Proceedings of the International Workshop, Automata for Cellular and Molecular Computing, MTA SZTAKI, Budapest, pages 52 - 63, 2007.

Networks of Mealy Multiset Automata

Gabriel Ciobanu and Mihai Gontineac

"A.I.Cuza" University of Iaşi, Romania
Faculty of Computer Science and Faculty of Mathematics
gabriel@info.uaic.ro, gonti@uaic.ro

Abstract

We introduce the networks of Mealy multiset automata, and study their computational power. The networks of Mealy multiset automata are computationally complete.

1 Learning from Molecular Biology

Systems biology represents a new cross-disciplinary approach in biology which has only recently been made possible by advances in computer science and technology. As it is mentioned in [10], it involves the application of experimental, theoretical, and modelling techniques to the study of biological organisms at all levels. Adding new abstractions, discrete models and methods able to help our understanding of the biological phenomena, systems biology may provide predictive power, useful classifications, new paradigms in computing and new perspectives on the dynamics of various biological systems.

Recent promising work [1] employs automata theory as an efficient tool of describing and controlling gene expression (a small automaton is encoded by DNA strands and then it is used in logical control of gene expression).

In [2], we present a way of interaction between gene machine and protein machine, namely the process of making proteins, in abstract terms of Mealy automata, transformation semigroups and abstract operations.

The Mealy automaton proposed as a formal model of the genetic message translation is a minimal one that accepts the mRNA messages and terminates the translation process (according to [9], there is no appropriate formalism for the process of translation).

However molecular biology "deals" not only with sequences, but also with multisets. The biological cells are "smart" enough to put together at work sequences and multisets of atoms and molecules, so if we try to get models from their functioning, we should not restrict ourselves to dealing only with sequential machines (like classical automata). To deal with multisets, the main approach is given by membrane systems [11]. There is also introduced and studied an automaton-like machine to work with multisets [7], i.e. *multiset automaton*. At a first glance, it seems that they are nothing else but weighted automata with weights in the semiring of positive integers. In [8] it is proven that such automata (weighted automata with

weights in the semiring of positive integers) have the same power as finite automata (accept only regular languages). In fact, a careful reader should remark that a multiset automaton is not a sequential machine, and it is not working with sequences of multisets as in the case of weighted automata. A multiset automaton accepts, together with a sequence of multisets, an entire class, namely the class of that sequence obtained by "abelianization" (as an example, together with the sequence, say *aab*, it accepts *aba* and *baa*). In this manner, multiset automata become very powerful (see [7], for details). Mealy multiset automata, [3], can be viewed as the corresponding Mealy machine. We study some of their (co)algebraic properties in [3] and [4] and we connect these properties with various aspects of their behaviour. In [5], we organize them in a *P machine*, in order to simulate a P system. However, the biological systems are not always organized in a hierarchical manner. This means that we also have to organize sets of Mealy multiset automata in networks. In order to obtain the computing power for networks of such automata, we relate them to neural P systems, proving that networks of Mealy multiset automata are computationally complete.

2 Networks of Mealy Multiset Automata

In order to define networks of Mealy multiset automata, we can connect these automata in many ways, having both parallel and serial connections. In [3] we define the *restricted direct product* of MmA for the parallel case, and the *cascade product* for a serial connection.

2.1 Multisets

The evolution rules performed by membranes are multiset operators; the multiset operators are associative and commutative, and have also an identity.

A *multiset* over an alphabet $A = \{a_1, a_2, ..., a_n\}$ is a mapping $\alpha : A \to \mathbb{N}$. It can be represented by $\{(a_1, \alpha(a_1)), (a_2, \alpha(a_2)), ..., (a_n,\alpha(a_n))\}$. Inspired from formal power polynomials, we denote by $\mathbb{N}\langle A\rangle = \{\alpha : A \to \mathbb{N} \mid \alpha \text{ is a mapping}\}$ the set of all multisets on A . The structure of $\mathbb{N}\langle A\rangle$ is mainly an additive one, since we add multiplicities of appearance (in fact, it is induced by the addition in $\mathbb{N}$). This argument is sustained also by the chemical reactions that are the base of the biological modelling. They provide a notation for defining the way a biological system evolves.

