Sequential P Systems with Unit Rules and Energy Assigned to Membranes

Rudolf FREUND[1], Alberto LEPORATI[2], Marion OSWALD[1], Claudio ZANDRON[2]

[1]Faculty of Computer Science
Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
E-mail: {rudi,marion}@emcc.at

[2]Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano - Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
E-mail: {leporati,zandron}@disco.unimib.it

Abstract. We introduce a new variant of membrane systems where the rules are directly assigned to membranes (and not to the regions as this is usually observed in the area of membrane systems) and, moreover, every membrane carries an energy value that can be changed during a computation by objects passing through the membrane. For the application of rules leading from one configuration of the system to the succeeding configuration we consider a sequential model and do not use the model of maximal parallelism. The result of a successful computation is considered to be the distribution of energy values carried by the membranes. We will show that for such systems using a kind of priority relation on the rules we already obtain universal computational power. When omitting the priority relation, we obtain a characterization of the family of Parikh sets generated by context-free matrix grammars (with $\lambda-$rules).

1 Introduction

In 1998 Gheorghe Păun introduced membrane systems (in [11]) as distributed and parallel computing devices that were abstracted from the biological functioning of living cells. For motivations and examples as well as for further details we refer to [12]; for recent developments in the area of P systems see [15].

Considering the energy balancing of processes in a cell first was investigated in [13] and then in [4]. There the energies of all rules to be used in a given step in a membrane are summed up; if the total amount of energies is positive ([13]) or within a given range ([4]), then this multiset of rules can be applied if it is maximal with this property.

We here take another approach. In contrast to most models of P systems where the evolution rules are placed within a region, in this paper we consider membrane systems where the rules are directly assigned to the membranes (as already done in [6]) and have

to be applied in a sequential way (for sequential variants of P systems see, e.g., [2] and [3]). Moreover, each membrane carries an energy value. As long as the energy value of a membrane is positive, by a rule application, singleton objects can be rewritten while passing through membranes, thereby consuming or producing energy that is added to or subtracted from the energy value of the respective membrane. We also consider a kind of priority relation on the rules assigned to the membranes by choosing the one first that changes the energy value of the membrane under consideration in a maximal way. The result of a successful computation is stored in the final energy values of the membranes.

In the following section we first give some preliminary definitions and recall some notions and results for register machines and matrix grammars, the computation models we use for proving the results elaborated in this paper; in the third section we introduce P systems with unit rules and energy assigned to membranes followed by an example. In the fourth section we show that when using a kind of priority among the rules, the introduced systems can simulate register machines quite easily, which proves their universal computational power. A characterization of MAT^{λ} is obtained when omitting the priority relation. A short summary and an outlook to future research conclude the paper.

2 Preliminary Definitions

The set of non-negative integers is denoted by $\mathbf{N}$. An *alphabet* V is a finite non-empty set of abstract *symbols*. Given V, the free monoid generated by V under the operation of concatenation is denoted by V^*; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. A multiset over V is represented as string over V (and any of its permutations). By $|x|$ we denote the length of the word x over V as well as the number of elements in the multiset represented by x.

For more details on formal language theory we refer to [14].

2.1 Register Machines

When considering multisets of symbols, register machines provide a simple universal computational model (see [10] for some original definitions and, e.g., [5], [7] for definitions like those we use here).

An *n-register machine* is a construct $M = (n, P, i, h)$, where:

- n is the number of registers,
- P is a set of labelled instructions of the form $j : (op(r), k, l)$, where $op(r)$ is an operation on register r of M, j, k, l are labels from the set $Lab(M)$ (which numbers the instructions in a one-to-one manner),
- i is the initial label, and
- h is the final label.

The machine is capable of the following instructions:

$(A(r), k, l)$: Add one to the contents of register r and proceed to instruction k or to instruction l (in the deterministic variants usually considered in the literature we demand $k = l$).

$(S(r), k, l)$: If register r is not empty, then subtract one from its contents and go to instruction k, otherwise proceed to instruction l.

$Halt$: Stop the machine. This additional instruction can only be assigned to the final label h.

In their *deterministic variant*, such n-register machines can be used to compute any partial recursive function $f : \mathbf{N}^\alpha \to \mathbf{N}^\beta$; starting with $(n_1, ..., n_\alpha) \in \mathbf{N}^\alpha$ in registers 1 to α, M has computed $f(n_1, ..., n_\alpha) = (r_1, ..., r_\beta)$ if it halts in the final label h with registers 1 to β containing r_1 to r_β. If the final label cannot be reached, $f(n_1, ..., n_\alpha)$ remains undefined.

A deterministic n-register machine can also analyse an input $(n_1, ..., n_\alpha) \in \mathbf{N}^\alpha$ in registers 1 to α, which is recognized if the register machine finally stops by the halt instruction with all its registers being empty. If the machine does not halt, the analysis was not successful.

In their *non-deterministic variant*, n-register machines can compute any recursively enumerable set of non-negative integers (or of vectors of non-negative integers). Starting with all registers being empty, we consider a computation of the n-register machine to be successful, if it halts with the result being contained in the first (β) register(s) and with all other registers being empty.

