle 3j+2\rangle$ or a rule $cZ_j \to cZ_j\langle 3j\rangle$, with $j < i$. If we use such a rule with $j > i$, then the membrane is dissolved with a nonterminal inside. The only continuation which does not lead to a computation which continues forever is that which correctly simulates the use of the matrix m_i.

When the symbol f is introduced we also have to completely simulate the corresponding matrix. At the next step, the rule $f \to f'\langle 3\rangle$ is used and membrane 2 is dissolved. If the derivation in G was terminal, then we send out all terminal symbols and the computation stops.

In conclusion, $N(\Pi) = \Psi_T(L(G))$, which concludes the proof. □

Theorem 2. $PsMAT \subset NEP_m(Cat, nPri, *), m \ge 2$.

Proof. In the previous proof, the priority relation is used only when simulating matrices used in the appearance checking manner. Consequently, we have the inclusion.

The inclusion is proper: let us consider the system

$$\Pi = (\{a\}, \emptyset, [_1[_2\ ]_2]_1, \lambda, a, (R_1, \emptyset), (R_2, \emptyset), 1, 1),$$

with

$$R_1 = \{a \to (a, out)\langle 0\rangle\},$$
$$R_2 = \{a \to aa\langle 0\rangle,\ a \to aa\langle 1\rangle\}.$$

At each step, the number of occurrences of the object a from membrane 2 is doubled. As long as only the rule $a \to aa\langle 0\rangle$ is used, the process can continue, but when the rule $a \to aa\langle 1\rangle$ is also used, the membrane is dissolved. All copies of a are sent out of the system and the computation stops. Consequently, we have $N(\Pi) = \{(2^n) \mid n \geq 1\}$, and this is not the Parikh image of a matrix language (all one-letter matrix languages are regular, [5]). □

Note that $PsCF \subset PsMAT$: while $PsCF$ is equal to the family of semilinear sets of vectors of natural numbers, MAT contains non-semilinear languages. An example is $\{a^n b^m \mid n \geq 1, 1 \leq m \leq 2^n\}$ (see [3]).

It is an *open problem* whether or not $PsRE$ can be characterized by P systems as above without using a priority relation among the rules (maybe also using an arbitrary number of membranes).

On the other hand, at the price of using a non-bounded number of membranes we can avoid the use of catalysts and, furthermore, we can also bound the energy associated with each rule.

Theorem 3. $PsRE = NEP_*(nCat, Pri, 1)$.

Proof. Let again $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and matrices of the four forms mentioned above. As in the proof of Theorem 1, we replace each matrix $X \to \lambda, A \to x)$ by $(X \to f, A \to x)$, for a new symbol f, and we label by $m_i, 1 \leq i \leq s$, the matrices $(X \to \alpha, A \to x)$, $\alpha \in N_1 \cup \{f\}$, and by $m_{s+i}, 1 \leq i \leq t$, the matrices $(X \to Y, A \to \#)$.

We construct the P system (of degree $s + 2$)

$$\Pi = (V, \emptyset, \mu, w_1, w_2, \ldots, w_{s+2}, (R_1, \rho_1), \ldots, (R_{s+2}, \rho_{s+2}), 1, 1, 1, \ldots, 1),$$

(note that all membranes are dissolved when one energy unit is produced) with

$$\begin{aligned}
V &= N \cup T \cup \{X' \mid X \in N_1\} \cup \{X_i \mid X \in N_1 \cup \{f\}, 1 \leq i \leq s\} \\
&\cup \{d, e, f, f', Z\} \cup \{Z_i \mid s+1 \leq i \leq s+t\}, \\
\mu &= [_1[_2[_3\ ]_3[_4\ ]_4 \cdots [_s\ ]_s[_{s+2}\ ]_{s+2}]_2]_1, \\
w_1 &= \emptyset, \\
w_2 &= XAZ, \text{ for } (S \to XA) \text{ being the initial matrix of } G, \\
w_i &= e, 3 \leq i \leq s+2,
\end{aligned}$$

and with the following sets of rules:

