P Machines: An Automata Approach to Membrane Computing *

Gabriel Ciobanu[1] and Mihai Gontineac[2]

[1] Romanian Academy, Institute of Computer Science
Blvd. Carol I nr.8, 700505 Iaşi, Romania
[2] "A.I.Cuza" University, Faculty of Mathematics
Blvd. Carol I nr.11, 700506 Iaşi, Romania
gabriel@iit.tuiasi.ro, gonti@uaic.ro

Abstract. In this paper we present P machines corresponding to membrane systems with a single membrane. We give examples of simple P machines for both P systems with promoters and P systems with priorities. For each case we show that the P machines provide the same result as their corresponding P systems. We present a way of connecting simple P machines, and give an example how the new resulting network corresponds to P systems with more than one membrane.

1 Introduction

Membrane systems (called also P systems) represent a new abstract model of parallel and distributed computing inspired by cell compartments and molecular membranes [10]. A cell is divided in various compartments, each compartment with a different task, and all of them working simultaneously to accomplish a more general task of the whole system. The membranes of a P system determine regions where objects and evolution rules can be placed. The objects evolve according to the rules associated with each region, and the regions cooperate in order to maintain the proper behaviour of the whole system. It is desirable to find good connections with various fields of computer science, including the classic automata theory. There exist some previous attempts [8, 6, 9, 1] in this direction: [8] and [9] present P automata, namely devices which works mainly with communication rules; [6] presents a P transducer as a form of Mealy membrane automata, but the dynamical aspects are not clearly described; in [1] we find a preliminary study of the dynamics of P systems, and the open problems listed at the end of this paper motivate our attempt. We have to mention here the existence of some devices working with multisets, namely multiset automata introduced in [7]. All these models are non-compositional.

In [4] we have introduced two versions of Mealy automata, namely Mealy multiset automata, and elementary Mealy membrane automata. The coalgebraic properties of Mealy multiset automata are presented in [5]. We define here an

* This work has been supported by the research grant CNCSIS 1426/2005

improved version of the elementary Mealy Membrane Automaton named simple P machine by extending the communication capabilities. We provide some examples on how we can use these P machines to describe various features of a membrane system.

2 Mealy Multiset Automata

In order to provide a suitable model for the rules of a membrane, we need the notion of *Mealy multiset automata* (MmA). Roughly speaking, a MmA consists of a *storage location* (a *box* for short) in which we place a multiset over an input alphabet, and a device to translate the input multiset into a multiset over an output alphabet. The way in which a MmA works is described in steps. We have a detection head which detects whether or not a given multiset appears in the input multiset available in the box. If the multiset is detected, then it is removed from the box, and the automaton inserts a multiset over the output alphabet (or a marked symbol if the output alphabet is the same) which cannot be viewed by the detection head. Our automaton stops when no further translation is possible. We say that the submultiset read by the head was translated to a multiset over the output alphabet. We give here only the definitions and the properties which we need for defining the P machines. For more informations see [4, 5].

Formally, a *Mealy multiset automaton* $\mathcal{A} = (Q, V, O, f, g, q_0)$ is defined by
a finite set Q of *states*;
a special state $q_0 \in Q$ which is both initial and final;
a finite set V of objects representing the *input alphabet*;
a finite set O of objects representing the *output alphabet*, and $O \cap V = \emptyset$;
a *state-transition* (*partial*) *mapping* $f : Q \times \mathbb{N}\langle V\rangle \to \mathcal{P}(Q)$;
an *output* (*partial*) *mapping* $g : Q \times \mathbb{N}\langle V\rangle \to \mathcal{P}(\mathbb{N}\langle O\rangle)$.

If $\mid f(q,a) \mid \leq 1$ we say that $\mathcal{A}$ is *Q-deterministic*, and if $\mid g(q,a) \mid \leq 1$ we say that $\mathcal{A}$ is *O-deterministic*.

A MmA is endowed with a box where it receives a multiset. After that, it begins to process this multiset over V passing through different *configurations*. It starts with a multiset from $\mathbb{N}\langle V\rangle$, and ends with a multiset from $\mathbb{N}\langle V \cup O\rangle$. A *configuration* of $\mathcal{A}$ is a triple $(q, \alpha, \bar{\beta})$ where $q \in Q$, $\alpha \in \mathbb{N}\langle V\rangle$, $\bar{\beta} \in \mathbb{N}\langle O\rangle$. We say that a configuration $(q, \alpha, \bar{\beta})$ *passes* to $(s, \alpha - a, \bar{\beta} + \bar{b})$ or that we have a *transition* between these configurations, if there is $a \subseteq \alpha$ such that $s \in f(q,a)$ and $\bar{b} \in g(q,a)$. We denote this by $(q, \alpha, \bar{\beta}) \vdash (s, \alpha - a, \bar{\beta} + \bar{b})$. We denote by $\vdash^*$ the reflexive and transitive closure of $\vdash$.

We can alternatively define a configuration to be a pair (q, α) where $\alpha \in \mathbb{N}\langle V \cup O\rangle$, and the transition relation is $(q, \alpha) \vdash (s, \alpha - a + \bar{b})$, with the same conditions as above.

A multiset $\alpha \in \mathbb{N}\langle V\rangle$ is said to be a *totally consumed multiset* (*tc-multiset*) for $\mathcal{A}$ if, starting from a configuration $(q_0, \alpha, \varepsilon)$, the MmA can pass through configurations until it arrives in a configuration $(q_0, \varepsilon, \bar{\beta})$ (i.e. there exists $(q_0, \alpha, \varepsilon) \vdash^* (q_0, \varepsilon, \bar{\beta})$).

A multiset $\alpha \in \mathbb{N}\langle V \rangle$ is said to be a *consumed multiset* (*c-multiset*) for $\mathcal{A}$ if, starting from a configuration $(q_0, \alpha, \varepsilon)$, the MmA can pass through configurations until it arrives in a configuration $(q, \varepsilon, \bar{\beta})$ (i.e. there exists $(q_0, \alpha, \varepsilon) \vdash^* (q, \varepsilon, \bar{\beta})$).

In both these cases, we say also that α was *entirely translated to* $\bar{\beta}$. In all the other situations we say that $\alpha \in \mathbb{N}\langle V \rangle$ is *partially consumed* (*pc-multiset*), or it is *partially translated.*

Given two MmA's $\mathcal{A} = (Q, V, O, f, g)$ and $\mathcal{A}' = (Q', V, O, f', g')$, a function $h : Q \to Q'$ is a *morphism* from $\mathcal{A}$ to $\mathcal{A}'$ if the following conditions are satisfied:

- $h(f(q, a)) = f'(h(q), a)$, for all $q \in Q$ and for all $a \in \mathbb{N}(V)$;
- $g(q, a) = g'(h(q), a)$, for all $q \in Q$ and for all $a \in \mathbb{N}(V)$.