The results proved in [5] (based on the results established in [10]) as well as in [7] and [8] immediately lead to the following results:

Proposition 1 *For any partial recursive function $f : \mathbf{N}^\alpha \to \mathbf{N}^\beta$ there exists a deterministic $(\max\{\alpha, \beta\} + 2)$-register machine M computing f in such a way that, when starting with $(n_1, ..., n_\alpha) \in \mathbf{N}^\alpha$ in registers 1 to α, M has computed $f(n_1, ..., n_\alpha) = (r_1, ..., r_\beta)$ if it halts in the final label h with registers 1 to β containing r_1 to r_β, and all other registers being empty; if the final label cannot be reached, $f(n_1, ..., n_\alpha)$ remains undefined.*

The following two corollaries are immediate consequences of the preceding proposition (by taking $\alpha = 0$ and $\beta = 0$, respectively):

Corollary 2 *For any recursively enumerable set $L \subseteq \mathbf{N}^\beta$ of vectors of non-negative integers there exists a non-deterministic $(\beta + 2)$-register machine M generating L in such a way that, when starting with all registers 1 to $\beta+2$ being empty, M non-deterministically computes and halts with n_i in registers i, $1 \leq i \leq \beta$, and registers $\beta + 1$ and $\beta + 2$ being empty if and only if $(n_1, ..., n_\beta) \in L$.*

Corollary 3 *For any recursively enumerable set $L \subseteq \mathbf{N}^\alpha$ of vectors of non-negative integers there exists a deterministic $(\alpha + 2)$-register machine M accepting L in such a way that M halts with all registers being empty if and only if M starts with some $(n_1, ..., n_\alpha) \in L$ in registers 1 to α and the registers $\alpha + 1$ to $\alpha + 2$ being empty.*

2.2 Matrix Grammars

A context-free *matrix grammar* (without appearance checking) is a construct

$$G = (N, T, S, M)$$

where N and T are sets of *non-terminal* and *terminal symbols,* respectively, with $N \cap T = \emptyset$, $S \in N$ is the *start symbol*, M is a finite set of *matrices*, $M = \{m_i \mid 1 \leq i \leq n\}$, where the matrices m_i are sequences of the form $m_i = (m_{i,1}, \ldots, m_{i,n_i})$, $n_i \geq 1$, $1 \leq i \leq n$, and the $m_{i,j}$, $1 \leq j \leq n_i$, $1 \leq i \leq n$, are context-free productions over (N, T).

For $m_i = (m_{i,1}, \ldots, m_{i,n_i})$ and $v, w \in (N \cup T)^*$ we define $v \Longrightarrow_{m_i} w$ if and only if there are $w_0, w_1, \ldots, w_{n_i} \in (N \cup T)^*$ such that $w_0 = v$, $w_{n_i} = w$, and for each $j, 1 \leq j \leq n_i$, w_j is the result of the application of $m_{i,j}$ to w_{j-1}.

The language generated by G is

$$L(G) = \{w \in T^* \mid \; S \Longrightarrow_{m_{i_1}} w_1 \ldots \Longrightarrow_{m_{i_k}} w_k,\ w_k = w, \\ w_j \in (N \cup T)^*,\ m_{i_j} \in M \ \text{ for } 1 \leq j \leq k, k \geq 1\} .$$

According to the definitions given in [1], the last matrix can already finish with a terminal word without having applied the whole sequence of productions.

The family of languages generated by matrix grammars without appearance checking is denoted by MAT^{λ}. It is known that $PsCF \subset PsMAT^{\lambda} \subset PsRE$. Further details about matrix grammars can be found in [1] and in [14].

3 P Systems with Unit Rules and Energy Assigned to Membranes

In this section we define the new model of membrane systems introduced in this paper.

A *P system with unit rules and energy assigned to membranes of degree* $d + 1$ is a construct Π of the following form:

$$\Pi = (O, \mu, e_0, ..., e_d, w_0, ..., w_d, R_0, ..., R_d) ,$$

where:

- O is an alphabet of *objects*;
- μ is a *membrane structure* (with the membranes labelled by numbers $0, ..., d$ in a one-to-one manner);
- $e_0, ..., e_d$ are the initial energy values assigned to the membranes $0, ..., d$;
- $w_0, ..., w_d$ are multisets over V associated with the regions $0, ..., d$ of μ;
- $R_0, ..., R_d$ are finite sets of *unit rules* associated with the membranes $0, ..., d$, which are of the form
$$(\alpha : a, \Delta e, b)$$
where $\alpha \in \{in, out\}$, $a, b \in O$, and $|\Delta e|$ is the amount of energy that - for $\Delta e \geq 0$ - is added to or - for $\Delta e < 0$ - is subtracted from e_i (the energy assigned to membrane i) by the application of the rule.

In a more depictive way, a rule
$$(in : a, \Delta e, b) ,$$
in R_i can be written in the following form:

$$a \to \overset{\Delta e}{\underset{i}{\mid}} b$$

A rule

$$(out : a, \Delta e, b)$$

in R_i can be written in the following form:

$$b \overset{\Delta e}{\underset{i}{\mid}} \leftarrow a.$$

Starting from the *initial configuration*, which consists of μ, $e_0, ..., e_d$, and $w_0, ..., w_d$, the system passes from one configuration to another one by non-deterministically choosing one rule from some R_i and applying it in the following sense (observe that here we consider a sequential model of applying the rules instead of choosing rules in a maximally parallel way as it is often required in P systems): applying $(in : a, \Delta e, b)$ means that an object a (being in the membrane immediately outside of i) is changed into b while entering membrane i thereby changing the energy value e_i of membrane i by Δe. On the other hand, the use of a rule $(out : a, \Delta e, b)$ changes object a into b while it passes out from membrane i changing its energy value by Δe. Yet the rules are only applicable if the amount e_i of energy assigned to membrane i fulfills the requirement $e_i + \Delta e \geq 0$; moreover, we use some sort of local priorities: if there is more than one applicable rule in membrane i, then one of the rules with max $|\Delta e|$ has to be used.