If $\alpha, \beta \in \mathbb{N}\langle A\rangle$, then their *sum* is the multiset $(\alpha + \beta) : A \to \mathbb{N}$ defined by $(\alpha + \beta)(a_i) = \alpha(a_i) + \beta(a_i)$, $i = \overline{1, n}$. Moreover, if we consider the letters from A as multisets, i.e. a_i is given by μ_{a_i}, where $\mu_{a_i} : A \to \mathbb{N}$, $\mu_{a_i}(a_i) = 1$ and $\mu_{a_i}(a_j) = 0$ for all $j \neq i$, then we can express every multiset $\alpha \in \mathbb{N}\langle A\rangle$ as a linear combination of a_i, i.e. $\alpha = \sum_{i=1}^{n} \alpha(a_i) \cdot a_i$ (see also [3]). The *length* of a multiset α, denoted by $|\alpha|$, is defined by $|\alpha| = \sum_{i=1}^{n} \alpha(a_i)$.

We can define an external operation $m\alpha = \sum_{i=1}^{n}(m\alpha(a_i)) \cdot a_i$, for all $m \in \mathbb{N}$ and $\alpha \in \mathbb{N}\langle A\rangle$.

Proposition 1 *$\mathbb{N}\langle A\rangle$ has a structure of $\mathbb{N}$-semimodule (semimodule over the semiring of positive integers).*

2.2 Mealy multiset automata

Roughly speaking, a *Mealy multiset automaton* (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 multiset into a multiset over an output alphabet. We have a detection head that detects whether or not a given multiset appears in the multiset available in the box. The multiset is removed from the box whenever it is detected, and the automaton inserts a multiset over the output alphabet. The output alphabet should be different from the input alphabet; if they are the same, we mark the output symbols just to make different the output and input alphabet. In this way the output symbols cannot be viewed by the detection head. This automaton stops when no further move is possible. We say that the sub-multiset read by the head was translated to a multiset over the output alphabet. We give here only the definitions and the properties that we need for networks of MmA. For more informations see [3] or [4].

From the formal point of view, a *Mealy multiset automaton* is a construct $\mathcal{A} = (Q, V, O, f, g, q_0)$ where

1. Q is a finite set, the set of *states*;
2. $q_0 \in Q$ is special state, which is both initial and final;
3. V is a finite set of objects, the *input alphabet*;
4. O is a finite set of objects, the *output alphabet*, such that $O \cap V = \emptyset$;
5. $f : Q \times \mathbb{N}\langle V\rangle \to \mathcal{P}(Q)$ is the *state-transition (partial) mapping*;
6. $g : Q \times \mathbb{N}\langle V\rangle \to \mathcal{P}(\mathbb{N}\langle O\rangle)$ is the *output (partial) mapping.*

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

An MmA $\mathcal{A}$ receives a multiset in its box, and processing this multiset it passes 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 those configurations) if there is $a \subseteq \alpha$ such that $s \in f(q,a), \bar{b} \in g(q,a)$. We denote this by $(q, \alpha, \bar{\beta}) \vdash (s, \alpha - a, \bar{\beta} + \bar{b})$, and 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.

Behaviour is often appropriately viewed as consisting of both dynamics and observations, which have to do with change of states and partial access to states, respectively. The main advantage of an MmA is that it has an output function that can play the main role in observability, i.e. we do not have to construct an other machine to describe the MmA' s behaviour.

Definition 1 *Let $\mathcal{A} = (Q, V, O, f, g)$ be a Mealy multiset automaton. The* general behaviour *of a state $q \in Q$ is a function* $\mathbf{beh}(q)$ *assigning to every multiset $\alpha \in \mathbb{N}\langle V\rangle$ the output multiset obtained after consuming α starting from q.*

When talking about the behaviour, we consider a specific order of consuming multisets, i.e. in terms of strings of multisets.

Definition 2 *Let $\mathcal{A} = (Q, V, O, f, g)$ be a Mealy multiset automaton. The* sequential behaviour *of a state $q \in Q$ is a function* $\mathbf{seqbeh}(q)$ *that assigns to every multiset $\alpha \in \mathbb{N}\langle V\rangle$ all the sequences of the output multisets obtained after consuming α starting from q.*