If $h : Q \to Q'$ is a morphism between $\mathcal{A}$ and $\mathcal{A}'$, we denote this by $h : \mathcal{A} \to \mathcal{A}'$. It can be easily proved that, for fixed alphabets V and O, the collection of all Mealy multiset automata together with the morphisms between them form a category denoted by $\mathcal{MA}_{VO}$.

Let $h : \mathcal{A} \to \mathcal{A}'$ be a morphism and $(q, \alpha, \bar{\beta})$ a configuration of $\mathcal{A}$ such that we have a transition $(q, \alpha, \bar{\beta}) \vdash (s, \alpha - a, \bar{\beta} + \bar{b})$. This means that $s = f(q, a)$ and $\bar{b} = g(q, a)$. We obtain then that $h(s) = h(f(q, a)) = f'(h(q), a)$, and $\bar{b} = g(q, a) = g'(h(q), a)$, and so we get $(h(q), \alpha, \bar{\beta}) \vdash (h(s), \alpha - a, \bar{\beta} + \bar{b})$.

As a consequence, we obtain the following result: Let $h : \mathcal{A} \to \mathcal{A}'$ be a morphism of MmA's. If the multiset $\alpha \in \mathbb{N}(V)$ is a tc (c, pc)-multiset for $\mathcal{A}$, then α has the same nature for$\mathcal{A}'$.

For the categorical properties of MmA's, as well as MmA's *behaviour* and *bisimulation relation*, we refer to [4, 5]

2.1 Restricted direct product of Mealy multiset automata

Let $\mathcal{A}_i = (Q_i, V, O, f_i, g_i)$ be a finite family of Mealy multiset automata, and B_i their corresponding boxes ($i = \overline{1, n}$). We can connect them in *parallel* in order to obtain a new MmA defined by $\mathcal{A} = \bigwedge_{i=1}^{n} \mathcal{A}_i = (\times_{i=1}^{n} Q_i, V, O, f, g)$, called the *restricted direct product* of $\mathcal{A}_i$, where:

- $f((q_1, q_2, ..., q_n), a) = (f_1(q_1, a), f_2(q_2, a), ..., f_n(q_n, a))$;
- $g((q_1, q_2, ..., q_n), a) = (g_1(q_1, a), g_2(q_2, a), ..., g_n(q_n, a))$;
- the box B of $\mathcal{A}$ is the disjoint union $\bigsqcup_{i=1}^{n} B_i$ of B_i, $i = \overline{1, n}$;
- a *configuration* of $\mathcal{A}$ is a triple $(q, \alpha, \bar{\beta})$, where $q = (q_1, q_2, ..., q_n)$, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$, and $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2, ..., \bar{\beta}_n)$;
- the (*asynchronous*) *transition relation* of $\mathcal{A}$ is given by $(q, \alpha, \bar{\beta}) \vdash (s, \alpha - a, \bar{\beta} + \bar{b})$ if and only if there is at least an $i \in \overline{1, n}$ such that $s_i \in f_i(q_i, a_i)$ and $\bar{b}_i \in g_i(q_i, a_i)$.

The asynchronous nature of the transition is closer to biology: "The biological systems are massively concurrent, heterogeneous, and asynchronous" [3].

3 P Machines

The previous automata-like approaches for P systems are based on "top-down" communication rules, and the "maximal parallel and nondeterministic" way of applying the rules is more or less visible. We present a new automata-like systems such that every membrane is able to evolve and communicate. While the parallel part is given by the way we connect various MmA's, the "maximality" and the "nondeterministic" aspects are ensured by a "smart" device that is formalized by a *control resource mapping*. The control resource mapping (CRM) is used in both *computing with priorities* and *computing with promoters*. The advantages of such an approach come from the fact that we proceed bottom-up, i.e. we start from a single membrane, then we indicate how we connect single membranes such that these devices can model P systems, as well as tissue-like P systems.

3.1 Simple P Machine

Generally speaking a *simple P machine* (shortly sPM) is built from:

- a *resource box* B together with a *control resource mapping* CRM;
- n Mealy multiset automata, connected in parallel. They consume and translate tc-multisets from their boxes (allocated by CRM) into marked multisets over the same alphabet. We use marked multisets for the output because the input alphabet and the output one must be disjoint; the marking also shows us where the corresponding multiset must go (i.e. all marks belong to a set of targets).

Attached to such a machine we have a *distribution map* denoted by DM. A DM is involved in refreshing the content of the resource box, and in communication with other P machines. We define a cascade product of a P machine with itself in order to go to the next computation step of the P system.

Definition 1. *A* simple P machine *is given by* $\mathcal{M} = (V, \bigwedge_{i=1}^{n} \mathcal{A}_i, O, B, CRM)$, *where:*

- $V = \{a_1, a_2, ..., a_m\}$ *is an input alphabet;*
- $\mathcal{A}_i = (Q_i, V, O_i, f_i, g_i)$ *are MmA's connected in parallel;*
- O *is an output alphabet* $O = \bigcup_{i=1}^{n} O_i$ *defined by the output alphabets* $O_i = V \times T_i$, *where target sets* T_i *indicate the indexes of the simple P machines connected to* $\mathcal{M}$; *whenever we consider only an isolated simple P machine, then* $T_i = \{0\}$ *for all* $i = 1, \ldots, n$;
- B *is the box where* $\mathcal{M}$ *receives a multiset for processing;*
- $CRM : \mathbb{N}\langle V \rangle \to \mathcal{P}(\mathbb{N}\langle V \rangle^{n+1})$ *is the* control resource mapping; *this map assigns the resources available in* B *to the bags of the* $\mathcal{A}_i's$. *It has one supplementary component indicating the multiset unassigned (the remaining multiset). So if* $w = w_1a_1 + w_2a_2 + ... + w_ma_m \in \mathbb{N}\langle V \rangle$ *is the input multiset,* $CRM(w)$ *is of the form* $(l_1x_1, l_2x_2, ..., l_nx_n, w')$, *where* $\sum_{i=1}^{n} l_ix_i + w' = w$ *and* x_i *are tc-multisets of* $\mathcal{A}_i$.

We have a *distribution mapping* DM attached to a simple P machine. It plays the role of an *interface*. DM is defined on $\mathbb{N}\langle V\rangle \cup \bigcup_{i=1}^{n} \mathbb{N}\langle O_i\rangle$, and takes values in $(\mathbb{N}\langle V\rangle)^T$, where $T = \bigcup_{i=1}^{n} T_i$. DM removes the target component of a multiset, and puts that multiset in the box of the corresponding P machine. A *step of computation* starts with an input multiset, followed by a repartition made by CRM, a translation made by the parallel MmA's (i.e. $\bigwedge_{i=1}^{n} \mathcal{A}_i$), and a distribution done by DM. The distribution conditions satisfied by $CRM(w)$ can define various features of a simple P machine.

