Natural Computing Modelling of the Polynomial Space Turing Machines

Bogdan Aman and Gabriel Ciobanu

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.11, 700506 Iaşi, Romania
baman@iit.tuiasi.ro, gabriel@info.uaic.ro

Abstract. In this paper we consider a bio-inspired description of the polynomial space Turing machines. For this purpose we use membrane computing, a formalism inspired by the way living cells are working. We define and use logarithmic space systems with active membranes, employing a binary representation in order to encode the positions on the Turing machine tape.

Keywords. Natural computing, membrane computing, Turing machines.
Key Terms. MathematicalModel, Research.

1 Introduction

Membrane computing is a branch of the natural computing inspired by the architecture and behaviour of living cells. Various classes of membrane systems (also called P systems) have been defined in [12], together with their connections to other computational models. Membrane systems are characterized by three features: (i) a membrane structure consisting of a hierarchy of membranes (which are either disjoint or nested), with an unique top membrane called the *skin*; (ii) multisets of objects associated with membranes; (iii) rules for processing the objects and membranes. When membrane systems are seen as computing devices, two main research directions are usually considered: computational power in comparison with the classical notion of Turing computability (e.g., [2]), and efficiency in algorithmically solving NP-complete problems in polynomial time (e.g., [3]). Such efficient algorithms are obtained by trading space for time, with the space grown exponentially in a linear time by means of bio-inspired operations (e.g., membrane division). Thus, membrane systems define classes of computing devices which are both powerful and efficient.

Related to the investigations of these research directions, there have been studied several applications of these systems; among them, modelling of various biological phenomena and the complexity and emergent properties of such systems presented in [7]. In [4] it is presented the detailed functioning of the sodium-potassium pump, while in [1] it is described and analyzed the immune system in the formal framework of P systems.

In this paper we consider P systems with active membranes [11], and show that they provide an interesting simulation of polynomial space Turing machines by using only logarithmic space and a polynomial number of read-only input objects.

2 Preliminaries

We consider P systems with active membranes extended with an input alphabet, and such that the input objects cannot be created during the evolution [14]. The original definition also includes dissolution and division rules, rules that are not needed here. The version used in this paper is similar to evolution-communication P systems used in [6] with additional read-only input objects and polarities.

Definition 1. *A* P system with active membranes and input objects *is a tuple* $\Pi = (\Gamma, \Delta, \Lambda, \mu; w_1, \ldots, w_d, R)$*, where:*

- $d \geq 1$ *is the* initial degree*;*
- Γ *is a finite non-empty alphabet of* objects*;*
- Δ *is an* input alphabet *of objects such that* $\Delta \cap \Gamma = \emptyset$*;*
- Λ *is a finite* set of labels *for membranes;*
- μ *is a* membrane structure *(i.e., a rooted unordered tree, usually represented by nested brackets) in which each membrane is labelled by an element of* Λ *in a one-to-one way, and possesses an attribute called* electrical charge*, which can be either neutral (0), positive (+) or negative (-);*
- $w_1, \ldots, w_d$ *are strings over* Γ*, describing the* initial multisets *of objects placed in a number of* d *membranes of* μ*; notice that* w_i *is assigned to membrane* i*;*
- R *is a finite* set of rules *over* $\Gamma \cup \Delta$*:*
  1. $[a \to w]_h^\alpha$ object evolution rules
  An object $a \in \Gamma$ *is rewritten into the multiset* w*, if* a *is placed inside a membrane labelled by* h *with charge* α*. An object* a *can be deleted by considering* w *the empty multiset* $\emptyset$*. Notice that these rules allow only to rewritten objects from* Γ*, but not from* Δ*.*
  2. $a[\ ]_h^\alpha \to [b]_h^\beta$ send-in communication rules
  An object a *is sent into a membrane labelled by* h *and with charge* α*, becoming* b*; also, the charge of* h *is changed to* β*. If* $b \in \Delta$*, then* $a = b$ *must hold.*
  3. $[a]_h^\alpha \to b[\ ]_h^\beta$ send-out communication rules
  An object a*, placed into a membrane labelled by* h *and having charge* α*, is sent out of membrane* h *and becomes* b*; simultaneously, the charge of* h *is changed to* β*. If* $b \in \Delta$*, then* $a = b$ *must hold.*