$$\begin{aligned}
R_1 &= \{\alpha \to \alpha\langle 0\rangle \mid \alpha \in N_1 \cup N_2 \cup \{d, e, f\}\} \\
&\cup \{a \to (a, out)\langle 0\rangle \mid a \in T\}, \\
R_2 &= \{X \to (\alpha_i, in)\langle 1\rangle \mid m_i : (X \to \alpha, A \to x), 1 \leq i \leq s\} \\
&\cup \{B \to (B, in)\langle -1\rangle \mid B \in N_2\} \\
&\cup \{f \to f'\langle 1\rangle,\ d \to d\langle 0\rangle,\ Z \to Z\langle 0\rangle,\ e \to (e, out)\langle 0\rangle\} \\
&\cup \{X \to Y'Z_i\langle 0\rangle,\ r_A : A \to d\langle -1\rangle, \\
&\quad r_i : Y' \to Y\langle -1\rangle,\ r_i' : Z_i \to f'\langle 1\rangle \mid \\
&\quad m_i : (X \to Y, A \to \#), s+1 \leq i \leq s+t\}, \\
R_{2+i} &= \{\alpha_i \to (\alpha, out)\langle 1\rangle,\ A \to (x, out)\langle -1\rangle\} \\
&\cup \{\alpha_j \to (d, out)\langle 0\rangle \mid 1 \leq j \leq s, j \neq i\} \\
&\cup \{B \to (e, out)\langle 0\rangle \mid B \in N_2\}, \\
&\quad \text{for all } 1 \leq i \leq s, m_i : (X \to \alpha, A \to x), \alpha \in N_1 \cup \{f\}, \\
&\quad \text{where } (x, out) \text{ is a short for the string of all symbols of} \\
&\quad x \text{ having associated the command } out.
\end{aligned}$$

All relations $\rho_i, 1 \leq i \leq s+2$, are empty, with the exception of $i = 2$:

$$\begin{aligned}
\rho_2 &= \{r_A > r_i,\ r_i' > r_B \mid \text{for all} \\
&\quad s+1 \leq i \leq s+t, m_i : (X \to Y, A \to \#), B \in N_2 - \{A\}\}.
\end{aligned}$$

This system works as follows. As in the proof of Theorem 1, the computation does not stop before dissolving membrane 2, because of the rule $Z \to Z\langle 0\rangle$; the (correct) dissolvation of membrane 2 is possible only by using the rule $f \to f'\langle 1\rangle$, which is introduced by a terminal matrix of G, used at the end of a terminal derivation.

Assume that we have a multiset in membrane 2 containing one copy of an object from N_1 and some objects from N_2 and T; initially, we have here XA, for $(S \to XA)$ being the initial matrix of G.

If the rule $X \to \alpha_i\langle 1\rangle$ is used, for some $m_i : (X \to \alpha, A \to x)$, then at the same step we have to use one rule of the form $B \to (B, in)\langle -1\rangle$ or a rule of the form $B \to d\langle -1\rangle$. If no such rule is applicable, then membrane 2 is dissolved and at least one object from $N_1 \cup N_2 \cup \{f\}$ is released in membrane 1, where it will evolve forever. If a rule $B \to d\langle -1\rangle$ is used, then the object d is produced and it will evolve forever both in membrane 2 and, providing that this membrane is dissolved, in membrane 1. Thus, we have to use one rule of the form $B \to (B, in)\langle -1\rangle$, hence an object from N_2 is sent to an inner membrane.

Now, if the object α_i arrives in a membrane $2+j$ with $i \neq j$, then the rule $\alpha_i \to (d, out)\langle 0\rangle$ from membrane $2+j$ is used and the trap-object d is introduced in membrane 2; the computation will never stop.

Let us now look to what happens in membrane $2+i$, where the symbol α_i arrives. If no other rule will be applied in this membrane, then the object α is sent out and the membrane is dissolved; the object e, present here from the beginning of the

computation, is left free in membrane 2 and then it is sent to membrane 1, where it evolves forever. The only rule with a negative associated energy in membrane $2+i$ is $A \to (x, out)\langle -1\rangle$, corresponding to the matrix m_i of G. Only one copy of A can evolve by such a rule, otherwise the total of energy is negative. If any other symbol B from N_2 is present, the rule $B \to (e, out)\langle 0\rangle$ should be used, and again the computation never finishes.

Therefore, the object from N_2 which has been sent to a lower membrane by the rule used at the same time with $X \to (\alpha_i, in)\langle 1\rangle$ is exactly that corresponding to matrix m_i. In this way, we correctly simulate this matrix m_i.

Assume now that in membrane 2 we use a rule $X \to Y' Z_i\langle 0\rangle$, associated with a matrix $m_i : (X \to Y, A \to \#)$, for some $s+1 \leq i \leq s+t$. No rule of the form $B \to (B, in)\langle -1\rangle$ can be used at the same time (the total energy would be negative). At the next step, the rule $Z_i \to f'\langle 1\rangle$ is applicable. Because of the priority relation, it forbids the use of any rule of the form $B \to (B, in)\langle -1\rangle$ with $B \neq A$. If any copy of A would exist, then the rule $A \to d\langle -1\rangle$ must be applied. The object d is introduced and the computation will never stop. This rule has priority over the rule $Y' \to Y\langle -1\rangle$. If no copy of A exists, then this latter rule may be applied, and the object Y is introduced. No energy remains, membrane 2 survives, while the matrix m_i has been correctly simulated.