Maximal parallel and nondeterministic *sPM*

Given a simple P machine $\mathcal{M} = (V, \bigwedge_{i=1}^{n} \mathcal{A}_i, O, B, CRM)$, we define the control resource mapping $CRM : \mathbb{N}\langle V\rangle \to \mathcal{P}(\mathbb{N}\langle V\rangle^{n+1})$ such that this map assigns the resources available in B to the bags of the $\mathcal{A}_i'$s in a maximal parallel and nondeterministic way. If $w = w_1a_1 + w_2a_2 + ... + w_ma_m = \sum_{j=1}^{m} w_ja_j \in \mathbb{N}\langle V\rangle$ is the input multiset, then $CRM(w)$ is of the form $(l_1x_1, l_2x_2, ..., l_nx_n, w')$ where $\sum_{i=1}^{n} l_ix_i + w' = w$, x_i are tc-multisets of $\mathcal{A}_i$ and $x_i \not\subseteq w'$, for all $i = \overline{1,n}$. In this way we model the maximal parallel feature, because the multiset which remains unprocessed in the box is minimal (does not contain any submultiset that can be processed by a MmA from the direct product). The nondeterminism feature is given by the fact that $CRM(w)$ can take any value from $\{(l_1x_1, l_2x_2, ..., l_nx_n, w') \mid \sum_{i=1}^{n} l_ix_i + w' = w,\ x_i$ is a tc-multiset of $\mathcal{A}_i, x_i \not\subseteq w', i = \overline{1,n}\}$. Due to the competition on resources, l_i can take any value between 0 and $max\{l \in \mathbb{N} \mid lx_i \subseteq w\}$. x_i belong to $\mathbb{N}\langle V\rangle$, and so it can be written as a linear combination of a_i's, namely $x_i = \sum_{j=1}^{m} \alpha_{ij}a_j$, $\alpha_{ij} \in \mathbb{N}$. Hence $\sum_{i=1}^{n} l_ix_i = \sum_{i=1}^{n}(\sum_{j=1}^{m} l_i\alpha_{ij}a_j) \subseteq w$, and so $\sum_{j=1}^{m}(\sum_{i=1}^{n} l_i\alpha_{ij})a_j \subseteq \sum_{j=1}^{m} w_ja_j$. This means that for all $j = 1, ..., m$ we have $\sum_{i=1}^{n} l_i\alpha_{ij} \leq w_j$.

Lemma 1. *$(l_1, l_2, ..., l_n)$ is a solution of the following system:*

$$\sum_{i=1}^{n} l_i'\alpha_{ij} \leq w_j,\ j = \overline{1,m}.$$

On the other hand, if $(l_1, l_2, ..., l_n)$ is a solution, it must satisfy the maximality condition, i.e. for all x_k, $x_k \not\subseteq w' = w - \sum_{i=1}^{n} l_ix_i$. It follows that for all $k = \overline{1,n}$, $x_k \not\subseteq \sum_{j=1}^{m} w_ja_j - \sum_{i=1}^{n} l_ix_i$ iff $\sum_{j=1}^{m} \alpha_{kj}a_j \not\subseteq \sum_{j=1}^{m} w_ja_j - \sum_{i=1}^{n}(\sum_{j=1}^{m} l_i\alpha_{ij}a_j) = \sum_{j=1}^{m}(w_j - \sum_{i=1}^{n} l_i\alpha_{ij})a_j$. Thus, if $(l_1, l_2, ..., l_n)$ is a solution for all $k = \overline{1,n}$, then there is a $j_k \in \overline{1,m}$ such that $\alpha_{kj_k} > w_{j_k} - \sum_{i=1}^{n} l_i\alpha_{ij_k}$.

In order to obtain a compact writing, we can use the following method of finding the coefficients $l_1, l_2, ..., l_n$ for $CRM(w)$ taken from linear algebra:

1. We use a vector $W = (w_1, w_2, ..., w_m)$, and the matrix of the tc-multisets of the restricted direct product $\bigwedge_{i=1}^{n} \mathcal{A}_i$, $A = (\alpha_{ij})_{n\times m}$
2. We find the set of *admissible solutions*, i.e. the solutions of the system $L'A \leq W$, where $L' = (l_1', l_2', ..., l_n')$. Let $\mathcal{L}^a$ be this set; obviously it is not empty, since $(0, 0, ..., 0) \in \mathcal{L}^a$.

3. Choose one $L \in \mathcal{L}^a$, and then calculate $W - LA$;
4. Calculate the matrix $M_L = \begin{pmatrix} W - LA \\ ... \\ W - LA \end{pmatrix}_{n \times m} - A.$

According to well-known results of linear algebra, we have the following theorem.

Theorem 1. *A vector $L = (l_1, l_2, ..., l_n)$ is an optimal solution (i.e. the simple P machine is working in a maximal parallel way) if and only if every line of the corresponding matrix M_L has at least a negative element.*

Maximal consuming and non deterministic *sPM*

We refer here to a maximal consuming P system, namely a maximal parallel P system which consumes the largest number of resources among all the possibilities. Considering a simple P machine $\mathcal{M} = (V, \bigwedge_{i=1}^{n} \mathcal{A}_i, O, B, CRM)$, the maximal parallelism and nondeterminism is given by $CRM(w) \in \{(l_1x_1, l_2x_2, ..., l_nx_n, w') \mid \sum_{i=1}^{n} l_ix_i + w' = w,\ x_i$ is a tc-multiset of $\mathcal{A}_i, x_i \nsubseteq w', i = \overline{1,n}\}$. Due to the competition on resources, l_i can take any value between 0 and $max\{l \in \mathbb{N} \mid lx_i \subseteq w\}$. x_i belong to $\mathbb{N}\langle V \rangle$, so it can be written as a linear combination of a_i, namely $x_i = \sum_{j=1}^{m} \alpha_{ij}a_j$, $\alpha_{ij} \in \mathbb{N}$. Hence $l_ix_i = \sum_{j=1}^{m} l_i\alpha_{ij}a_j$. Since $l_ix_i \subseteq w = \sum_{j=1}^{m} w_ja_j$, we obtain that $l_i\alpha_{ij} \leq w_j$ for all $j = 1, ..., m$. The difference with respect to a maximal parallel approach is given by an additional *objective function* $\Phi : \mathbb{N}^n \to \mathbb{N}\langle V \rangle$ given by $\Phi(l'_1, l'_2, ..., l'_n) = \sum_{i=1}^{n} l'_ix_i$.