Each configuration $\mathcal{C}_i$ of a P system with active membranes and input objects is described by the current membrane structure, including the electrical charges, together with the multisets of objects located in the corresponding membranes. The initial configuration of such a system is denoted by $\mathcal{C}_0$. An evolution step from the current configuration $\mathcal{C}_i$ to a new configuration $\mathcal{C}_{i+1}$, denoted by $\mathcal{C}_i \Rightarrow \mathcal{C}_{i+1}$, is done according to the principles:

- Each object and membrane is involved in at most one communication rule per step.
- Each membrane could be involved in several object evolution rules that can be applied in parallel inside it.
- The application of rules is maximally parallel: the only objects and membranes that do not evolve are those associated with no rule, or only to rules that are not applicable due to the electrical charges.
- When several conflicting rules could be applied at the same time, a nondeterministic choice is performed; this implies that multiple configurations can be reached as the result of an evolution step.
- In each computation step, all the chosen rules are applied simultaneously.
- Any object sent out from the skin membrane cannot re-enter it.

A *halting evolution* of such a system Π is a finite sequence of configurations $\overrightarrow{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$, such that $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \ldots \Rightarrow \mathcal{C}_k$, and no rules can be applied any more in $\mathcal{C}_k$. A *non-halting* evolution $\overrightarrow{\mathcal{C}} = (\mathcal{C}_i \mid i \in \mathbb{N})$ consists of an infinite evolution $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \ldots$, where the applicable rules are never exhausted.

Example 1. Addition is trivial; we consider n objects a and m objects b placed in a membrane 0 with charge $+$. The rule $[b \rightarrow a]_+$ says that an object b is transformed in one object a. Such a rule is applied in parallel as many times as possible. Consequently, all objects b are erased. The remaining number of objects a represents the addition $n + m$. More examples can be found in [5].

In order to solve decision problems (i.e., decide languages over an alphabet Σ), we use *families* of recognizer P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^*\}$ that respect the following conditions: (1) all evolutions halt; (2) two additional objects *yes* (successful evolution) and *no* (unsuccessful evolution) are used; (3) one of the objects *yes* and *no* appears in the halting configuration [13]. Each input x is associated with a P system Π_x that decides the membership of x in the language $L \subseteq \Sigma^*$ by accepting or rejecting it. The mapping $x \vdash \Pi_x$ must be efficiently computable for each input length [10].

In this paper we use a logarithmic space uniformity condition [14].

Definition 2. *A family of P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^*\}$ is said to be $(\mathbf{L}, \mathbf{L})$-uniform if the mapping $x \vdash \Pi_x$ can be computed by two deterministic logarithmic space Turing machines F (for "family") and E (for "encoding") as follows:*

- *F computes the mapping $1^n \vdash \Pi_n$, where Π_n represents the membrane structure with some initial multisets and a specific input membrane, while n is the length of the input x.*
- *E computes the mapping $x \vdash w_x$, where w_x is a multiset encoding the specific input x.*
- *Finally, Π_x is Π_n with w_x added to the multiset placed inside its input membrane.*

In the following definition of space complexity adapted from [14], the input objects do not contribute to the size of the configuration of a P system. In this way, only the actual working space of the P system is measured, and P systems working in sublinear space may be analyzed.

Definition 3. *Given a configuration $\mathcal{C}$, the space size $|\mathcal{C}|$ is defined as the sum of the number of membranes in μ and the number of objects in Γ it contains. If $\overrightarrow{\mathcal{C}}$ is a halting evolution of Π, then $|\overrightarrow{\mathcal{C}}| = max\{|\mathcal{C}_0|, \ldots, |\mathcal{C}_k|\}$ or, in the case of a non-halting evolution $\overrightarrow{\mathcal{C}}$, $|\overrightarrow{\mathcal{C}}| = sup\{|\mathcal{C}_i| \mid i \in \mathbb{N}\}$. The space required by Π itself is then $|\Pi| = sup\{|\overrightarrow{\mathcal{C}}| \mid \overrightarrow{\mathcal{C}}$ is an evolution of $\Pi\}$.*

Notice that $|\Pi| = \infty$ if Π has an evolution requiring infinite space or an infinite number of halting evolutions that can occur such that for each $k \in \mathbb{N}$ there exists at least one evolution requiring most than k steps..