The process can be iterated, hence any derivation in G can be simulated in Π and, conversely, all computations which will end correspond to correct derivations in G. When, after simulating a matrix $m_i, 1 \leq i \leq s$, we introduce the object f, then at the next step we have to use the rule $f \to f'\langle 1\rangle$ and membrane 2 is dissolved. If no nonterminal is present, then, after sending out of the system all terminal symbols, the computation is finished, otherwise it continues forever. (Note that the object f' is never evolving.)

In conclusion, $N(\Pi) = \Psi_T(L(G))$, hence we have the inclusion $PsRE \subseteq NEP_*(nCat, Pri, 1)$. □

Corollary 1. $PsMAT \subset NEP_*(nCat, nPri, 1)$.

Proof. In the previous construction, the priority relation is used only when simulating the matrices used in the appearance checking mode. In view of Theorem 2, the inclusion is proper. □

We do not know whether or not the number of membranes and the maximal absolute value of energy associated with rules can be simultaneously bounded without decreasing the power of our systems. Another *open problem* of interest is whether or not we can get rid of priorities (using, if necessary, both arbitrarily many membranes and arbitrarily large quantities of energy associated with the rules).

4 The Case of String-Objects

Let us now consider the case when the objects of our systems are represented by strings and the evolution rules are rewriting rules. Formally, a P system of this type

is a construct

$$\Pi = (V, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m, d_1, \ldots, d_m),$$

where:

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

(ii) μ is a membrane structure consisting of m membranes, with the membranes and the regions labeled in a one-to-one manner with $1, 2, \ldots, m$; the skin membrane is labeled with 1;

(iii) $L_1, \ldots, L_m$, are multisets of strings over V associated with the regions $1, 2, \ldots, m$ of μ;

(iv) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* of the form $a \to v(tar)\langle e\rangle$, where $a \in V, v \in V^*$, tar is one of the target indications *here, out, in*, and e is an integer number specifying the *energy* associated with the rule;

(v) $d_1, \ldots, d_m$ are natural numbers indicating the *dissolving thresholds* of each membrane.

In each time unit and in each membrane of the system, each copy of a string which can be rewritten by a rule in that region should be rewritten; in each step, each string is rewritten only by one rule. After rewriting a copy of the string x by using a rule $a \to v(tar)\langle e\rangle$, that copy of x is no longer present in the system, while the string resulting from rewriting is sent to the membrane indicated by tar, in the usual manner. Moreover, we have to take into account the energy restriction, in the same way as in the previous sections: the total energy of rules used in a membrane, for rewriting the strings present in that membrane, should be positive. The energy is passed from a step to the next one. If the energy overpasses the dissolving threshold (more precisely, if it is greater than or equal to the corresponding d_i), then the membrane is dissolved, all strings remain free in the upper membrane, the energy and the local rules are lost. The skin membrane is never dissolved.

A computation is correctly finished only if it halts, a configuration is obtained where no rule can be applied to any string. The result of a halting computation consists of all strings which are sent out of the system during the computation. In this way, a P system Π generates a language $L(\Pi)$, consisting of all strings which are produced as described above by all possible halting computations.

We denote by $REP_m(\varepsilon)$ the family of languages generated by P systems with at most m membranes, $m \geq 1$, and with the maximal absolute value of energy associated with rules at most ε. When m or ε can be arbitrarily large, then we replace it by $*$.

As we have announced above, a result similar to Theorem 3 holds true also for the case of string-objects.

Theorem 4. $RE = REP_*(2)$.

Proof. Let $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, $N = N_1 \cup N_2 \cup \{S, \#\}$. Assume that $N_2 = \{A_1, \ldots, A_k\}$. Add

to N_1 the new symbols $Z_1, \ldots, Z_k$ and to T the new symbol f. Replace each matrix $(X \to \lambda,\ A \to x)$ by a matrix $(X \to Z_1, A \to x)$ and then add also the matrices $(Z_j \to Z_{j+1},\ A_j \to \#)$, for $j = 1, 2, \ldots, k-1$, as well as the matrix $(Z_k \to f, A_k \to \#)$; all the rules $A_j \to \#$ are to be used in the appearance checking manner. Let us denote by $G' = (N', T', S, M', F')$ the grammar obtained in this way (the significance of the components is clear from the previous discussion; we only note that $N' = N_1' \cup N_2 \cup \{S, \#\}$, for $N_1' = N_1 \cup \{Z_1, \ldots, Z_k\}$). It is easy to see that $L(G') = \{f\}L(G)$ and that each derivation in G' ends with a phase when, in the presence of symbols Z_j, one checks whether or not any nonterminal symbol from N_2 is still present in the sentential form. In the last step, one uses the matrix $(Z_k \to f, A_k \to \#)$.