Theorem 2. *A simple P machine is working in a maximal consuming and nondeterministic way if and only if for any given input multiset w, the coefficients l_i of $CRM(w)$ are solutions of the following problem of integer programming:*

$$\begin{cases} \sum_{i=1}^{n} l'_i\alpha_{ij} \leq w_j & j = \overline{1,m} \\ l'_i \geq 0 & i = \overline{1,n} \\ max\, \Phi(l'_1, l'_2, ..., l'_n) \end{cases}.$$

Example 1. We give here a simple example.

$$\mathcal{M} = (\{a, b, c\}, \bigwedge_{i=1}^{2} \mathcal{A}_i, \{(a, 0), (b, 0), (c, 0)\}, B, CRM),$$

with $TC(\mathcal{A}_1) = \{n(a + b) | n \in \mathbb{N}\}$, $TC(\mathcal{A}_2) = \{n(a + c) | n \in \mathbb{N}\}$. $\mathcal{A}_1$ translates $l(a+b)$ into $l(c, 0)$, and $\mathcal{A}_2$ translates $l(a+c)$ into $l(b, 0)$. Let us consider the input multiset $w = 3a+2b+2c$. Then $CRM(w) \in \{(2(a+b), a+c, c), (a+b, 2(a+c), b)\}$. Simple calculations lead us to the output of the machine which can be either $(b, 0) + 3(c, 0)$ or $3(b, 0) + (c, 0)$.

Simple P Machines with promoters

In P machines with promoters, some of the objects of the input alphabet of arbitrary $\mathcal{A}_i$ are special elements called *promoters*. Only their presence in the resource box allows the corresponding $\mathcal{A}_i$ to receive the related resources in its input bag. If $V = \{a_1, a_2, ..., a_m\}$, we consider that the last $m - p$ elements are promoters, and denote their set by P, i.e. $P = \{a_{p+1}, a_{p+2}, ..., a_m\}$. We denote by P_i the set of promoters of $\mathcal{A}_i$; it can be empty if $\mathcal{A}_i$ has not promoters.

Let $w = w_1a_1 + w_2a_2 + ... + w_ma_m \in \mathbb{N}\langle V\rangle$ the input multiset of the sPM $\mathcal{M} = (V, \bigwedge_{i=1}^{n} \mathcal{A}_i, O, B, CRM)$ with promoters in P. If $P_i = \{a_{i_1}, a_{i_2}, ..., a_{i_t}\}$, then $\mathcal{A}_i$ can receive from CRM an input multiset in its bag if and only if $w_{i_1}, w_{i_2}, ..., w_{i_t}$ are all different from 0, i.e. $\prod_{s=1}^{t} w_{i_s} \neq 0$. As usual, we have that $CRM(w) \in \{(l_1x_1, l_2x_2, ..., l_nx_n, w') \mid \sum_{i=1}^{n} l_ix_i + w' = w\}$. An approach similar to the previous subsections leads us to the following result:

Proposition 1. *$\mathcal{A}_i$ receives an input multiset in its bag iff* $(\star)$ $\prod_{a_{i_s} \in P_i} w_{i_s} \neq 0$. *Moreover, the coefficients of the tc-multisets for all the $\mathcal{A}_i'$s which satisfy the previous condition* $(\star)$ *are obtained as solutions of the system:*

$$\sum_{i \in I} l_i' \alpha_{ij} \leq w_j, j = \overline{1, m},$$

where we denote by $I \subseteq \{1, 2, ..., n\}$ the set of indices of specific $\mathcal{A}_i'$s where the condition $(\star)$ *is satisfied.*

Simple P Machines with priorities

Let $\mathcal{M} = (V, \bigwedge_{i=1}^{n} \mathcal{A}_i, O, B, CRM)$ be a simple P machine. We consider a *partial priority order relation* over the set of the Mealy multiset automata that are part of $\mathcal{M}$, and we denote this priority order by " $\leq$ ". The possible values of CRM for an input multiset $w = \sum_{j=1}^{m} w_j a_j$ are more difficult to be obtained, at least from the formal point of view. We define a relation of *immediate precedence*:

$\mathcal{A}_i \preceq \mathcal{A}_j$ if $\mathcal{A}_i \leq \mathcal{A}_j$ and there is no $\mathcal{A}_k$ such that $\mathcal{A}_i < \mathcal{A}_k < \mathcal{A}_j$.

This relation defines some *levels of precedence* given by a partition of the set $\{1, 2, ..., n\}$ of MmA's indices. The levels of precedence are defined by:

level 0: $\mathcal{L}_0$ contains the indices of the MmA's which have no predecessor

...

level $(s + 1)$: $\mathcal{L}_{s+1} = \{j \in \overline{1, m} \mid (\exists) r_j \in L_s$ such that $\mathcal{A}_j \prec \mathcal{A}_{r_j}\}$

...

Suppose now that $\mathcal{L}_0, \mathcal{L}_1, ..., \mathcal{L}_t$ are the levels of precedence. As in the nondeterministic and maximal parallel case, we denote by $A = (\alpha_{ij})_{n \times m}$ the matrix of the tc-multisets of the restricted direct product $\bigwedge_{i=1}^{n} \mathcal{A}_i$. With respect to the levels of precedence, we denote by A_s the matrix obtained from A by erasing all its lines which do not have indexes from $\mathcal{L}_s$. We apply a similar procedure for the unknown coefficients, denoting by L_s' the vector obtained from $L' = (l_1', l_2', ..., l_n')$ by removing the components which have not indexes from $\mathcal{L}_s$. We obtain the possible values for $CRM(w)$ by solving the following set of problems:

Level $\mathcal{L}_0$:

Step 0.1 Find the set of *admissible solutions* for level $\mathcal{L}_0$, i.e. the solutions of the system $L_0'A_0 \leq W$. Let $\mathcal{L}_0^a$ be this set (obviously it is non-empty because $(0,0,...,0) \in \mathcal{L}_0^a$).

Step 0.2 For all $L_0 \in \mathcal{L}_0^a$, calculate $W - L_0A_0$, and then consider a matrix $M_{L_0} = \begin{pmatrix} W - L_0A_0 \\ ... \\ W - L_0A_0 \end{pmatrix} - A_0$.

Step 0.3 If every line of M_{L_0} contains at least a negative element, the corresponding admissible solution L_0 is an optimal one, and so it can be chosen. We have a nondeterministic choice, since it can have several optimal solutions. Let $w^0 = w - \sum_{i \in L_0} l_i x_i$ such that $w^0 = \sum_{j=1}^m w_j^0 a_j$. We denote by W^0 its corresponding vector.

Level $\mathcal{L}_1$:

Step 1.1 Find the set of *admissible solutions* for level $\mathcal{L}_1$, i.e. the solutions of the system $L_1'A_1 \leq W^0$. Let $\mathcal{L}_1^a$ be this set (obviously it is non-empty because $(0,0,...,0) \in \mathcal{L}_1^a$).