3 A Membrane Structure for Simulation

Let M be a single-tape deterministic Turing machine working in polynomial space $s(n)$. Let Q be the set of states of M, including the initial state s; we denote by $\Sigma' = \Sigma \cup \{\sqcup\}$ the tape alphabet which includes the blank symbol $\sqcup \notin \Sigma$. A computation step is performed by using $\delta : Q \times \Sigma' \to Q \times \Sigma' \times \{-1, 0, 1\}$, a (partial) transition function of M which we assume to be undefined on (q, σ) if and only if q is a final state. We describe a uniform family of P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^*\}$ simulating M in logarithmic space.

Let $x \in \Sigma^n$ be an input string, and let $m = \lceil log\ s(n) \rceil$ be the minimum number of bits needed in order to write the tape cell indices $0, \ldots, s(n)$-1 in binary notation. The P system Π_n associated with the input length n and computed as $F(1^n)$ has a membrane structure consisting of $|\Sigma'| \cdot (m + 1) + 2$ membranes. The membrane structure contains:

- a skin membrane h;
- an inner membrane c (the input membrane) used to identify the symbol needed to compute the δ function;
- for each symbol $\sigma \in \Sigma'$ of M, the following set of membranes, linearly nested inside c and listed inward:
  - a membrane σ for each symbol σ of the tape alphabet Σ' of M;
  - for each $j \in \{0, \ldots, (m-1)\}$, a membrane labelled by j.

This labelling is used in order to simplify the notations. To respect the one-to-one labelling from Definition 1, the membrane j can be labelled j_σ. Thus in all rules using membranes j, the σ symbol is implicitly considered. Furthermore, the object z_0 is located inside the skin membrane h.

The encoding of x, computed as $E(x)$, consists of a set of objects describing the tape of M in its initial configuration on input x. These objects are the symbols of x subscripted by their position $bin(0), \ldots, bin(n-1)$ (where $bin(i)$ is the binary representation of i on m positions) in x, together with the $s(n) - n$

blank objects subscripted by their position $bin(n), \ldots, bin(s(n)-1)$. The binary representation, together with the polarities of the membranes, is essential when the membrane system has to identify the symbol needed to simulate the δ function (e.g., rule (13)). The multiset $E(x)$ is placed inside the input membrane c. Figure 1 depicts an example.

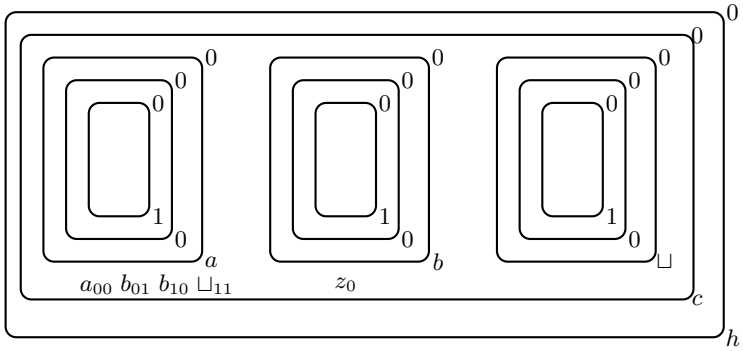

Fig. 1. Initial configuration of the P system Π_3 with tape alphabet $\Sigma' = \{a, b, \sqcup\}$, working in space $s(n) = n + 1 = 4$ on the input abb.