Example 1 *Suppose that we have the following sequence of transitions $(q, \alpha, \varepsilon) \vdash (q_1, \alpha - a_1, b_1) \vdash (q_2, \alpha - a_1 - a_2, b_1 + b_2) \vdash \ldots \vdash (q_n, \alpha - a_1 - \ldots - a_n, b_1 + \ldots + b_n)$ and MmA stops. Then $\mathbf{beh}(q)(\alpha) = b_1 + \ldots + b_n$ and $\mathbf{seqbeh}(q)(\alpha) \ni b_1 \ldots b_n$. Moreover, $b_1 + \ldots + b_n$ belongs to $\mathbb{N}\langle O\rangle$, while $b_1 \ldots b_n$ belongs to $(\mathbb{N}\langle O\rangle)^*$.*

Consider the canonical inclusion $i : \mathbb{N}\langle O\rangle \to (\mathbb{N}\langle AO\rangle)^*$ and the identity map $id : \mathbb{N}(O) \to \mathbb{N}(O)$. By the universal property of the free monoid, we know that there exists a unique homomorphism of monoids $\mathbf{I}_O : (\mathbb{N}\langle O\rangle)^* \to \mathbb{N}\langle O\rangle$ defined by $\mathbf{I}_O(b_1 \ldots b_n) = b_1 + \ldots + b_n$ such that $\mathbf{I}_O \circ i = id$. Since id is onto, it follows that $\mathbf{I}$ is onto, and so, applying the isomorphism theorem for monoids, we obtain that $(\mathbb{N}\langle O\rangle)^*/ker\mathbf{I}_O \backsimeq \mathbb{N}\langle O\rangle$. Moreover, for all the states q of a Mealy multiset automaton we have

$$\mathbf{I}_O \circ \mathbf{seqbeh}(q) = \mathbf{beh}(q).$$

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

- $f((q_1, q_2, \ldots, q_n), a) = (f_1(q_1, a), f_2(q_2, a), \ldots, f_n(q_n, a))$,
- $g((q_1, q_2, \ldots, q_n), a) = (g_1(q_1, a), g_2(q_2, a), \ldots, g_n(q_n, a))$,
- box of $\mathcal{A}$ is the disjoint union $\bigsqcup_{i=1}^{n} B_i$ of $\{B_i \mid i = \overline{1,n}\}$,
- a *configuration* of $\mathcal{A}$ is a triple $(q, \alpha, \bar{\beta})$, where $q = (q_1, q_2, \ldots, q_n)$, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, and $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2, \ldots, \bar{\beta}_n)$,
- the *transition relation* of $\mathcal{A}$: $(q, \alpha, \bar{\beta}) \vdash (s, \alpha - a, \bar{\beta} + \bar{b})$ iff $s_i \in f_i(q_i, a_i)$ and $\bar{b}_i \in g_i(q_i, a_i)$ for all $i \in \overline{1,n}$.

The *cascade product* is useful to describe a serial connection, and provide also some results in decompositions of such machines in irreducible ones.

Let $\mathcal{A} = (Q, V, O, f, g)$, $\mathcal{A}' = (Q', V', O', f', g')$ be two Mealy multiset automata. In order to connect them, we need a multiset mapping linking the output of one of them to the input of the other. This can be done using a $\mathbb{N}$-homomorphism from $\mathbb{N}\langle O'\rangle$ to $\mathbb{N}\langle V\rangle$ (this homomorphism can be obtained by using a mapping from O' to V). We denote by $\Lambda : \mathbb{N}\langle O'\rangle \to \mathbb{N}\langle V\rangle$ this homomorphism. Then we can define a mapping $\Omega : Q' \times \mathbb{N}\langle V'\rangle \to \mathbb{N}\langle V\rangle$ by $\Omega(q', a') = \Lambda(g'(q', a'))$.

- This mapping gives us *the cascade product induced by* Ω:

$$\mathcal{A}\Omega\mathcal{A}' = (Q \times Q', V', O, f^{\Omega}, g^{\Omega})$$

  where $f^{\Omega}((q, q'), a') = (f(q, \Omega(q', a')), f'(q', a'))$, $g^{\Omega}((q, q'), a') = g(q, \Omega(q', a'))$, for all $a' \in \mathbb{N}\langle V'\rangle, (q, q') \in Q \times Q'$.