Assume that in the grammar G' there are s matrices $m_i : (X \to Y, A \to x), 1 \leq i \leq s$, and t matrices $m_{s+i} : (X \to \alpha, A \to \#), 1 \leq j \leq t$, with $\alpha \in N_1' \cup \{f\}$.

We construct the P system (of degree $s + t + 2$)

$$\Pi = (V, \mu, L_1, \ldots, L_{s+t+2}, R_1, \ldots, R_{s+t+2}, 2, \ldots, 2),$$

with

$$\begin{aligned}
&V = N_1' \cup N_2 \cup T \cup \{d, e, e', f\} \cup \{X_i \mid X \in N_1', 1 \leq i \leq s+t\},\\
&\mu = [_1[_2[_3\,]_3 \cdots [_{s+2}\,]_{s+2}[_{s+3}\,]_{s+3} \cdots [_{s+t+2}\,]_{s+t+2}]_2]_1,\\
&L_1 = \emptyset,\\
&L_2 = \{(XA, 1)\},\ \text{for } (S \to XA) \text{ the initial matrix of } G,\\
&L_{2+i} = \{(d, 1)\}, 1 \leq i \leq 2,\\
&L_{s+i+2} = \{(de, 1)\}, 1 \leq i \leq t,
\end{aligned}$$

and with the following sets of rules

$$\begin{aligned}
R_1 &= \{f \to \lambda(out)\langle 0\rangle\},\\
R_2 &= \{X \to \alpha_i(in)\langle 0\rangle \mid \text{for } m_i : (X \to \alpha, A \to \beta), 1 \leq i \leq s+t\}\\
&\cup\ \{d \to d\langle 0\rangle,\ f \to f\langle 2\rangle\},\\
R_{2+i} &= \{Y_i \to Y\langle 0\rangle,\ Y \to Y(out)\langle -1\rangle,\ A \to x\langle 1\rangle\}\\
&\cup\ \{Y_j \to d\langle 2\rangle \mid 1 \leq j \leq s+t, j \neq i\},\\
&\quad \text{for all } m_i : (X \to Y, A \to x), 1 \leq i \leq s,\\
R_{2+s+i} &= \{Y_i \to \alpha(out)\langle -1\rangle,\ A \to d\langle 2\rangle\}\\
&\cup\ \{Y_j \to d\langle 2\rangle \mid 1 \leq j \leq s+t, j \neq i\}\\
&\cup\ \{e \to e'\langle 0\rangle,\ e' \to e\langle 1\rangle,\ e' \to e\langle 0\rangle\},\\
&\quad \text{for all } m_{s+i} : (X \to \alpha, A \to \#), 1 \leq i \leq t, \alpha \in N_1' \cup \{f\}.
\end{aligned}$$

We start with the string XA in membrane 2, one copy of d in each membrane $2 + i, 1 \leq i \leq s$, and one copy of de in each of the membranes $s + 3, \ldots, s + t + 2$. In each membrane $s + 2 + i, i \geq 1$, the strings de can be rewritten forever by the rules $e \to e'\langle 0\rangle, e' \to e\langle 0\rangle$; if we use twice the rules $e \to e'\langle 0\rangle, e' \to e\langle 1\rangle$, then the membrane is dissolved. In membrane 2, the string de can also be rewritten forever, by the rule

$d \to d\langle 0\rangle$. Therefore, the only way to halt a computation is to first introduce the symbol f and then to dissolve membrane 2 by using the rule $f \to f\langle 2\rangle$. This means that a terminal derivation in G is simulated.

Indeed, let us consider that in membrane 2 we have a string Xw; initially, $w = A$, for $(S \to XA)$ being the initial matrix of G. If no rule $X \to \alpha$ can be used, then the computation continues forever as we have seen above.