Step 1.2 For all $L_1 \in \mathcal{L}_1^a$, calculate $W^0 - L_1A_1$, and then consider a matrix $M_{L_1} = \begin{pmatrix} W^0 - L_1A_1 \\ ... \\ W^0 - L_1A_1 \end{pmatrix} - A_1$.

Step 1.3 If every line of M_{L_1} contains at least a negative element, the corresponding admissible solution L_1 is an optimal one, and so it can be chosen. We have a nondeterministic choice, since it can have several optimal solutions. Let $w^1 = w^0 - \sum_{i \in L_1} l_i x_i$ such that $w^1 = \sum_{j=1}^m w_j^1 a_j$. We denote by W^1 its corresponding vector.

...

Suppose that we have $\{l_i, i \in L_r\}$, $w^r = w^{r-1} - \sum_{i \in L_r} l_i x_i$, $w^r = \sum_{j=1}^m w_j^r a_j$, and we denote by W^r its corresponding vector.

Level $\mathcal{L}_r$:

Step r.1 Find the set of *admissible solutions* for level $\mathcal{L}_{r+1}$, i.e. the solutions of the system $L_{r+1}'A_{r+1} \leq W^r$. Let $\mathcal{L}_{r+1}^a$ be this set (obviously it is non-empty because $(0,0,...,0) \in \mathcal{L}_{r+1}^a$).

Step r.2 For all $L_{r+1} \in \mathcal{L}_{r+1}^a$, calculate $W^r - L_{r+1}A_{r+1}$, and then consider the matrix $M_{L_{r+1}} = \begin{pmatrix} W^r - L_{r+1}A_{r+1} \\ ... \\ W^r - L_{r+1}A_{r+1} \end{pmatrix} - A_{r+1}$.

Step r.3 If every line of $M_{L_{r+1}}$ contains at least a negative element, the corresponding admissible solution L_{r+1} is an optimal one, and so it can be chosen.

Solving the existing levels of precedence, we get the possible values of $CRM(w)$.

4 Examples of Simple P Machines

We describe first how priorities and promoters are useful in defining arithmetical operations in membrane systems. After each example we provide the simple P machine describing the corresponding membrane systems.

4.1 Control Mechanisms in P Systems

In P systems several mechanisms allow to control the computation. In general the objects and the rules governing the computation are chosen in a nondeterministic way. Moreover, this choice is exhaustive in the sense that no rule can be further applied in the same evolution step: this is the maximal parallel rewriting. A global clock is assumed, that is the same clock for all the regions of a membrane system. At each tick of this clock, a current configuration of the system is transformed into another one, and so defining a transition between the configurations of the system. A sequence of transitions is called a computation. A computation is halting if it reaches a halting configuration, one where no rules are applicable at all.

We have various control mechanisms in membrane systems. They are inspired by some biological entities. For instance, we have catalysts representing objects which appear on both left-hand and right-hand sides of a rule. The catalysts directly participate in rules (but are not modified by them), and they are counted as any other object such that the number of applications of a rule involving a catalyst is as large as the number of copies of the catalyst. They can be used to apply a certain rule in a sequential way, increasing the control of using the rule. Another controlling mechanism is given by activators, a formal representation of enzymes. An activator is related to a rule. The rules need activators to be applied, so the parallelism of each rule is limited to the number of its activators. The activators can evolve in the same step (this is not possible for catalysts).

Here we refer mainly to control mechanisms defined over sets of rules rather than individual rules; such mechanisms are given by priorities and promoters. A priority relation among rules means that in each region we have a partial order relation on the set of rules, and a rule can be chosen (to process a multiset of objects) only if no rule of a higher priority is applicable in the same region. Promoters and inhibitors formalize the reaction enhancing and reaction prohibiting roles of various substances (molecules) present in cells. In membrane systems, promoters and inhibitors are represented as multisets of objects associated with given sets of rules. A rule from such a set of a given region can be used only if all the promoting objects are present, and all the inhibiting objects are not present in that region. From the generative point of view, there is a symmetry between the two ideas: systems with promoters are equal in power to systems with inhibitors, and they characterize the recursively enumerable sets of natural numbers. Membrane systems with promoters/inhibitors achieve universal computations in a simpler way. From a technical point of view, it is much easier to work with promoters. The systems become simpler; if we have enough promoters, then systems with only one membrane are already universal.

Regarding the difference between promoters and catalysts, we can say that the catalysts directly participate in rules, and they are counted as objects required by rules, and the number of rule application in parallel is as large as the number of catalysts. In the case of promoters, the presence of only one promoter makes it possible to use a rule involving that promoter as many times as possible, without any restriction.

4.2 Membrane systems and their corresponding *sPM*

We present some examples of P systems implementing arithmetic operations on numbers represented by the numbers of objects. In these examples we use priorities and promoters as control mechanisms in membrane computing, presenting membrane systems with priorities and promoters for multiplication. Other arithmetical operations on numbers represented by using unary and binary compact encodings are presented in [2].

Multiplication with promoters: Figure 1 presents a P system Π_1 with promoters for multiplication of n (objects a) by m (objects b), the result being the number of objects d in membrane 0. The object a is a promoter in the rule $b \rightarrow bd|_a$, i.e., this rule can only be applied in the presence of object a. The available m objects b are used in order to apply m times the rule $b \rightarrow bd|_a$ in parallel; based on the availability of a objects the rule $au \rightarrow u$ where u is a catalyst is applied in the same time and consumes an a. The procedure is repeated until no object a is present within the membrane. Note that each time when one object a is consumed, then m objects d are generated.

$$\Pi_1 = (V, \mu, w_0, R_0, 0),$$
$$V = \{a, b, d, u\},\ \mu = [_0]_0,\ w_0 = a^n b^m u,$$
$$R_0 = \{r_1 : b \rightarrow bd|_a,\ r_2 : au \rightarrow u\}.$$

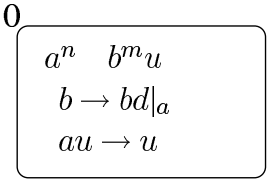


Fig. 1. Multiplication with promoters

We describe now a simple P machine computing the multiplication presented by the previous P systems. We have only one membrane, and we denote it with 0; therefore the target can be only 0. $\mathcal{M} = (V, \mathcal{A}_1 \bigwedge \mathcal{A}_2, O, B, CRM)$ with the input alphabet is $V = \{a, b, c, d, u\}$ and the output alphabet is $O = \{(a, 0), (b, 0), (c, 0), (d, 0), (u, 0)\}$.

$\mathcal{A}_1$ translates b into $(b, 0) + (d, 0)$, $\mathcal{A}_2$ translates $a + u$ into $(u, 0)$.
CRM is defined as

$CRM(k_1 a + k_2 b + k_3 u + k_4 d) = (l_1 b, l_2(a + u), w')$, where
$l_1 b + l_2(a + u) + w' = k_1 a + k_2 b + k_3 + k_4 d$, and

$l_1 = \begin{cases} k_2 \text{ if } k_1 \neq 0 \\ 0 \text{ if } k_1 = 0 \end{cases}$, $l_2 = min\{k_1, k_3\}$.