A sequence of transitions is called a *computation*; it is *successful* if and only if it halts. The *result* of a successful computation is considered to be the distribution of energies among the membranes (a non-halting computation does not produce a result). If we consider the energy distribution of the membrane structure as the input to be analysed, we obtain a model for accepting sets of (vectors of) non-negative integers.

Observe that in this model we do not take into account the environment.

To illustrate the definitions given above, we now elaborate an example of a simple comparator.

Example 4 *Consider the following system*

$$\Pi = (\{p_1, p_2\}, [_0[_1]_1[_2]_2]_0, 0, e_1, e_2, p_1, \emptyset, \emptyset, \emptyset, R_1, R_2)\,,$$

where:

$$\begin{aligned} R_1 &= \{(in : p_1, 0, p_1)\,, (out : p_1, -1, p_2)\}\,, \\ R_2 &= \{(in : p_2, 0, p_2)\,, (out : p_2, -1, p_1)\}\,. \end{aligned}$$

In a more depictive way, the rules of Π *can be described in the following way:*

$$p_1 \to \overset{0}{\underset{1}{\mid}} p_1 \qquad p_2 \overset{-1}{\underset{1}{\mid}} \leftarrow p_1 \qquad p_2 \to \overset{0}{\underset{2}{\mid}} p_2 \qquad p_1 \overset{-1}{\underset{2}{\mid}} \leftarrow p_2$$

We start by applying $(in : p_1, 0, p_1)$*, sending the object* p_1 *into membrane* 1 *neither changing the object itself nor the energy value of membrane* 1*. Using* $(out : p_1, -1, p_2)$ *in the next step,* p_1 *is changed into* p_2 *by passing out again, thereby decreasing* e_1 *by* 1*. From here,* p_2 *can cross membrane* 2 *by* $(in : p_2, 0, p_2)$*, from where it goes out again as* p_1*, having decreased* e_2 *by* 1 *(application of* $(out : p_2, -1, p_1)$*). As long as both energy values* e_1 *and*

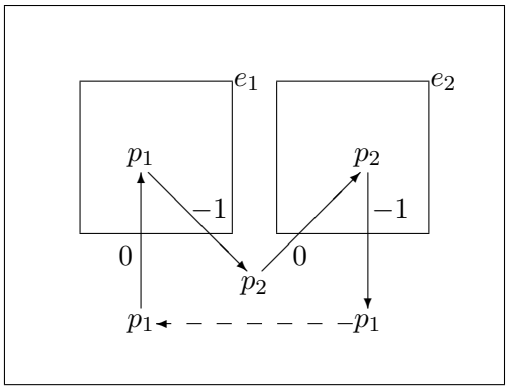

Figure 1: Example Comparator

e_2 are positive, this procedure goes on. One cycle of this procedure is illustrated in Figure 1.
Whenever one of the energy values becomes 0, *the system halts in one of two possible configurations (see figure 2):*

- $e_1 \leq e_2$: *The energy value of membrane* 1 *was initially greater than or equal to the energy value of membrane* 2. *Hence the system halts (after* $e_1 * 4 + 1$ *steps) when* p_1 *has just entered membrane* 1, *but cannot pass out again because the use of the rule* $(p_1, out, p_2, -1)$ *would require energy that is not available anymore. This situation corresponds to the illustration given on the left-hand side of Figure 2.*

- $e_1 > e_2$: *In case the energy value of membrane* 2 *in the initial configuration was greater than the one of membrane* 1, *the system now stops (after* $e_2 * 4 + 3$ *steps) because the energy of membrane* 2 *is already consumed. Hence,* p_2 *cannot leave membrane* 2 *anymore, which is illustrated on the right-hand side of Figure 2.*

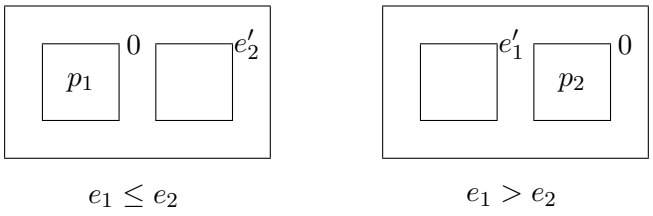

Figure 2: Possible end configurations for the comparator

4 Results

Theorem 5 *Each partial recursive function* $f : \mathbf{N}^\alpha \rightarrow \mathbf{N}^\beta$ *can be computed by a P system with unit rules and energy assigned to membranes with (at most)* $\max\{\alpha, \beta\} + 3$ *membranes.*

Proof. Consider a (deterministic) register machine $M = (m, P, i, h)$ with m registers, where $m = \max\{\alpha, \beta\} + 2$ (according to the result stated in Proposition 1). Now let P be a program which computes the function f such that the initial instruction has the label 1 and the halting instruction has the label n.

The input values $x_1, ..., x_\alpha$ are expected to be in the first α registers and the output values from $f(x_1, ..., x_\alpha)$ are expected to be in registers 1 to β at the end of a successful computation. Moreover, without loss of generality, we may assume that at the beginning of a computation all the registers except eventually the registers 1 to α contain zero.