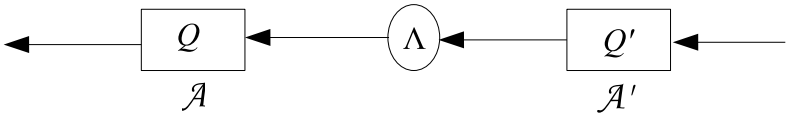
- The transition relation becomes $((q, q'), \alpha', \bar{\beta}) \vdash ((s, s'), \alpha' - a', \bar{\beta} + \bar{b})$ if there is $a' \subseteq \alpha'$ such that $(s, s') = f^{\Omega}((q, q'), a')$ and $\bar{b} = g^{\Omega}((q, q'), a')$, where $a', \alpha' \in \mathbb{N}\langle V'\rangle$, $(q, q') \in Q \times Q'$, and $\bar{\beta} \in \mathbb{N}\langle O\rangle$.

We can alternatively define the transition relation by

$$((q, q'), \alpha', \bar{\beta}) \vdash ((s, s'), \alpha' - a', \bar{\beta} + \bar{b}),$$

if there is $a' \subseteq \alpha'$ such that $s{=}f(q, \Lambda(g'(q', a')))$, $s'{=}f'(q', a')$, $\bar{b}{=}g(q, \Lambda(g'(q', a')))$, where $a', \alpha' \in \mathbb{N}\langle V'\rangle$, $(q, q') \in Q \times Q'$, $\bar{\beta} \in \mathbb{N}\langle O\rangle$.

The graphical representation of the cascade product is given in the following figure:



In order to obtain the behaviour of a network of MmA's, we should also consider the behaviour of the cascade product. Roughly speaking, the two types of behaviour depends mainly on the corresponding behaviours of $\mathcal{A}'$. On the other hand, when we have a cascade product, the observable part is strongly connected with the observations that could be made after we pass through $\mathcal{A}$. We may also emphasize the important role played by the connection homomorphism given by Λ.

Theorem 2 *Let $\mathcal{A} = (Q, V, O, f, g)$, $\mathcal{A}' = (Q', V', O', f', g')$ be two MmA's, $\mathcal{A}\Omega\mathcal{A}'$ their cascade product, and (q, q') a state of this product. The behaviour of (q, q') is* $\mathbf{beh}((q, q')) = \mathbf{beh}(q) \circ \Lambda \circ \mathbf{beh}(q')$.

If we want to get the sequential behaviour starting from $\mathbf{beh}((q, q')) = \mathbf{beh}(q) \circ \Lambda \circ \mathbf{beh}(q')$, then $(\mathbf{I}, \mathbf{I}') \circ \mathbf{seqbeh}((q, q')) = (\mathbf{I} \circ \mathbf{seqbeh}(q)) \circ \Lambda \circ (\mathbf{I}' \circ \mathbf{seqbeh}(q'))$.

Other properties of MmA's, their behaviours and bisimulation relations are presented in [3] and [4].

2.3 Networks of automata

The formal description of a network of Mealy multiset automata is not intuitive. On the other hand, these networks could be very powerful, so we think that they deserve our attention. We can consider several variants of such networks. Some of them can have no inter-communication and, in this case, the network is, in fact, a bigger MmA. The same remark can be done if we have only MmA connected in a serial manner, without any ramifications. The case that we consider in this paper is inspired by the definition of neural P systems (nP systems). Neural P systems are defined in [11] as a computing model inspired by the network of cells. Each cell has a finite state memory, and processes multisets of symbol (impulses); it can send some impulses (called excitations) to the neighbouring cells. It is proved that such networks are rather powerful: they can simulate Turing machines using a small number of cells, every cell having in a small number of states. It is also proved that, in appropriate organization, such a network can solve in linear time the Hamiltonian Path Problem.

We consider a set of MmA that can communicate by means of some *communication channels*. All of them have the same input alphabet V, and their boxes contain an input multiset over V (they can also have an empty multiset ε as input). The output alphabet has a "real" part O of output alphabet, and a "specific" part used for communication. The specific part is, in fact, a Cartesian product between the input alphabet V and the set of targets T (the set of the indexes of the MmA forming the net). We can also have a special MmA to collect in its box the result of the computation (i.e. a multiset over O) for such a network. Alternatively, we can consider as result of the computation the tuple of multisets obtained in the box of every MmA of the net.