The distribution mapping is defined by

$DM(l_1((d,0)+(b,0))+l_2(u,0)) = w' + l_1 b + l_1 d + l_2 u$,

where w' is the content of B before applying the resource distribution.

Proposition 2. *$\mathcal{M}$ computes the product of two positive integers.*

Proof. We insert initially $na+mb+u$ in the resource box. We have the following sequence of computations:

$na + mb + 1u \Longrightarrow_{CRM} (mb, a+u, (n-1)a) \Longrightarrow_{\mathcal{A}_1 \wedge \mathcal{A}_2}$
$(m((b,0)+(d,0)), (u,0), (n-1)a) \Longrightarrow_{DM} (n-1)a + mb + u + md \Longrightarrow_{CRM}$
$(mb, a+u, (n-2)a + md) \Longrightarrow_{\mathcal{A}_1 \wedge \mathcal{A}_2}$
$(m((b,0)+(d,0)), (u,0), (n-2)a + md) \Longrightarrow_{DM} (n-2)a + mb + u + 2md$
...
$\Longrightarrow_{CRM} (mb, a+u, a+(n-2)md) \Longrightarrow_{\mathcal{A}_1 \wedge \mathcal{A}_2}$
$(m((b,0)+(d,0)), (u,0), a+(n-2)md) \Longrightarrow_{DM} a + mb + u + (n-1)md$
$\Longrightarrow_{CRM} (mb, a+u, (n-1)md) \Longrightarrow_{\mathcal{A}_1 \wedge \mathcal{A}_2}$
$(m((b,0)+(d,0)), (u,0), (n-1)d) \Longrightarrow_{DM} mb + u + nmd.$

Multiplication with priorities: Figure 2 presents a membrane system Π_2 with priorities for multiplication of n (objects a) by m (objects b), the result being the number of objects d in membrane 0.

$\Pi_2 = (V, \mu, w_0, (R_0, \rho_0), 0)$,
$V = \{a, b, d, e, u, v\}, \mu = [_0]_0, w_0 = a^n b^m u$,
$R_0 = \{r_1 : bv \to dev,\ r_2 : av \to u,\ r_3 : eu \to dbu,\ r_4 : au \to v\}$,
$\rho_0 = \{r_1 > r_2,\ r_3 > r_4\}$.

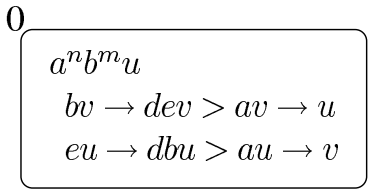


Fig. 2. Multiplication with priorities

We use the priority relation between rules; for instance $bv \to dev$ has a higher priority than $av \to u$, meaning the second rule is applied only when the first one cannot be applied anymore. Initially only the rule $au \to v$ can be applied, generating a catalyst v which activates m times the rule $bv \to dev$. Then $av \to u$ consumes an a, and transform the catalyst v into a catalyst u. Now $eu \to dbu$ is applied m times, followed by another change of catalyst u into a catalyst v by consuming an a (this is done by the rule $au \to v$). The procedure is repeated until no object a is present within the membrane. It is easy to note that each time when one object a is consumed, then m objects d are generated.

We describe now a simple P machine that does the same job. We have only one membrane denoted by 0, and the target can be only 0. We have $\mathcal{M} = (V, \bigwedge_{i=1}^{4} \mathcal{A}_i, O, B, CRM)$, where the alphabets are $V = \{a, b, c, d, e, u, v\}$ and $O = \{(a,0), (b,0), (c,0), (d,0), (e,0), (u,0), (v,0)\}$.

$\mathcal{A}_1$ translates $e+u$ into $(d,0)+(b,0)+(u,0)$, $\mathcal{A}_2$ translates $a+u$ into $(v,0)$, $\mathcal{A}_3$ translates $b+v$ into $(d,0)+(e,0)+(v,0)$, and $\mathcal{A}_4$ translates $a+v$ into $(u,0)$. CRM is defined as $CRM(k_1a + k_2b + k_3d + k_4e + k_5u + k_6v)=$

$=(l_1(e+u), l_2(a+u), l_3(b+v), l_4(a+v), w')$, where

$l_1(e+u)+l_2(a+u)+l_3(b+v)+l_4(a+v)+w' = k_1a+k_2b+k_3c+k_4d+k_5e+k_6f$, and

$l_1 = min\{k_4, k_5\}$,

$l_2 = \begin{cases} 0 & l_1 \neq 0 \\ min\{k_1, k_5\} & l_1 = 0 \end{cases}$,

$l_3 = min\{k_2, k_6\}$,

$l_4 = \begin{cases} 0 & l_3 \neq 0 \\ min\{k_1, k_6\} & l_3 = 0 \end{cases}$

The distribution mapping is defined by

$DM(l_1((d,0)+(b,0)+(u,0))+l_2(v,0)+l_3((d,0)+(e,0)+(v,0))+l_4(u,0)) =$

$= w' + l_1b + (l_1 + l_3)d + l_3e + (l_1 + l_4)u + (l_2 + l_3)v,$

where w' is the content of B before applying the distribution mapping.

Proposition 3. *$\mathcal{M}$ can compute the product of two positive integers*

Proof. We insert initially $na + mb + u = na + mb + 0d + 0e + 1u + 0v$ in the resource box; the result is obtained as the coefficient of d. We have the following sequence of computations.

Step 1:

We consume first one a and one u to produce one v:

$na + mb + u = na + mb + 0d + 0e + 1u + 0v \Longrightarrow_{CRM}$

$(0, a + u, 0, 0, (n-1)a + mb) \Longrightarrow_{\bigwedge \mathcal{A}_i}$

$(0, (v,0), 0, 0, (n-1)a + mb) \Longrightarrow_{DM} (n-1)a + mb + v.$

Next m steps consume m objects b, and produce m objects d and m objects e:

$\Longrightarrow_{CRM} (0, 0, b + v, 0, (n-1)a + (m-1)b) \Longrightarrow_{\bigwedge \mathcal{A}_i}$

$(0, 0, (d,0) + (e,0) + (v,0), (n-1)a + (m-1)b) \Longrightarrow_{DM}$

$(n-1)a + (m-1)b + d + e + v \Longrightarrow_{CRM}$

$(0, 0, b + v, 0, (n-1)a + (m-2)b + d + e) \Longrightarrow_{\bigwedge \mathcal{A}_i}$

$(0, 0, (d,0) + (e,0) + (v,0), (n-1)a + (m-2)b + d + e) \Longrightarrow_{DM}$

$(n-1)a + (m-2)b + 2d + 2e + v$

...