During the first evolution steps of Π_x, each input object σ_i is moved from the input membrane c to the innermost membrane $(m-1)$ of the corresponding membrane σ by means of the following communication rules:

$$\sigma_i[\ ]^0_\sigma \to [\sigma_i]^0_\sigma \qquad \text{for } \ \sigma \in \Sigma',\ bin(0) \le i < bin(s(n)) \tag{1}$$

$$\sigma_i[\ ]^0_j \to [\sigma_i]^0_j \qquad \text{for } \ \sigma \in \Sigma',\ bin(0) \le i < bin(s(n)),\ 0 \le j < m \tag{2}$$

Since only one communication rule per membrane can be applied during each evolution step of Π_x, all $s(n)$ input objects pass through m membranes, in order to reach the innermost membranes $(m-1)$, in at most $l = s(n) + m$ evolution steps. In the meantime, the subscript of object z_0 is incremented up to $max\{0, l-3\}$ before object z_{l-3} exits and enters membrane c changing the membrane charge from 0 to +:

$$[z_t \to z_{t+1}]^0_c \qquad \text{for } \ 0 \le t < l-3 \tag{3}$$

$$[z_{l-3}]^0_c \to z_{l-3}[\ ]^0_c \tag{4}$$

$$z_{l-3}[\ ]^0_c \to [z_{l-3}]^+_c \tag{5}$$

The object z_{l-3} is rewritten to a multiset of objects containing an object z' (used in rule (9)) and $|\Sigma'|$ objects z_+ (used in rules (7)

$$[z_{l-3} \to z' \underbrace{z_+ \cdots z_+}_{|\Sigma'| \text{ copies}}\ ]^+_c \tag{6}$$

The objects z_+ are used to change the charges from 0 to + for all membranes $\sigma \in \Sigma'$ using parallel communication rules, and then are deleted:

$$z_+[\ ]^0_\sigma \to [\#]^+_\sigma \qquad \text{for } \sigma \in \Sigma' \tag{7}$$

$$[\# \to \emptyset]^+_\sigma \qquad \text{for } \sigma \in \Sigma' \tag{8}$$

In the meantime, the object z' is rewritten into z'' (in parallel with rule (7)), and then, in parallel with rule (8), into s_{00} (where s is the initial state of M):

$$[z' \to z'']^+_c \tag{9}$$

$$[z'' \to s_{00}]^+_c \tag{10}$$

The configuration reached by Π_x encodes the initial configuration of M:

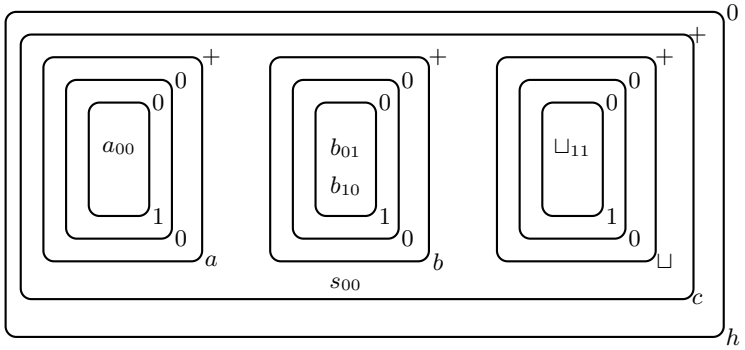

Fig. 2. Configuration of Π_x (from Figure 1) encoding the initial configuration of M on input $x = abb$ and using $s(|x|) = 4$ tape cells.

An arbitrary configuration of M on input x is encoded by a configuration of Π_x as it is described in Figure 3:

- membrane c contains the state-object q_i, where q is the current state of M and $i \in \{bin(0), \ldots, bin(s(n) - 1)\}$ is the current position of the tape head;
- membranes $(m - 1)$ contain all input objects;
- all other membranes are empty;
- all membranes are neutrally charged, except those labelled by $\sigma \in \Sigma'$ and by c which all are positively charged.

We employ this encoding because the input objects must be all located in the input membrane in the initial configuration of Π_x (hence they must encode both symbol and position on the tape), and they can never be rewritten.