Assume that a rule $X \to Y_i(in)\langle 0\rangle$ is used, for some $1 \leq i \leq s$. If the symbol Y_i arrives in a membrane j with $3 \leq j \leq s+2, j \neq 2+i$, then the rule $Y_i \to d\langle 2\rangle$ is eventually used, because the string does not leaves the membrane. In this way, the membrane is dissolved, the string d remains free in membrane 2 and the computation will never finish: the symbol Y_i will remain unchanged, hence f is never introduced. If the symbol Y_i arrives in a membrane j with $j \geq s+2$, then the result is the same.

If Y_i arrives in the right membrane $2+i$, then the only way to send it out is to use the rules $A \to x\langle 1\rangle$ once (if we use it twice, the membrane is dissolved and the computation will continues forever) and $Y_i \to Y\langle 0\rangle$, in any order, and then the rule $Y \to Y(out)\langle -1\rangle$. The energy is consumed, the string is sent out in the form corresponding to the correct use of the matrix $m_i : (X \to Y, A \to x)$.

Note that the simulation of a matrix as above takes four steps.

Asume now that we use a rule $X \to \alpha_i(in)\langle 0\rangle$, for some $s+1 \leq i \leq s+t$. If the symbol Y_i arrives in a membrane j with $3 \leq j \leq s+2$, or with $s+3 \leq j \leq s+t$, but with $j \neq i$, then the computation will never stop – the argument is the same as in the previous discussion. If the symbol α_i arrives in the right membrane $2+s+i$, then this is done exactly at the same step when the rule $e \to e'\langle 0\rangle$ is used in the same membrane $2+s+i$. No energy is available, hence the rule $\alpha_i \to \alpha(out)\langle -1\rangle$ cannot be used. If A is present in the string, then the rule $A \to d\langle 2\rangle$ must be used, the membrane is dissolved, and the string de arrives in membrane 2. The computation will never stop, because the rule $d \to d\langle 0\rangle$ can be used forever, while f will never be introduced (α_i remains in this form in membrane 2). If, after introducing α_i in membrane $2+s+i$, we use the rule $e' \to e\langle 0\rangle$, then again no energy is available and we have to wait until the rule $e' \to e\langle 1\rangle$ is used. This happens at an even step and, at the same step, because we work on different strings, we can also use the rule $\alpha_i \to \alpha(out)\langle -1\rangle$. The total energy returns to zero, the string exits and returns to membrane 2 in the correct form as after using the matrix $m_i : (X \to \alpha, A \to \#)$.

Note the important detail that the simulation of a matrix of G used in the appearance checking mode needs an even number of steps; also an even number of steps (four) was necessary for simulating matrices which are not used in the appearance checking mode. In this way, the synchronization with the use of the energy providing rules $e \to e'\langle 0\rangle, e' \to e\langle 1\rangle$ is ensured.

Therefore, the derivation in G can be correctly simulated in Π and each computation in Π corresponds to a correct derivation in G. In the moment when the symbol f is introduced, we know that the derivation in G was a terminal one (the non-occurrence of all symbols $A_1, \ldots, A_k$ is ensured by the passing through the symbols $Z_1, \ldots, Z_k$). The object f entails the dissolvation of membrane 2, hence no further rule is available here. The string is sent to membrane 1, f is removed and the string leaves the system. In order to finish the computation we have to also dissolve all membranes $s+3, \ldots, s+t+2$

and this can be done by using twice the rules $e \to e'\langle 0\rangle, e' \to e\langle 1\rangle$. The string de will arrive in membrane 1 (membrane 2 was already dissolved), but no rule can be applied to it here, the computation halts.

In conclusion, we get $L(\Pi) = L(G)$, which completes the proof. □

The number of membranes used in the previous proof depends on the number of matrices in the grammar G' (hence both on the number of matrices and of nonterminal symbols in the initial grammar G). We do not know whether or not the number of membranes can be bounded, that is, whether or not a result of the form $RE = REP_m(\varepsilon)$ or, maybe, $RE = REP_m(*)$ is true, for some $m \geq 1$. We have only a related result of this type: the matrix languages generated without appearance checking can be obtained in this way.

Theorem 5. $MAT \subset REP_3(*)$.