$\Longrightarrow_{CRM} (0, 0, b + v, 0, (n-1)a + (m-1)d + (m-1)e) \Longrightarrow_{\bigwedge \mathcal{A}_i}$

$(0, 0, (d,0) + (e,0) + (v,0), (n-1)a + (m-1)d + (m-1)e) \Longrightarrow_{DM}$

$(n-1)a + md + me + v.$

Step 2:

We consume one a and one v in order to produce one u:

$(n-1)a+md+me+v \Longrightarrow_{CRM} (0,0,0,a+v,(n-2)a+md+me) \Longrightarrow_{\bigwedge \mathcal{A}_i}$
$(0,0,0,(u,0),(n-2)a+md+me) \Longrightarrow_{DM} (n-2)a+md+me+u$

Next m steps consume m objects e, and produce m objects b and m objects d:

$\Longrightarrow_{CRM} (e+u,0,0,0,(n-2)a+md+(m-1)e) \Longrightarrow_{\bigwedge \mathcal{A}_i}$
$((d,0)+(b,0)+(u,0),0,0,0,(n-2)a+md+(m-1)e) \Longrightarrow_{DM}$
$(n-2)a+b+(m+1)d+(m-1)e+u \Longrightarrow_{CRM}$
$((d,0)+(b,0)+(u,0),0,0,0,(n-2)a+b+(m+1)d+(m-2)e) \Longrightarrow_{DM}$
$(n-2)a+2b+(m+2)d+(m-2)e+u$
...
$\Longrightarrow_{CRM} (e+u,0,0,0,(n-2)a+(m-1)b+(2m-1)d) \Longrightarrow_{\bigwedge \mathcal{A}_i}$
$((d,0)+(b,0)+(u,0),0,0,0,(n-2)a+(m-1)b+(2m-1)d) \Longrightarrow_{DM}$
$(n-2)a+mb+2md+u$

Return to Step 1

The computations goes on until the all the objects a are consumed from B.

Both steps have $(m+1)$ computations steps defined by the application of CRM followed by the application of the restricted direct product of MmA's and an application of DM. Hence after $2(m+1)$ computation steps, we consume $2a$ in order to produce $2md$. Finally we obtain the following multisets in the box:

1. If $n=2k+1$ is an odd integer, then we execute k pairs (**Step 1, Step 2**), then one additional **Step 1**, and finally the box contains $mnd+me+v$;
2. If $n=2k$ is an even integer, then we execute k pairs (**Step 1, Step 2**), and the box contains $mb+mnd+u$.

In both situations we get mn objects d.

Remark 1. An interesting feature of the membrane systems for multiplication presented in this paper is that the computation may continue after reaching a certain result, and so the system acts as a P transducer [6]. Thus if initially there are n (objects a) and m (objects b), the system evolves and produces $n \cdot m$ objects d. Afterwards, the user can inject more objects a and the system continues the computation obtaining the same result as if the objects a are present from the beginning. For example, if the user wishes to compute $(n+k) \cdot m$, it is enough to inject k objects a at any point of the computation.

5 Connecting Simple P Machines

Let us to consider a family of simple P machines indexed by a finite set T (of targets). We can define the *neighbourhood* of a simple P machine $\mathcal{M}^j$ $(j \in T)$ to be the set of all the simple P machines which can communicate with $\mathcal{M}^j$. In a P system environment (i.e. hierarchical system of simple P machines), by a neighbourhood of $\mathcal{M}^j$ we understand its parent, its children, and itself.

The output of a simple P machine $\mathcal{M}^j = (V, \bigwedge_{i=1}^{n_j} \mathcal{A}_i^j, O, B^j, CRM^j)$ is given by a multiset from $\mathbb{N}\langle V \times T \rangle$ of the form $w''^j + \sum_{i=1}^{n_j} k_i \cdot (y_i, tar_i)$; it can be viewed

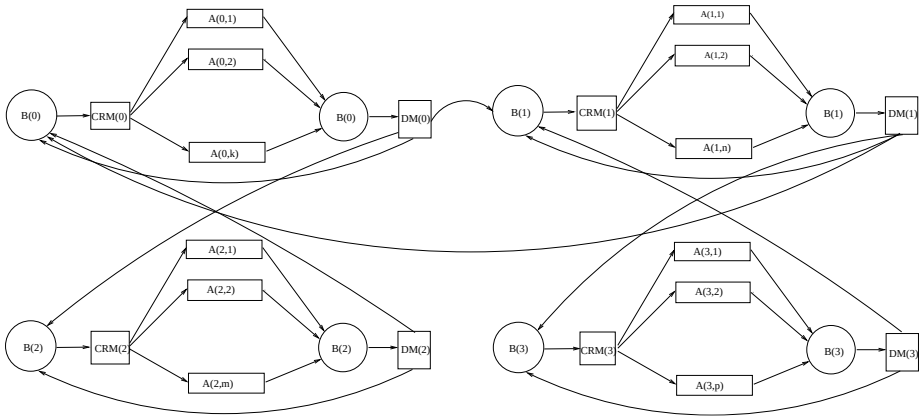
as a translation mapping from $\mathbb{N}\langle V\rangle$ into $\mathbb{N}\langle V \times T\rangle$. w''^j represents the updated content of the box; it contains the unprocessed multiset w'^j added with possible other multisets from its neighbourhood.

We show how we can connect a simple P machine with other simple P machines. This can be done by using a cascade-like product and the *distribution mapping* $DM^j : \mathbb{N}\langle V \times T\rangle \to (\mathbb{N}\langle V\rangle)^{m_j}$, where m_j is the number of simple P machines in the neighbourhood of $\mathcal{M}^j$.

$$DM^j(\sum_{i=1}^{m_j} k_i \cdot (y_i, tar_i)) = (w^1 + k_1 \cdot y_1, w^2 + k_2 \cdot y_2, \ldots, w^{m_j} + k_{m_j} \cdot y_{m_j}),$$

where w^i represents the box content of the simple P machine indexed by i.

An example of a graphical representation of a network composed of four simple P machines $\mathcal{M}^0$, $\mathcal{M}^1$, $\mathcal{M}^2$ and $\mathcal{M}^3$ is given by the following picture:



The simple P machine $\mathcal{M}^0$ (left upper corner) is the skin membrane of this network. We have two children represented by the simple P machines $\mathcal{M}^1$ (right upper corner) and $\mathcal{M}^2$ (left lower corner), and the simple P machine $\mathcal{M}^3$ (right lower corner) is the child of $\mathcal{M}^1$. As far as we can observe, $\mathcal{M}^3$ can communicate only with $\mathcal{M}^1$, $\mathcal{M}^1$ can communicate only with $\mathcal{M}^0$ and $\mathcal{M}^3$, and $\mathcal{M}^2$ can communicate only with $\mathcal{M}^0$. If we denote by $N(j)$ the set of indexes of the simple P Machines in the neighbourhood of the sPM indexed by j, in our diagram we have following sets: $N(0) = \{0, 1, 2\}$, $N(1) = \{0, 1, 3\}$, $N(2) = \{0, 2\}$, and $N(3) = \{1, 3\}$.