We construct the P system

$$\Pi = (O, \mu, e_0, .., e_m, w_0, ..., w_m, R_0, ..., R_m),$$

where:

$$\begin{array}{lcll} O & = & \{p_j, \widetilde{p_j} | 1 \leq j \leq n, j \in Lab(M)\}, & \\ \mu & = & [_0[_1]_1 ... [_\alpha]_\alpha ... [_m]_m]_0, & \\ e_i & = & x_i, & \text{for } 1 \leq i \leq \alpha, \\ & & 0, & \text{for } \alpha + 1 \leq i \leq m, \\ w_0 & = & p_1, & \\ w_i & = & \lambda & \text{for } 1 \leq i \leq m, \\ R_i & = & \{(in : p_j, 1, \widetilde{p_j}), (out : \widetilde{p_j}, 0, p_k) | j : (A(i), k, k) \in P\} & \\ & \cup & \{(in : p_j, 0, \widetilde{p_j}), (out : \widetilde{p_j}, -1, p_k), (out : \widetilde{p_j}, 0, p_l) | & \\ & & j : (S(i), k, l) \in P\}, & \text{for } 1 \leq i \leq m. \end{array}$$

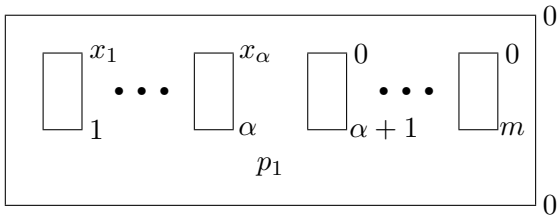

Figure 3: Initial configuration of Π

The contents of register i, $1 \leq i \leq m$, is represented by the energy value e_i of membrane i. The membrane structure of Π and the initial distribution of energies are illustrated in figure 3.

The sets of rules R_i depend on the instructions of P; in more detail, the simulation works as follows:

1. Each add-instruction $j : (A(i), k, k) \in P$, $1 \leq i \leq m$ is simulated in two steps by using the unit rules

$$p_j \to |^{1}_{i} \widetilde{p_j} \qquad \text{and} \qquad p_l |^{0}_{i} \leftarrow \widetilde{p_j}.$$

2. Each conditional subtract-instruction $j : (S(i), k, l) \in P$ is simulated in two steps

by the following unit rules:

$$p_j \to \overset{0}{|}_i \widetilde{p_j} \qquad \text{and}$$

$$p_k \overset{-1}{|}_i \leftarrow \widetilde{p_j} \qquad \text{or} \qquad p_l \overset{0}{|}_i \leftarrow \widetilde{p_j}$$

The condition of priority guarantees that $(out : \widetilde{p_j}, -1, p_k)$ is applied as long as e_i has a positive value. Only if in the current configuration $e_i = 0$, i.e., register i is empty, the rule $(out : \widetilde{p_j}, 0, p_l)$ can be used.

It follows from the description given above that after each simulation of an instruction each energy value e_i equals the contents of register i, $1 \leq i \leq m$. Hence, after having simulated the instruction $Halt$ and halting the system by just doing nothing with the halting symbol p_n anymore, the energy values $e_1, .., e_m$ equal the output of the program P. The only object remaining within the system is the final label p_n in region 0. □

The following corollaries are immediate consequences of Theorem 5 by taking $\beta = 0$ and $\alpha = 0$, respectively.

Corollary 6 *Let $L \subseteq \mathbf{N}^{\alpha}$, $\alpha \geq 1$, be a recursively enumerable set of (vectors of) non-negative integers. Then L can be accepted by a P system with unit rules and energy assigned to membranes with (at most) $\alpha + 3$ membranes.*

Proof (sketch). As the input is already contained in the system as the distribution of energy values assigned to the membranes 1 to α, we can immediately start the $(\alpha + 2)$ −register machine from Corollary 3. Hence, we define

$$\Pi_{acc} = (O, \mu, e_0, .., e_{\alpha+2}, w_0, ..., w_{\alpha+2}, R_0, ..., R_{\alpha+2}),$$

where:

$$\begin{array}{lcl}
O & = & \{p_j, \widetilde{p_j} | 1 \leq j \leq n, j \in Lab(M)\}, \\
\mu & = & [_0[_1]_1 ... [_\alpha]_\alpha [_{\alpha+1}]_{\alpha+1} [_{\alpha+2}]_{\alpha+2}]_0, \\
e_i & = & x_i, \qquad \text{for } 1 \leq i \leq \alpha, \\
 & & 0, \qquad \text{for } i \in \{\alpha + 1, \alpha + 2\}, \\
w_0 & = & p_1, \\
w_i & = & \lambda, \qquad \text{for } 1 \leq i \leq \alpha + 2, \\
R_i & = & \{(in : p_j, 1, \widetilde{p_j}), (out : \widetilde{p_j}, 0, p_k) | j : (A(i), k, k) \in P\} \\
 & \cup & \{(in : p_j, 0, \widetilde{p_j}), (out : \widetilde{p_j}, -1, p_k), (out : \widetilde{p_j}, 0, p_l) | \\
 & & j : (S(i), k, l) \in P\}, \qquad \text{for } 1 \leq i \leq m.
\end{array}$$

The membrane structure of Π_{acc} together with the initial energy values of the membranes can also be seen in figure 4.