Definition 3 *A* network of Mealy multiset automata *(shortly, nMmA) is a construct* $\mathcal{N} = (V, O, \{\mathcal{A}_i\}_{i=\overline{1,n}}, \{\Lambda_i\}_{i=\overline{1,n}}, B)$ *where:*

- $V = \{a_1, a_2, \ldots, a_m\}$ *is a finite set of objects, the* input alphabet*;*
- O *is the* output alphabet *such that* $O \cap V = \emptyset$*;*
- $\mathcal{A}_i = (Q_i, V, \overline{O}, f_i, g_i, s_{0,i})$ *are MmA's connected in the network. Their output alphabets are of the form* $\overline{O} = O \cup (V \times T)$*, where* $T = \{1, 2, \ldots, n\}$*;*
- B *is a box where* $\mathcal{N}$ *"receives" the output multiset. Depending on features that we consider for the net,* B *can be a specific box of a specific MmA in the network, or* B *can be the Cartesian product of all the boxes.*
- $\Lambda_i : \mathbb{N}\langle\overline{O}\rangle \to (\mathbb{N}\langle O\rangle \cup \mathbb{N}\langle V\rangle)^n$ *are the* communication mappings *associated to all the* $\mathcal{A}_i$*,* $i \in T$*.*

A *computation* starts with some input multisets $w_{0,i}$ in the boxes of the MmA's that are in their initial states, $s_{0,i}$; then we have a *big step* given by a translation (made by the MmA's, in fact by their restricted direct product $\bigwedge_{i=1}^{n} \mathcal{A}_i$) and a communication (done by $\{\Lambda_i\}$) - a kind of "parallel cascade product", since every MmA is in cascade with the restricted direct product of itself and the other MmA.

A *configuration* of the network is of the form (s, w), where $s = (s_1, s_2, \ldots, s_n)$ with $s_i \in Q_i$ is the *global state*, and $w = (w_1, \ldots, w_n)$ where $w_i \in \mathbb{N}\langle O\rangle \cup \mathbb{N}\langle V\rangle$.

A *transition between configurations* is denoted by $(s, w) \vdash (s', w')$ and is defined in the following manner:

$s' = (s'_1, s'_2, \ldots, s'_n)$, where $s'_i \in f_i(s_i, a_i)$ with $a_i \in \mathbb{N}\langle V\rangle$; we allow some of the a's to be ε if in the corresponding MmA there is no transition.

$w' = \Lambda_1(b_1) + \Lambda_2(b_2) + \ldots + \Lambda_n(b_n) + (w_1 - a_1, w_2 - a_2, \ldots, w_n - a_n)$, where $b_i \in g_i(s_i, a_i)$.

A network of MmA can be used in various modes. We can use it as a generative system, looking to the number of output objects that we find in the boxes (without considering the final state for the MmA). It can be used also to compute functions from $\mathbb{N}\langle V\rangle$ to $\mathbb{N}\langle O\rangle$.

An example of such a network used as a generative system could clarify these aspects:

Example 2 *Let*

$$\mathcal{N} = (V, O, \{\mathcal{A}_i\}_{i=\overline{1,3}}, \{\Lambda_i\}_{i=\overline{1,3}}, B_1)$$

where:

- $V = \{a\}$ is the input alphabet;
- $O = \{b\}$ is the output alphabet $b \neq a$;
- $\mathcal{A}_i = (\{s_i\}, V, \overline{O}, f_i, g_i, s_i)$, are the MmA's connected in the network. Their output alphabet is $\overline{O} = \{b, (a,1), (a,2), (a,3)\}$
- B_1 is the box where $\mathcal{N}$ "receives" the output multiset.
- $\Lambda_i : \mathbb{N}\langle\overline{O}\rangle \to (\mathbb{N}\langle O\rangle \cup \mathbb{N}\langle V\rangle)^3$ are the communication mappings associated to all the $\mathcal{A}_i$, $i \in T = \{1, 2, 3\}$.

We describe now the mappings. The transition mappings are

- $f_i(s_i, a) = s_i, i = \overline{1,3}$.

The output mappings:

- $g_1(s_1, a) \in \{b, (a,2) + (a,3)\}$, so is a nondeterministic mapping;
- $g_2(s_2, a) = (a, 1)$;
- $g_3(s_3, a) = (a, 1)$.

The communication mappings:

- $\Lambda_1(nb + k_1(a,1) + k_2(a,2) + k_3(a,3)) = (nb + k_1 a,\ k_2 a,\ k_3 a))$;
- $\Lambda_2(nb + k_1(a,1) + k_2(a,2) + k_3(a,3)) = (k_1 a,\ \varepsilon,\ \varepsilon))$;

- $\Lambda_3(nb + k_1(a,1) + k_2(a,2) + k_3(a,3)) = (k_1a,\ \varepsilon,\ \varepsilon)$.