Proof. Let $G = (N, T, S, M, \emptyset)$ be a matrix grammar without appearance checking in the binary normal form, $N = N_1 \cup N_2 \cup \{S\}$. As in the previous proofs, we replace the matrices $(X \to \lambda, A \to x)$ with $(X \to f, A \to x)$, for a new symbol f. We denote by G' the obtaind grammar; assume that its matrices are $m_i : (X \to \alpha, A \to x), 1 \leq i \leq s$. We construct the P system

$$\Pi = (V, [_1[_2[_3\ ]_3]_2]_1, L_1, L_2, L_3, R_1, R_2, R_3, 2, 1, 1),$$

with

$$\begin{aligned}
V &= N_1 \cup N_2 \cup \{C, C', d, f\} \cup \{C_i \mid 1 \leq i \leq s\},\\
L_1 &= \emptyset,\\
L_2 &= \{(XA, 1), (C, 1)\}, \text{ for } (S \to XA) \text{ being the initial matrix of } G,\\
L_3 &= \{(d, 1)\},\\
R_1 &= \{C_i \to C_i\langle 0\rangle \mid 1 \leq i \leq s\}\\
&\cup\ \{f \to \lambda(out)\langle -2\rangle,\ C \to C'\langle 1\rangle,\ C' \to \lambda\langle 1\rangle\}\\
&\cup\ \{A \to A(in)\langle 0\rangle \mid A \in N_2\},\\
R_2 &= \{X \to \alpha(in)\langle -i\rangle,\ C \to C_i(in)\langle i\rangle \mid 1 \leq i \leq s, m_i : (X \to \alpha, A \to x)\}\\
&\cup\ \{C \to C(out)\langle 1\rangle,\ f \to f(out)\langle -1\rangle\}\\
&\cup\ \{\alpha \to \alpha\langle 0\rangle \mid \alpha \in N_1 \cup N_2 \cup \{d\}\},\\
R_3 &= \{A \to x(out)\langle -3i\rangle,\ C_i \to C(out)\langle 3i\rangle \mid 1 \leq i \leq s, m_i : (X \to \alpha, A \to x)\}.
\end{aligned}$$

This system works as follows. Simultaneously, in membrane 1 we have two strings, one which corresponds to a sentential form of the grammar G', and C, both of them in one copy each. The string C can be rewritten by a rule $C \to C_i(in)\langle 3i\rangle$ if and only if at the same time we also use a rule $X \to \alpha(in)\langle -3j\rangle$, for some $1 \leq i, j \leq s + t$ such that $i = j$: indeed, on the one hand, the total energy must be positive, hence $i \geq j$, but it cannot be different from zero, because otherwise membrane 2 will be dissolved and the string C_i will arrive in membrane 1 where it will evolve forever. Thus, we must have $j \geq i$, that is, the subscript of C_i identifies the matrix of G' whose first rule has been already simulated. The two strings arrive at the same time in membrane 3.

The rule $C_i \to C(out)\langle i\rangle$ can be used in membrane 3. If no other rule is used here, then this membrane is dissolved and the string d is left free in membrane 2, where it will evolve forever. If a rule $B \to y(out)\langle -j\rangle$ with $j < i$ is used, then again membrane 3 is dissolved and the computation cannot finish. Because we need $j \leq i$, it follows that we must have $i = j$, that is the matrix m_i is correctly simulated.

The process is iterated. Note that we cannot use the rule $C \to C(out)\langle 1\rangle$ before having introduced the object f: no rule $X \to \alpha(in)\langle -3i\rangle$ can be used at the same time, hence we have to dissolve membrane 2. When also f is present, the only way to continue without dissolving membrane 2 is by using simultaneously the rules $f \to f(out)\langle -1\rangle, C \to C(out)\langle 1\rangle$. Both strings arrive in membrane 1.

Here, the rule $f \to \lambda(out)\langle -2\rangle$ cannot be used, hence if any nonterminal $A \in N_2$ is present in the string which corresponds to a sentential form of G', then the rule $A \to A(in)\langle 0\rangle$ must be used, the string is sent back to membrane 2 and it will evolve here forever. If no symbol A is present, then at the first step we use only the rule $C \to C'\langle 1\rangle$ (this is allowed, because the dissolving threshold of membrane 1 is 2), then we simultaneously use the rules $C' \to \lambda\langle 1\rangle$ and $f \to \lambda(out)\langle -2\rangle$ (we have accumulated the two energy quanta we need). The string is sent out.

Therefore, we can stop the computation and send out a string only if this string is terminal with respect to G. Consequently, $L(G) = L(\Pi)$, hence $MAT \subseteq REP_3(*)$.