Thus the main emphasis again lies on the simulation of a register machine, which is done in the same way as in the proof of Theorem 5. As the rules are applied in a sequential manner, in any moment of time, there is only one symbol present in the system. □

Corollary 7 *Let $L \subseteq \mathbf{N}^{\beta}$ be a recursively enumerable set of (vectors of) non-negative integers. Then L can be generated by a P system with unit rules and energy assigned to membranes with (at most) $\beta + 3$ membranes.*

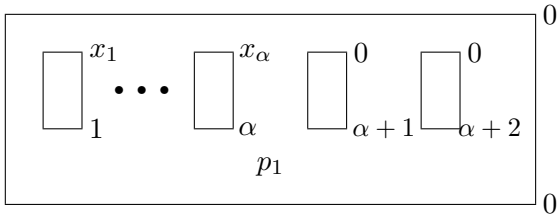

Figure 4: Initial configuration of Π_{acc}

Proof (sketch). In the same way as in the proof of Theorem 5 the P system Π was constructed in order to simulate the (deterministic) register machine from Proposition 1, we now construct a P system Π' which simulates the non-deterministic register machine from Corollary 2 and in that way non-deterministically generates a representation of any vector from the given language L by the corresponding energy values e_1 to e_β. Hence, let us define

$$\Pi_{gen} = (O, \mu, e_0, .., e_{\beta+2}, w_0, ..., w_{\beta+2}, R_0, ..., R_{\beta+2}),$$

$$\begin{array}{rrl}
 & O & = \{p_j, \widetilde{p_j} | 1 \leq j \leq n, j \in Lab(M)\}, \\
 & \mu & = [_0\ [_1\ ]_1 ... [_\beta\ ]_\beta [_{\beta+1}\ ]_{\beta+1} [_{\beta+2}\ ]_{\beta+2}\ ]_0, \\
 & e_i & = 0, \qquad \text{for } 0 \leq i \leq \beta+2, \\
 & w_0 & = p_1, \\
\text{where:} & w_i & = \lambda, \qquad \text{for } 1 \leq i \leq \beta+2, \\
 & R_i & = \{(in : p_j, 1, \widetilde{p_j}), (out : \widetilde{p_j}, 0, p_k), (out : \widetilde{p_j}, 0, p_l) | \\
 & & \quad j : (A(i), k, l) \in P\} \\
 & \cup & \{(in : p_j, 0, \widetilde{p_j}), (out : \widetilde{p_j}, -1, p_k), (out : \widetilde{p_j}, 0, p_l) | \\
 & & \quad j : (S(i), k, l) \in P\}, \qquad \text{for } 1 \leq i \leq \beta+2.
\end{array}$$

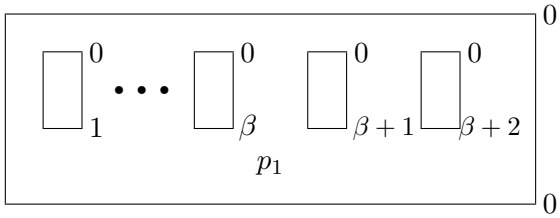

Figure 5: Initial configuration of Π_{gen}

The membrane structure of Π_{gen} is depicted in figure 5. $R_0, ..., R_{\beta+2}$ are constructed in a similar way as in the proof of Theorem 5, except that now in the non-deterministic case we also have to consider add-instructions of the form $j : (A(i), k, l)$ with $k \neq l$. □

On the other hand, when omitting the priority feature, we do not get systems with universal computational power. In the following, $PsPE_*(unit)$ denotes the family of Parikh vectors generated by P systems with unit rules and energy assigned to membranes without priorities and with an arbitrary number of membranes. The following two lemmas prove that

$$PsPE_*(unit) = PsMAT^\lambda,$$

i.e., we get a characterization of $PsMAT^{\lambda}$ by the new family $PsPE_*(unit)$ introduced in this paper.

Lemma 8 $PsPE_*(unit) \supseteq PsMAT^{\lambda}$.

Proof. Let $G = (N, T, S, M)$ be a matrix grammar with $\lambda-$rules with every matrix being of the form $m_i = (m_{i,1}, \ldots, m_{i,n_i})$, $1 \leq i \leq n$, where $m_{i,j} = A_{i,j} \rightarrow w_{i,j,1}....w_{i,j,n_{i,j}}$. Without loss of generality, we may assume that $n_{i,j} \leq 2$. Then we can construct a P system Π simulating G in the following way:

We label the skin membrane by 0 and for all elements B_i in $N \cup T$ we take a membrane labelled by i, $1 \leq i \leq m$, where $m = card(N \cup T)$ and $m' = card(T)$; moreover, we define a bijective function $index : \{1, ..., m\} \rightarrow N \cup T$ such that the terminal symbols have the indices 1 to m' and the start symbol S has the label m. Initially every membrane has the energy value 0, i.e., $e_j = 0$ for $0 \leq j \leq m$. The initial configuration of Π can also be seen in figure 6.

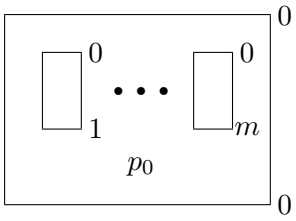

Figure 6: Initial configuration of Π

Before starting the simulation of the matrices, we first have to add an additional step in order to get $e_m = 1$ as well as to have a non-deterministic choice for m_i by taking the unit rules

$$p_0 \rightarrow \overset{1}{\underset{m}{|}} \widetilde{p_0} \qquad \text{as well as} \qquad p_{i,1,0} \overset{0}{\underset{m}{|}} \leftarrow \widetilde{p_0}$$

for every $1 \leq i \leq$ m.