Since $\mathcal{A}_1$ has a nondeterministic output mapping, the behaviour of our network is nondeterministic. We denote the global state by $s = (s_1,\ s_2,\ s_3)$. We start our computation from $(s, (a, \varepsilon, \varepsilon))$ Applying the restricted direct product we can obtain $(s, (b,\ \varepsilon,\ \varepsilon))$ or $(s, (\varepsilon, a, a))$.

In the first case we obtain one b, so we generate 1. In the second case, the computation continues with communication, and we obtain $(s, (a+a,\ \varepsilon,\ \varepsilon) = (s, (2a,\ \varepsilon,\ \varepsilon)$, and, again, we have various possibilities to choose. Anyway, it should be clear now that we can generate any number of b's, so $\mathcal{N}$ can generate every positive integer.

In order to study the computational power of nMmA, we are trying to simulate neural-like P systems. To be more specific, we try to simulate the neural P systems working in minimal mode and replicative manner. To keep the paper self-contained, we remember some facts about neural P systems and adapt the notations from [11].

3 Neural P Systems

The former tissue P systems were called neural-like P systems in [11]. We start with the classical definition, and later we adapt the notation to our needs. We consider a class of networks of membranes inspired by the way the neurons cooperate to process impulses in the complex net established by synapses. A possible model of this symbol processing machinery can be given by a network of membranes, each of them containing a multiset of objects and a state according to which the objects are processed. The membranes can communicate along "axons" channels. We make some minor modifications to the original notations, having in mind that in the Mealy multiset automata we distinguish between multisets and strings that could represent them (since we can deal with two kinds of behaviours, a global one and a sequential one). We also restrict our presentation of neural P systems working in *minimal mode and replicative manner.*

Definition 4 *A* neural P system *(nP system) of degree $m \geq 1$ is a construct*

$$\Pi = (V, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, i_{out}),$$

where

1. *V is a finite non-empty alphabet (of* objects*);*

2. *$syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ (*synapses *among cells);*

3. *$i_{out} \in \{1, 2, \ldots, m\}$ indicates the* output cell*; we can put $i_{out} = 1$;*

4. *$\sigma_1, \sigma_2, \ldots, \sigma_m$ are* cells *of the form $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, R_i), 1 \leq i \leq m$,*

where:

- *Q_i is a finite set (of states);*

- *R_i is a finite set of rules of the form $sw \to s'(x + y_{go} + z_{out})$, where $s, s' \in Q_i$, $w, x \in \mathbb{N}\langle V\rangle$, $y_{go} \in \mathbb{N}\langle V \times \{go\}\rangle$, $z_{out} \in \mathbb{N}\langle V \times \{out\}\rangle$, with the restriction that $z_{out} = \varepsilon$ for all i different from 1.*

The objects that appears in the left hand multiset w of the rule $sw \to s'w'$ are called *impulses*, while those from w' are called *excitations*.

Such a system is called to be *cooperative* if it contains at least one rule $sw \to s'w'$ such that $|w| > 1$, and *non-cooperative* in the opposite case.

An m-tuple of the form $(s_1w_1, s_2w_2, \ldots, s_mw_m)$ is called a *configuration* of Π. Using the rules defined above, we can define *transitions* among the configurations of the system. To this end, there are considered three modes of processing the *impulse-objects* and three modes of transmitting *excitation-objects* from one cell to another one. As we already mentioned, we restrict ourselves to the *minimal processing mode.*

Notation: $V_{go} = \{(a; go) \mid a \in V\}$, $V_{out} = \{(a; out) \mid a \in V\}$, and $V_{tot} = V \cup V_{go} \cup V_{out}$. For $s, s' \in Q_i$, $x \in \mathbb{N}\langle V\rangle$ and $y \in \mathbb{N}\langle V_{tot}\rangle$, we write $sx \Rightarrow_{min} s'y$ iff $sw \to s'w' \in R_i$, $w \subseteq x$ and $y = (x - w) \cup w'$. In this case, only one occurrence of the multiset from the left-hand side of a rule is processed, being replaced by the multiset from the right-hand of the rule, and at the same time changing the state of the cell.

We also write $sx \Rightarrow_{min} sx$ for $s \in Q_i$ and $x \in \mathbb{N}\langle V\rangle$ whenever there is no rule $sw \to s'w' \in R_i$ such that $w \subseteq x$. This encodes the case when a cell cannot process the current objects in a given state (it can be "unblocked" after receiving new impulses from the cells which are active and can send objects to it).