This is a strict inclusion. In order to prove this assertion, let us consider the following P system:

$$
\begin{aligned}
\Pi &= (V, [_1[_2[_3\,]_3]_2]_1, L_1, L_2, L_3, R_1, R_2, R_3, 4, 1, 1),\\
V &= \{X, X', Y, Y', Z, Z', A, B, C, C_1, C_1', C_2, C_2', C_3, C_3', C_4, C_4', C_5, C_5', d, f\},\\
L_1 &= \emptyset,\\
L_2 &= \{(XA, 1), (C, 1)\},\\
L_3 &= \{(d, 1)\},\\
R_1 &= \{C_1 \to C_1\langle 0\rangle,\ C_3 \to C_3\langle 0\rangle,\ C_5 \to C_5\langle 0\rangle,\\
&\quad Y' \to Y(in)\langle -2\rangle,\ A \to A(in)\langle 2\rangle,\\
&\quad C_2 \to C_2'\langle 1\rangle,\ C_2' \to C(in)\langle 1\rangle,\\
&\quad X' \to X(in)\langle -6\rangle,\ B \to B(in)\langle -3\rangle,\\
&\quad C_4 \to C_4'\langle 3\rangle,\ C_4' \to C(in)\langle 3\rangle,\\
&\quad Z' \to Z(in)\langle -6\rangle,\ f \to \lambda(out)\langle -2\rangle,\\
&\quad C \to C'\langle 1\rangle,\ C' \to \lambda\langle 1\rangle\},\\
R_2 &= \{X \to X(in)\langle -3\rangle,\ C \to C_1(in)\langle 3\rangle,\\
&\quad X \to Y(in)\langle -9\rangle,\ C \to C_3(in)\langle 9\rangle,\\
&\quad Z \to Z(in)\langle -15\rangle,\ C \to C_5(in)\langle 15\rangle,\\
&\quad X \to Y'(out)\langle -6\rangle,\ C \to C_2(out)\langle 6\rangle,\\
&\quad Y \to X'(out)\langle -12\rangle,\ C \to C_4(out)\langle 12\rangle,\\
&\quad Y \to Z'(out)\langle -12\rangle,\\
&\quad Z \to f(out)\langle -1\rangle,\ C \to C(out)\langle 1\rangle,
\end{aligned}
$$

$$
\begin{aligned}
& f \to f\langle 0\rangle,\ X' \to X'\langle 0\rangle, \\
& Y' \to Y'\langle 0\rangle,\ Z' \to Z'\langle 0\rangle,\ d \to d\langle 0\rangle\}, \\
R_3 \ = \ & \{A \to BB(out)\langle -3\rangle,\ C_1 \to C(out)\langle 3\rangle, \\
& B \to A(out)\langle -9\rangle,\ C_3 \to C(out)\langle 9\rangle, \\
& A \to a(out)\langle -15\rangle,\ C_5 \to C(out)\langle 15\rangle\}.
\end{aligned}
$$

This system generates the language $L(\Pi) = \{a^{2^n} \mid n \geq 1\}$, and this can be seen as follows. In the presence of X, in membrane 3, each occurrence of A is replaced by two copies of B. X can be changed to Y only when no copy of A is present. This is checked in the following way. At the same time, C_2 and Y' are introduced and the strings are sent to membrane 1. Neither the rule $Y' \to Y(in)\langle -2\rangle$, nor the rule $B \to B(in)\langle -3\rangle$ can be used. If A is present, then the string which contains Y' is sent back to membrane 2, where it will evolve forever by using the rule $Y' \to Y'\langle 0\rangle$. The string is returned to membrane 2, with Y' replaced by Y, after using the rules $C_2 \to C_2'\langle 1\rangle, C_2' \to C(in)\langle 1\rangle$, $Y' \to Y(in)\langle -2\rangle$.

In the presence of Y, all copies of B are replaced by A, then Y is replaced by X when no copy of B is present. In order to check this, the strings are sent to membrane 1, by the rules $Y \to X'(out)\langle -12\rangle, C \to C_4(out)\langle 12\rangle$. If we apply the rule $A \to A(in)\langle 2\rangle$, then the rule $C_4 \to C_4\langle 3\rangle$ cannot be applied (the total energy is larger than that admitted by the skin membrane); after using $A \to A(in)\langle 2\rangle$ the computation will never stop, because of the presence of X', which can be rewritten forever in membrane 2. However, if we apply the rule $C_4 \to C_4\langle 3\rangle$, then we cannot apply the rule $A \to A(in)\langle 2\rangle$. In both cases, the rule $X' \to X(in)\langle -6\rangle$ cannot be applied. If any B is present, then the rule $B \to B(in)\langle -3\rangle$ can be used together with $C_4 \to C_4'\langle 3\rangle$ and again the computation will never stop. Thus, we can correctly continue with a computation which will end if and only if no B is present and we use first the rule $C_4 \to C_4'\langle 3\rangle$ and then, simultaneously, the rules $C_4' \to C(in)\langle 3\rangle, X' \to X(in)\langle -6\rangle$. Observe that this is allowed by the energy constraints.