For the simulation of $m_{i,j}$, $1 \leq j \leq n_{i,j}$, $1 \leq i \leq n$, we have to take the following rules:

1. $p_{i,j,0} \rightarrow \overset{0}{\underset{index(A_{i,j})}{|}} \widetilde{p_{i,j,0}}$ and $\alpha_{i,j-1} \overset{-1}{\underset{index(A_{i,j})}{|}} \leftarrow \widetilde{p_{i,j,0}}$

   where

   - $\alpha_{i,j} \in \{p_{k,1,0} | 1 \leq k \leq n\}$ for $w_{i,j} = \lambda$ and $j = n$;
   - $\alpha_{i,j} = p_{i,j,1}$ otherwise.

2. $p_{i,j,1} \rightarrow \overset{1}{\underset{index(w_{i,j,1})}{|}} \widetilde{p_{i,j,1}}$ and $\beta_{i,j} \overset{0}{\underset{index(w_{i,j,1})}{|}} \leftarrow \widetilde{p_{i,j,1}}$

   where

   - $\beta_{i,j} \in \{p_{k,1,0} | 1 \leq k \leq n\}$ for $|w_{i,j}| = 1$ and $j = n_i$,
   - $\beta_{i,j} = p_{i,j+1,0}$ for $|w_{i,j}| = 1$ and $j < n_i$,

- $\beta_{i,j} = p_{i,j,2}$ for $|w_{i,j}| = 2$.

3. $p_{i,j,2} \rightarrow \overset{1}{\underset{index(w_{i,j,2})}{|}} \widetilde{p_{i,j,2}}$ and $\gamma_{i,j} \overset{0}{\underset{index(w_{i,j,2})}{|}} \leftarrow \widetilde{p_{i,j,2}}$

   where

   - $\gamma_{i,j} \in \{p_{k,1,0} | 1 \leq k \leq n\}$ for $j = n_i$,
   - $\gamma_{i,j} = p_{i,j+1,0}$ for $j < n_i$.

4. $p_f \overset{0}{\underset{index(A_{i,j})}{|}} \leftarrow \widetilde{p_{i,j,0}}$

   is a kind of "emergency exit" which allows us to finish whenever the current sentential form is already terminal.

5. To check whether the current sentential form is already terminal (i.e., $e_i = 0$ for $m' + 1 \leq j \leq m$) we take the rules

   $p_f \rightarrow \overset{-1}{\underset{j}{|}} \widetilde{p_f}$ and $\#\overset{0}{\underset{j}{|}} \leftarrow \widetilde{p_f}$ for $m' + 1 \leq j \leq m$.

6. Finally, in case the "emergency exit" was taken too early, we have to make sure that the system does not halt by adding an infinite loop with the trap symbol # :

   $\# \rightarrow \overset{0}{\underset{m}{|}} \#$ and $\#\overset{0}{\underset{m}{|}} \leftarrow \#$

If p_f cannot enter any of the membranes $m' + 1 \leq j \leq m$ this means that no non-terminal symbol occurs any more in the current sentential form of the simulated derivation in G, hence, it is correct to halt and thus get the result stored in the values of e_j, $1 \leq j \leq m$, which by construction represents the corresponding result obtained by the simulated derivation in G. □

Lemma 9 $PsPE_*(unit) \subseteq PsMAT^{\lambda}$.

Proof. We first construct a matrix grammar which generates a suitable representation of all configurations reachable from the initial configuration in Π. Eliminating all non-final configurations from this set of reachable configurations by intersection with regular languages we obtain the set of halting configurations which immediately allows us to extract the terminal results by using a projection. As the family of matrix languages is closed under intersection with regular languages and projections (see [1]) this will prove the desired inclusion $PsPE_*(unit) \subseteq PsMAT^{\lambda}$.

We first start the construction of a matrix grammar G generating the reachable configurations in Π :

Let

$$\Pi = (O, \mu, e_0, ..., e_d, w_0, ..., w_d, R_0, ..., R_d)$$

be an arbitrary P system with unit rules and energy assigned to membranes (arbitrary membrane structure, arbitrary number of membranes, arbitrary number of symbols); then the matrix grammar $G = (V, T, M, S)$ is constructed in the following way:

Taking $D = \{0, 1, ..., d\}$, we first define the mapping σ from the set of all possible configurations of Π to

$$(O \times D)^* \{D_0\} \{E_0\}^* ... \{D_d\} \{E_d\}^*$$

which for every configuration c of Π yields all its valid representations in such a way that:

- for every a in region i the symbol $(a, i) \in (O \times D)$ occurs in the string representation of c;
- the number of symbols (a, i) occurring in the string representation of c exactly coincides with the number of symbols a occurring in region i;
- the second part of the string representation of c is of the form

  $$D_0 E_0^{e_0} ... D_d E_d^{e_d}$$

  such that e_i is the energy value assigned to membrane i in configuration c, $0 \leq i \leq d$.

In G, we start with an initial matrix

$$[S \to s]$$

such that $s \in \sigma(initial\ configuration)$, i.e., is a valid string representation of the initial configuration in the form defined above.