Now, recall that the multiset w' from a rule $sw \to s'w'$ contains symbols from V, but also symbols of the form (a, go) (or, in the case of the cell 1, of the form (a, out)). Such symbols are sent to the cells related by synapses to the cell σ_i where the rule $sw \to s'w'$ is applied, according to various manners of communication. As we already mentioned, we choose the *replicative manner*, i.e. each symbol a from (a, go) appearing in w', it is sent to each of the cells σ_j such that $(i; j) \in syn$.

In order to formally define the transition among the configurations of Π, some further notations are needed. For a multiset w over V_{tot}, we consider the projections on V, V_{go} and V_{out}, namely $pr_V(w); pr_{V_{go}}(w)$, and $pr_{V_{out}}(w)$ (see [11] for details). For a node i in the graph defined by syn, the ancestors and the successors of node i are denoted by $anc(i) = \{j \mid (j, i) \in syn\}$ and $succ(i) = \{j \mid (i, j) \in syn\}$, respectively.

Each transition lasts one time unit, and the network is synchronized: a global clock define the passage of time for all the cells.

For two configurations $C_1 = (s_1w_1, \ldots, s_mw_m)$ and $C_2 = (s'_1w''_1, \ldots, s'_mw''_m)$ we write $C_1 \Rightarrow C_2$ if there are $w'_1, \ldots, w'_m$ in $\mathbb{N}\langle V_{tot}\rangle$ such that

$$s_iw_i \Rightarrow s'_iw'_i, 1 \leq i \leq m,$$

and

$$w''_i = pr_V(w'_i) + \sum_{j \in anc(i)} pr_{V_{go}}(w'_j).$$

Obviously, objects are always sent to a cell i only from its ancestors, namely from cells j such that a direct synapse exists from j to i. In the case of the cell 1, we

remove from w_1' all the symbols $a \in V$ which appear in w_1' in the form (a, out). If during a transition a cell does nothing (no rule is applicable to the available multiset of objects in the current state), then the cell waits until new objects are sent to it from its ancestor cells.

A sequence of transitions among the configurations of Π is called a *computation* of Π. A computation ending in a configuration where no rule in no cell can be used is called a halting computation. The result of a halting computation is the number of objects in the output cell 1 (or sent to the environment from the output cell 1). We denote by $N(\Pi)$ the set of all natural numbers computed in this way by a system Π. We denote by $NOnP_{m,r}(coo)$ the family of sets $N(\Pi)$ computed by all cooperative neural-like P systems with at most $m \geq 1$ cells, each of them using at most $r \geq 1$ states. When non-cooperative systems are used, we write $NOnP_{m,r}(ncoo)$ for the corresponding family of sets $N(\Pi)$.

3.1 Computational power

We denote by NRE the family of Turing computable sets of natural numbers.

Following [11], we mention that the minimal mode of using the rules turns out to be computationally universal. If we consider the apparently weak neural-like P systems, then the fact that we obtain universality even in the non-cooperative case when using the mode *min* of applying the rules is rather unexpected. The same result holds true also when using cooperative rules. Among the results presented in [11] we mention here only those for minimal mode and for replicative manner.

Theorem 3 $NOnP_{2,5}(ncoo) = NRE$.

For the cooperative rules, the number of states can be decreased.

Theorem 4 $NOnP_{2,2}(coo) = NRE$.

4 Universality of the Networks

In order to obtain the generative power of a network of MmA, we give the following result.

Theorem 5 *Any nP System working in min mode and replicative manner can be simulated by a network of MmA (possibly nondeterministic).*

Proof. Let $\Pi = (V, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, 1)$ be an nP system with its components described as in the previous section. We remind that $\sigma_1, \sigma_2, \ldots, \sigma_m$ are cells of the form $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, R_i)$ $(1 \leq i \leq m)$, where Q_i is a finite set (of *states*) and R_i is a finite set of rules of the form $sw \to s'(x + y_{go} + z_{out})$ with $s, s' \in Q_i$, $w, x \in \mathbb{N}\langle V \rangle$, $y_{go} \in \mathbb{N}\langle V \times \{go\} \rangle$, $z_{out} \in \mathbb{N}\langle V \times \{out\} \rangle$ (with the restriction that $z_{out} = \varepsilon$ for all i different from 1).