The symbol Y can also be replaced by Z when no B is present, and this is done exactly as above. In the presence of Z, each copy of A is replaced by a. Eventually, the symbol f is introduced and the computation stops in the same way as the system Π from the first part of the proof does.

The reader can check all the details of the way the previous system works. Of a significant help can be the observation that Π corresponds to the matrix grammar

$$G = (\{S, A, B, X, Y, Z, \#\}, \{a\}, S, M, F),$$

with

$$
\begin{aligned}
M \ = \ & \{(S \to XA),\ (X \to X, A \to BB),\ (X \to Y, A \to \#), \\
& (Y \to Y, B \to A),\ (Y \to X, B \to \#),\ (Y \to Z, B \to \#), \\
& (Z \to Z, A \to a),\ (Z \to \lambda, A \to a)\},
\end{aligned}
$$

and with F containing the three occurrences of the rules $A \to \#, B \to \#$.

We close this proof by observing that the obtained language is not in the family MAT, [5]. □

5 Final Remarks

We have here considered P systems both with symbol-objects and string-objects with quantities of energy associated with each evolution rule, in the form of an integer number. In each step, only combinations of rules with a positive total of involved energies can be used. In this way, the membrane dissolvation can be defined in a natural manner, as occurring when the total amount of energy accumulated in a membrane overpasses a given threshold. This new feature added to P systems seems to be both biochemically well motivated and powerful from a computational point of view: characterizations of recursively enumerable languages are obtained in the string-object case and of recursively enumerable sets of vectors of natural numbers in the case of symbol-objects, providing that a priority relation is used (although known, this last result is proved in a much easier way in the new framework).

We have not considered here P systems where also the possibility of creating membranes is provided, in the sense of the paper [6], or as in [19], [20], depending on the objects present in the membrane. This is an interesting case, which remains as a research topic for further investigations.

Actually, many other problems seem to be of interest in this area. For instance, taking into account that the living systems act in such a way to preserve their state as secure as possible, we can consider that rules which are involving less energy should be promoted. In this way, we can define a priority relation among rules on the basis of the associated energy: less energy, in absolute value, means a higher priority. Which is the effect of this assumption on the behavior of P systems remains to be found. A related idea is to start a computation with a given amount of energy provided in the initial configuration and to stop the computation when no energy is available in the system (that is, the system works as long as the available energy is strictly positive in at least one membrane of it). Then, we may also consider non-integer quantities of energy associated with rules, even non-rational. It is expected that in such a case, the "arithmetics of energies" has a strong influence on the power of systems.

References

[1] C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000 (Chapter 3: "Computing with Membranes").

[2] J. Castellanos, A. Rodriguez-Paton, Gh. Păun, Computing with membranes: P systems with worm-objects, *IEEE Conf. SPIRE 2000*, La Coruna, Spain, 2000, and *Auckland University, CDMTCS Report* No 123, 2000 (www.cs.auckland.ac.nz/CDMTCS).

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

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

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

[6] M. Ito, C. Martin-Vide, Gh. Păun, A characterization of Parikh sets of ET0L languages in terms of P systems, submitted, 2000.

[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] S. N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, *International J. of Computer Math.*, in press.

[9] C. Martin-Vide, V. Mitrana, P systems with valuations, *Proc. Conf. UMC 2000*, Brusells, 2000.

[10] C. Martin-Vide, Gh. Păun, String-objects in P systems, *Proc. of Algebraic Systems, Formal Languages and Computations Workshop*, Kyoto, 2000, RIMS Kokyuroku, Kyoto Univ., 2000.

[11] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).

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

[13] Gh. Păun, Computing with membranes – A variant: P systems with polarized membranes, *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*, 5 (2000), in press, and *Auckland University, CDMTCS Report* No 102, 1999 (www.cs.auckland.ac.nz/CDMTCS).

[15] Gh. Păun, Computing with membranes (P systems): Twenty six research topics, *Auckland University, CDMTCS Report* No 119, 2000 (www.cs.auckland.ac.nz/CDMTCS).

[16] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 259–266,

[17] Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems, submitted, 2000.

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

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

[20] Y. Suzuki, H. Tanaka, Chemical evolution among artificial proto-cells, *Artificial Life VII*, MIT Press, 2000, to appear.

[21] Cl. Zandron, Cl. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes, *Proc. Conf. UMC 2000*, Brusells, 2000.

[22] Cl. Zandron, Cl. Ferretti, G. Mauri, Using membrane features in P systems, *Proc. Workshop on Multiset Processing*, Curtea de Argeş, Romania, 2000.