The unit rules in $R_0, ..., R_d$ are simulated in the following way:

- For a rule $(in : a, \Delta e, b)$ assigned to membrane i with $\Delta e \geq 0$ we take the matrix

  $$\left[(a, j) \to (b, i), D_i \to D_i E_i^{\Delta e}\right],$$

  where j is the label of the membrane encapsulating membrane i.
- For a rule $(in : a, \Delta e, b)$ assigned to membrane i with $\Delta e < 0$ we take the matrix

  $$\left[(a, j) \to \widetilde{(b, i)}, (E_i \to \lambda,)^{-\Delta e} \widetilde{(b, i)} \to (b, i)\right],$$

  where j is the label of the membrane encapsulating membrane i and the notation $(E_i \to \lambda,)^n$, $n > 0$, is taken for a sequence of n productions $E_i \to \lambda$. We should like to recall the fact that the uniport rule $E_i \to \lambda$ is only applicable if the amount e_i of energy assigned to membrane i fulfills $e_i + \Delta e \geq 0$; hence we may be forced to stop in the middle of a matrix, because not enough energy is assigned to membrane i.
- For a rule $(out : a, \Delta e, b)$ assigned to membrane i with $\Delta e \geq 0$ we take the matrix

  $$\left[(a, i) \to (b, j), D_i \to D_i E_i^{\Delta e}\right],$$

  where j is the label of the membrane encapsulating membrane i.
- For a rule $(out : a, \Delta e, b)$ assigned to membrane i with $\Delta e < 0$ we take the matrix

  $$\left[(a, i) \to \widetilde{(b, j)}, (E_i \to \lambda,)^{-\Delta e} \widetilde{(b, j)} \to (b, j)\right],$$

  where j is the label of the membrane encapsulating membrane i.

After the application of a matrix described above, we obtain a valid string representation of the configuration obtained from the previous configuration by applying the corresponding rule in Π. On the other hand, every string obtained from the (complete) application of a matrix to a valid string representation of a reachable configuration c is a valid string representation of the configuration resulting from the application of the corresponding rule in Π to c.

All the symbols introduced so far are non-terminal symbols. Except for the objects of the form$\widetilde{(b,j)}$ we now introduce the corresponding terminal symbol a_t for the non-terminal symbol a and we add the matrices $[a \to a_t]$.

Hence, in total we have obtained the matrix grammar $G = (N, T, S, M)$ with:

$$\begin{aligned}
N &= \{S, D_i, E_i \mid 0 \le i \le d\} \cup \left\{(a,i), \widetilde{(a,i)} | a \in O, 0 \le i \le d\right\}, \\
T &= \left\{a_t \mid a \in \left(N - \left\{\widetilde{(b,j)} \mid b \in O, 0 \le j \le d\right\}\right)\right\}, \\
M &= \{[S \to s] \,|s \in \sigma(initial\ configuration)\} \\
&\cup \left\{\left[(a,j) \to (b,i), D_i \to D_i E_i^{\Delta e}\right] \mid (in : a, \Delta e, b) \in R_i, \Delta e \ge 0\right\} \\
&\cup \left\{\left[(a,j) \to \widetilde{(b,i)}, (E_i \to \lambda,)^{-\Delta e} \widetilde{(b,i)} \to (b,i)\right]\right. \\
&\quad \left. \mid (in : a, \Delta e, b) \in R_i, \Delta e < 0\right\} \\
&\cup \left\{\left[(a,i) \to (b,j), D_i \to D_i E_i^{\Delta e}\right] \mid (out : a, \Delta e, b) \in R_i, \Delta e \ge 0\right\} \\
&\cup \left\{\left[(a,i) \to \widetilde{(b,j)}, (E_i \to \lambda,)^{-\Delta e} \widetilde{(b,j)} \to (b,j)\right]\right. \\
&\quad \left. \mid (out : a, \Delta e, b) \in R_i, \Delta e < 0\right\}.
\end{aligned}$$

Due to the given construction, for $L(G)$ the following holds:

1. Every element in $L(G)$ represents a reachable configuration of Π.
2. If c is a reachable configuration in Π, then $L(G)$ contains a valid string representation of c.

Now we construct a regular set R describing the non-halting configurations of Π :

Let n be the total number of symbols (in the multiset sense) occurring in the initial configuration. Then $R = R_1 \cup R_2 \cup R_3 \cup R_4$ where

- R_1 is the (finite) union of all (regular) sets of the form

$$(O \times D)^{n_1} \{(a,i)\} (O \times D)^{n_2} \{D_0\} \{E_0\}^* ... \{D_j\} \{E_j\}^* ... \{D_d\} \{E_d\}^*$$

  such that $n_1 + n_2 + 1 = n$, $(in : a, \Delta e, b) \in R_j$, region i contains membrane j, and $\Delta e \ge 0$;

- R_2 is the (finite) union of all (regular) sets of the form

$$\begin{aligned}
&(O \times D)^{n_1} \{(a,i)\} (O \times D)^{n_2} \{D_0\} \{E_0\}^* ... \\
&\{D_j\} \{E_j\}^{-\Delta e} \{E_j\}^* ... \{D_d\} \{E_d\}^*
\end{aligned}$$

  such that $n_1 + n_2 + 1 = n$, $(in : a, \Delta e, b) \in R_j$, region i contains membrane j, and $\Delta e < 0$;

- R_3 is the (finite) union of all (regular) sets of the form

$$(O \times D)^{n_1} \{(a,j)\} (O \times D)^{n_2} \{D_0\} \{E_0\}^* ... \{D_j\} \{E_j\}^* ... \{D_d\} \{E_d\}^*$$

  such that $n_1 + n_2 + 1 = n$, $(out : a, \Delta e, b) \in R_j$, and $\Delta e \ge 0$;

- R_4 is the (finite) union of all (regular) sets of the form

$$(O \times D)^{n_1} \{(a,j)\} (O \times D)^{n_2} \{D_0\} \{E_0\}^* ...$$
$$\{D_j\} \{E_j\}^{-\Delta e} \{E_j\}^* ... \{D_d\} \{E_d\}^*$$

  such that $n_1 + n_2 + 1 = n$, $(out : a, \Delta e, b) \in R_j$, and $\Delta e < 0$.