We can build a nMmA $\mathcal{N} = (V, O, \{\mathcal{A}_i\}_{i=\overline{1,m}}, \{\Lambda_i\}_{i=\overline{1,m}}, B_1)$ where:

- the output alphabet O is V_{out};

- $\mathcal{A}_i = (Q_i, V, \overline{O}, f_i, g_i, s_{0,i})$ is the MmA simulating the activity of cell σ_i. The output alphabets are of the form $\overline{O} = O \cup (V \times T)$, where $T = \{1, 2, \ldots, m\}$;
- B_1 is the box where $\mathcal{N}$ collects the output multisets;
- $\Lambda_i : \mathbb{N}\langle\overline{O}\rangle \to (\mathbb{N}\langle O\rangle \cup \mathbb{N}\langle V\rangle)^n$ are the communication mappings associated to $\mathcal{A}_i$, $i \in T$.

Consider a rule $sw \to s'(x + y_{go} + z_{out})$ from R_i. We can simulate this rule with f_i and g_i by defining them in the following manner:

- $f_i(s, w) = s'$;
- $g_i(s, w) = (z_{out} + x + k_1(y, 1) + k_2(y, 2) + \ldots + k_m(y, m))$,

with the following restrictions in g_i:

- if $i \neq 1$, then $z_{out} = \varepsilon$;
- if there is no synapse from σ_i to σ_j, we define $k_j = 0$, else $k_j = 1$.

In this manner we can also simulate the replicative manner of applying the rules, since y is marked to be sent to all the cells having synapse from σ_i. It is easy to see that we have a transition $(s_1w_1, \ldots, s_mw_m) \Rightarrow (s'_1w''_1, \ldots, s'_mw''_m)$ in Π if and only if $((s_1, s_2, \ldots, s_n), (w_1, \ldots, w_n)) \vdash ((s'_1, s'_2, \ldots, s'_n), (w''_1, \ldots, w''_n))$. □

As an immediate consequence of this result we get the following

Theorem 6 *Nondeterministic networks of Mealy multiset automata are universal.*

Proof. We already know that $NOnP_{2,2}(coo) = NRE$. Applying the previous theorem we obtain that the generative power of a nondeterministic network of MmA is NRE.

Moreover, according to the proof for $NOnP_{2,2}(coo) = NRE$ ([11], page 261), the universality is obtained for a network with two Mealy multiset automata, the first one having two states, while the second one has only one state. □

Therefore a network of Mealy multiset automata is able to simulate Turing machines, and so it is computationally complete. The number of cells and states sufficient to characterize the power of Turing machines is rather small.

P systems are simulated on a cluster (network) of computers [6]. It would be interesting to see whether such an implementation can be related to the network of Mealy multiset automata.

References

[1] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, E. Shapiro. An autonomous molecular computer for logical control of gene expression. *Nature* 429:423-429, 2004.

[2] G. Ciobanu, M. Gontineac. An Automata Description of the Genetic Message Translation. *Fundamenta Informaticae*, 64:93-107, 2005.

[3] G. Ciobanu, M. Gontineac. Mealy Multiset Automata. *International Journal of Foundations of Computer Science*, 17(1):111-126, 2006.

[4] G. Ciobanu, M. Gontineac. Algebraic and Coalgebraic Aspects of Membrane Computing. In volume 3850 of *Lecture Notes in Computer Science*, pages 181-198, Springer-Verlag, 2006.

[5] G. Ciobanu, M. Gontineac. P Machines: An Automata Approach to Membrane Computing. In volume 4361 of *Lecture Notes in Computer Science*, pages 314-329, Springer-Verlag, 2006.

[6] G. Ciobanu, W. Guo. P Systems Running on a Cluster of Computers. In volume 2933 of*Lecture Notes in Computer Science*, pages 123-139, Springer-Verlag, 2004.

[7] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana. Multiset Automata. In *Multiset Processing*, volume 2235 of *Lecture Notes in Computer Science*, pages 69-83, Springer-Verlag, 2001.

[8] S. Eilenberg. *Automata, Languages and Machines*. Vol. A, Academic Press, 1976.

[9] H. de Jong. Modelling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology*, 9:67-103, 2002.

[10] H. Kitano. Computational Systems Biology. *Nature*, 420:206-210, 2002.

[11] Gh. Păun. *Membrane Computing: An Introduction*. Springer-Verlag, 2002.