We give now an example of a P system with two membranes, and we describe its network of their corresponding simple P machines having the same computation. This P system also computes the multiplication nm. The P system (with two membranes) is given by

$$\Pi = (\{a, b, c, d, e, f\}, \{e\}, \emptyset, [_0[_1]_1]_0, a^n b^m c, \emptyset, (R_0, \rho_0), (\emptyset, \emptyset), 1), \text{ where}$$

– $R_0 = \{r_0 : ac \to f,\ r_1 : fb \to fd(e, 1),\ r_2 : f \to c,\ r_3 : d \to b\}$

– $\rho_0 = \{r_1 > r_2,\ r_1 > r_3\}$

We consider two simple P machines $\mathcal{M}^0 = (V, \bigwedge_{i=1}^{4} \mathcal{A}_i^0, O, B^0, CRM^0)$, and $\mathcal{M}^1 = (V, \emptyset, O, B^1, \emptyset)$ where the alphabets are $V = \{a, b, c, d, e, f\}$ and $O = \{(a,0), (b,0), (c,0), (d,0), (e,0), (f,0), (a,1), (b,1), (c,1), (d,1), (e,1), (f,1)\}$
$\mathcal{A}_1^0$ translates $a + c$ in $(f, 0)$, $\mathcal{A}_2^0$ translates $f + b$ in $(f,0) + (d,0) + (e,1)$, $\mathcal{A}_3^0$ translates f in $(c, 0)$, and $\mathcal{A}_4^0$ translates d in $(b, 0)$.
CRM^0 is defined as

$CRM^0(k_1a + k_2b + k_3c + k_4d + k_5e + k_6f) = (l_1(a + c), l_2(f + b), l_3f, l_4d, w')$, where

$l_1 = min\{k_1, k_2\}, l_2 = min\{k_2, k_6\}, l_3 = \begin{cases} 0 & l_2 \neq 0 \\ k_6 & l_2 = 0 \end{cases}, l_4 = \begin{cases} 0 & l_2 \neq 0 \\ k_4 & l_2 = 0 \end{cases}$

The distribution mapping is defined by

$DM(l_1(f,0) + l_2((f,0) + (d,0) + (e,1)) + l_3(d,0) + l_4(b,0)) =$

$(w' + (l_1 + l_2)f + (l_2 + l_3)d + l_4b, w" + l_2e)$, where w' and $w"$ represent the contents of B^0 and B^1, respectively, before applying distribution.

It can be verified that if we insert initially $na + mb + c$ in B^1, after doing the computation we get $c + md$ in B^1, and mne in B^2.

6 Conclusion and Further Work

According to our knowledge, we present for the first time an automaton corresponding faithfully to a membrane system with a single membrane and emphasizing on its maximal parallel evolution rules. We call simple P machine such an automaton. We give examples of simple P machines for both P systems with promoters and P systems with priorities. For each case we show that the P machines provide the same result as their corresponding P systems. We present a way of connecting simple P machines according to their communication rules, and give am example how the new resulting network corresponds to P systems with more than one membrane.

Considering the class $\mathcal{P}$ of the P systems involving priorities, promoters, activators, or catalysts, we claim that we can build a P machine having the same behaviour and result. In such a P machine

– every rule is modelled by a very simple Mealy multiset automata;
– various features involved by priority, promoters, as well as maximal parallel and nondeterministic application of the rules are captured by the control resource mappings;
– communications are modelled by the distribution mappings.

The converse of this claim is true because of the universality of P systems. However it is hard to have a constructive converse, namely an effective procedure of building a P systems having the same behaviour and results as a given P machine. In fact we use very simple MmA's incorporated in sPM to model a P system; they have only one generator (i.e., the left-hand side of the rule) for the set of tc-multisets (a cyclic semimodule). On the other hand, the definition

of a P machine does not restrict the Mealy multiset automata representing its parallel components to have only one generator for the set of all tc-multiset. This means that such a machine can change the "rules" after every computation step, for instance. These things are subject of further investigations.

Another direction of further research is given by the definition and the study of various aspects of machine-like theory (behaviour, bisimulation, composition). It is also interesting to continue the study of Mealy multiset automata by using linear algebra, detecting, for example, which are the links between the matrix associated with two different tc-multisets.

Automata theory has mainly a sequential nature, in contrast with P systems. Membrane systems represent abstract models inspired by the compartments of a cell. We try to connect the theory of membrane computing with the classic theory of (Mealy) automata. On the other hand, our machine has the capability to be highly adaptable, i.e. we can easily pass from strings to multisets and back, and so on. The inductive description is not able to distinguish between deterministic and nondeterministic automata. As we are more interested in their behaviour, we think to a co-inductive point of view (see [5]. It is worth to point out that while strings are of algebraic nature, multisets can be also viewed as their duals, so they have a coalgebraic nature.

References

1. F. Bernardini, V. Manca. Dynamical aspects of P systems. *BioSystems* vol.70, 85-93, 2002.
2. C. Bonchiş, G. Ciobanu, C. Izbaşa. Encodings and Arithmetic Operations in Membrane Computing. In *Theory and Applications of Models of Computation*, Lecture Notes in Computer Science vol.3959, Springer, 618–627, 2006.
3. L. Cardelli. Languages and notations for systems biology. *Unconventional Programming Paradigms*, Le Mount St.Michel, 2004.
4. G. Ciobanu, M. Gontineac. Mealy Multiset Automata. *International Journal of Foundations of Computer Science* vol.17, 111-126, 2006.
5. G. Ciobanu, M. Gontineac. Algebraic and Coalgebraic Aspects of Membrane Computing. In *Membrane Computing. WMC6*, Lecture Notes in Computer Science vol.3850, Springer, 181-198, 2006.
6. G. Ciobanu, Gh. Păun, Gh. Ştefănescu. P Transducers, *New Generation Computing* vol.24, 1-28, 2006.
7. E. Csuhaj-Varju, C. Martin-Vide, V. Mitrana. Multiset Automata. In *Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View*, Lecture Notes in Computer Science vol.2235, Springer, 69-83, 2001.
8. E. Csuhaj-Varju, G. Vaszil. P automata or purely communicating accepting P systems. In *Membrane Computing. WMC-CdeA 2002*, Lecture Notes in Computer Science vol. 2597, Springer, 219–233, 2003.
9. M. Oswald. *P Automata*, PhD Thesis, Technical University Vienna, 2004.
10. Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.