The set R is a finite union of regular sets, i.e., R is a regular set, too, and it describes the situations where a rule of Π is still applicable, hence, the non-halting configurations. Therefore, $(N^* - R)$ contains a lot of garbage, but also all the strings being a valid representation of a reachable configuration must represent a halting configuration.

Finally, let $p : N^* \to \{e_i | 1 \leq i \leq d\}^*$ be the projection mapping E_i to e_i (i.e., $p(E_i) = e_i$), $1 \leq i \leq d$, and erasing all other symbols ($p(X) = \lambda$ for all $X \in N - \{E_i | 1 \leq i \leq d\}$).

In sum, we obtain

$$L(\Pi) = p(L(G) \cap (N^* - R)),$$

i.e., (in the representation as multisets over T) $L(\Pi)$, the set of Parikh vectors generated by Π, is the projection of the intersection of a matrix language with a regular set, hence, due to the closure properties of the family of matrix languages, $L(\Pi)$ is a matrix language, too, which observation concludes the proof. □

If we now combine the two previous lemmas we get the following characterization of $PsMAT^{\lambda}$:

Theorem 10 $PsPE_*(unit) = PsMAT^{\lambda}$.

Due to the construction in Lemma 8 we not only have obtained a characterization of MAT^{λ} by P systems with unit rules and energy assigned to membranes but also a normal form for this kind of P systems, i.e., only one symbol moving through a membrane structure is already sufficient (which of course is the minimal resource needed to obtain reasonable results).

5 Conclusion

We have investigated P systems with unit rules and energy assigned to membranes, which obtain universal computational power when using a priority relation on the rules. In that way, the introduced systems can be used as generating as well as accepting devices for recursively enumerable sets of (vectors of) non-negative integers. On the other hand, for P systems with unit rules and energy assigned to membranes without using the priority relation we rather unexpectedly obtained a characterization of the family of languages generated by context-free matrix grammars with $\lambda-$rules. The results obtained in this paper are already optimal with respect to the size of the multisets transported through a membrane, as in this model we use only one object to be present in the system, i.e., all the results proved for P systems with unit rules and energy assigned to membranes have been obtained by using (the minimal number of) only one symbol moving around the membrane structure. Yet the optimal numbers of membranes necessary for obtaining computational completeness or for characterizing MAT^{λ} still remain open problems (although we conjecture that the number of membranes needed in the universality results is already optimal).

Acknowledgements. This paper was inspired by a research proposal exposed during the Second Brainstorming Week taking place in Sevilla in the first week of February, 2004. Some details of the proposal, concerning the association of energy values to objects, membranes, and/or rules of P systems, are contained in [9], where the simulation of a Fredkin gate is also shown using this new variant of P systems.

References

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

[2] R. Freund: Generalized P-Systems. In: G. Ciobanu, Gh. Păun, Eds.: *Proceedings Fundamentals of Computation Theory, LNCS* 1684, Springer-Verlag, Berlin (1999), 281–292.

[3] R. Freund: Sequential P-Systems. *Romanian Journal of Information Science and Technology* **4**, 1-2 (2001), 77–88.

[4] R. Freund: Energy-Controlled P Systems. In: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds., *Membrane Computing. International Workshop, WMC-CdeA 2002*, Curtea de Argeş, Romania, August 2002, Springer-Verlag, LNCS 2597, Berlin, 2003, 247-260.

[5] R. Freund, M. Oswald: GP Systems with Forbidding Context. *Fundamenta Informaticae* **49**, 1-3 (2002), 81–102.

[6] R. Freund, M. Oswald: P Systems with Conditional Communication Rules Assigned to Membranes, *JALC*, to appear.

[7] R. Freund, Gh. Păun: On the Number of Non-Terminals in Graph-Controlled, Programmed, and Matrix Grammars. In: Margenstern, M., Rogozhin, Y., Eds.: *Proc. Conf. Universal Machines and Computations, Chişinău (2001). LNCS* 2055, Springer-Verlag, Berlin, 2001, 214–225.

[8] R. Freund, Gh. Păun: From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies. *Theoretical Computer Science* **312** (2004), 143–188.

[9] A. Leporati, G. Mauri, C. Zandron: Simulating the Fredkin Gate with Energy–based P Systems, in the present volume.

[10] M.L. Minsky: *Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, New Jersey, 1967.

[11] Gh. Păun: Computing with Membranes. *Journal of Computer and System Sciences* **61**, 1 (2000), 108–143, and TUCS Research Report 208 (1998) (`http://www.tucs.fi`)

[12] Gh. Păun: *Membrane Computing: An Introduction*. Springer-Verlag, Berlin (2002).

[13] Gh. Păun, Y. Suzuki, H. Tanaka: P Systems with Energy Accounting. *Int. J. Computer Math.* **78**, 3 (2001), 343–364.

[14] Rozenberg, G., Salomaa, A., Eds.: *Handbook of Formal Languages.* Springer-Verlag, Berlin, Heidelberg, 1997.

[15] The P Systems Web Page, `http://psystems.disco.unimib.it`