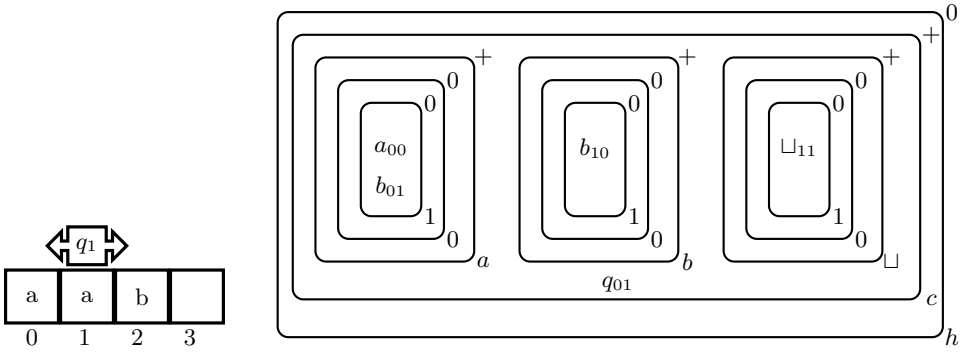

Fig. 3. A configuration of M (from Figure 1) and the corresponding configuration of Π_x simulating it. The presence of b_{01} inside membrane 1 of a indicates that tape cell 1 of M contains the symbol a.

4 Simulating Polynomial Space Turing Machines

Starting from a configuration of the single-tape deterministic Turing machine M, the simulation of a computation step of M by the membrane system Π_x is directed by the state-object q_i. As stated above, q_i encodes the current state of M and the position of the head on the tape (in binary format). To simulate the transition function δ of the Turing machine M in state q, it is necessary to identify the actual symbol occurring at tape position i. In order to identify this σ_i object from one of the $(m-1)$ membranes, the object q_i is rewritten into $|\Sigma'|$ copies of q_i', one for each membrane $\sigma \in \Sigma'$:

$$[q_i \to \underbrace{q_i' \cdots q_i'}_{|\Sigma'| \text{ copies}}]_c^+ \qquad q \in Q,\, bin(0) \le i < bin(s(n)) \tag{11}$$

The objects q_i' first enter the symbol-membranes in parallel, without changing the charges:

$$q_i'[\ ]_\sigma^+ \to [q_i']_\sigma^+ \quad \text{for} \quad \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{12}$$

The object q_i' traverses the membranes $0, \ldots, (m-1)$ while changing their charges such that they represent the bits of i from the least to the most significant one, where a positive charge is interpreted as 1 and a negative charge as 0. For instance, the charges of $[[[\ ]_2^-]_1^-]_0^+$ encode the binary number 001 (that is, decimal 1). By the j-th bit of a binary number is understood the bit from the j-th position when the number is read from right to left (e.g, the 0-th bit of the binary number 001 is 1). The changes of charges are accomplished by the rules:

$$q_i'[\ ]_j^0 \to [q_i']_j^\alpha \quad \text{for} \quad \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)),\, 0 \le j < m, \tag{13}$$

where α is $-$ if the j-th bit of i is 0, and α is $+$ if the j-th bit of i is 1.

The membranes j, where $0 \le j < m$, behave now as "filters" for the input objects σ_k occurring in membrane $(m-1)$: these objects are sent out from each membrane j if and only if the j-th bit of k corresponds to the charge of j.

$$[\sigma_k]_j^\alpha \to [\ ]_j^\alpha \sigma_k \quad \text{for} \quad \sigma \in \Sigma',\, bin(0) \le k < bin(s(n)),\, 0 < j < m, \tag{14}$$

where α is $-$ if the j-th bit of k is 0, and α is $+$ if j-th bit of k is 1.

If an object σ_k reaches membrane 0, it is sent outside if the 0-th bit of k corresponds to the charge of membrane 0. In order to signal that it is the symbol occurring at location i of the tape, the charge of the corresponding membrane 0 is changed (either from $+$ to $-$ or from $-$ to $+$). By applying the rules (15) to (17), exactly one object σ_k, with $k = i$, will exit through membrane c:

$$\begin{array}{l} [\sigma_k]_0^+ \rightarrow [\ ]_0^- \sigma_k \quad \text{for} \ \ \sigma \in \Sigma', \, bin(0) \leq k < bin(s(n)), \\ [\sigma_k]_0^- \rightarrow [\ ]_0^+ \sigma_k \quad \text{for} \ \ \sigma \in \Sigma', \, bin(0) \leq k < bin(s(n)), \end{array} \tag{15}$$

where α is $-$ if the j-th bit of k is 0, and α is $+$ if the j-th bit of k is 1;

$$[\sigma_k]_\tau^+ \rightarrow \sigma_k [\ ]_\tau^- \quad \text{for} \ \ \sigma, \tau \in \Sigma', \, bin(0) \leq k < bin(s(n)). \tag{16}$$

After an σ_i exits from membrane c it gets blocked inside membrane h, by the new charge of membrane c, until it is allowed to move to its new location according to function δ of the Turing machine M. Thus, if another object τ_j reached membrane σ due to the new charge of membrane 0 established by rule (15), τ_j is contained in membrane σ until reintroduced in a membrane (m-1) using rule (2).

$$[\sigma_k]_c^+ \rightarrow \sigma_k [\ ]_c^- \quad \text{for} \ \ \sigma \in \Sigma', \, bin(0) \leq k < bin(s(n)) \tag{17}$$

Since there are $s(n)$ input objects, and each of them must traverse at most $(m + 1)$ membranes, the object σ_i reaches the skin membrane h after at most $l + 1$ steps, where l is as defined in Section 3 before rule (3). While the input objects are "filtered out", the state-object q_i' "waits" for l steps using the rules:

$$[q_i' \rightarrow q_{i,1}'']_{m-1}^\alpha \quad \text{for} \ \ \sigma \in \Sigma', \, q \in Q, \, bin(0) \leq i < bin(s(n)), \, \alpha \in \{-, +\} \tag{18}$$

$$\begin{array}{c} [q_{i,t}'' \rightarrow q_{i,t+1}'']_{m-1}^\alpha \quad \text{for} \ \ \sigma \in \Sigma', \, q \in Q, \, bin(0) \leq i < bin(s(n)), \\ \alpha \in \{-, +\}, \, 1 \leq t \leq l \end{array} \tag{19}$$

$$[q_{i,l+1}'' \rightarrow q_i'']_{m-1}^\alpha \quad \text{for} \ \ \sigma \in \Sigma', \, q \in Q, \, bin(0) \leq i < bin(s(n)), \, \alpha \in \{-, +\} \tag{20}$$

In order to reach membrane c, the objects q_i'' are sent out through membranes j $(0 < j \leq m - 1)$ using rule (21), through membrane 0 by rules (22) and (24), and through membranes $\sigma \in \Sigma'$ by rule (23). While passing through all these membranes, the charges are changed to neutral. This allows the input objects to move back to the innermost membrane $(m - 1)$ by using rules of type (2).

$$\begin{array}{c} [q_i'']_j^\alpha \rightarrow [\ ]_j^0 q_i'' \quad \text{for} \ \ \sigma \in \Sigma', \, q \in Q, \, bin(0) \leq i < bin(s(n)) \\ 0 < j \leq m - 1, \, \alpha \in \{-, +\} \end{array} \tag{21}$$

When q_i'' reaches the membranes 0, only one has the charge different from the 0-th bit of i, thus allowing q_i'' to identify the symbol in tape location i of M:

$$[q_i'']_0^\alpha \rightarrow [\ ]_0^0 q_i''' \quad \text{for} \ \ \sigma \in \Sigma', \, q \in Q, \, bin(0) \leq i < bin(s(n)), \tag{22}$$

where α is $-$ if the 0-th bit of i is 1, and α is $+$ if the 0-th bit of i is 0.

$$[q_i'''\,]_\sigma^- \to [\ ]_\sigma^0 q_{i,\sigma,1} \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{23}$$

The other copies of q_i'' are sent out as objects # through membrane 0, and then deleted by rules of type (8):

$$[q_i''\,]_0^\alpha \to [\ ]_0^0 \# \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)), \tag{24}$$

where α is $-$ if the 0-th bit of i is 0, and α is $+$ if the 0-th bit of i is 1.

The state-object $q_{i,\sigma,1}$ waits in membrane c for l steps, l representing an upper bound of the number of steps needed for all the input objects to reach the innermost membranes:

$$[q_{i,\sigma,t} \to q_{i,\sigma,t+1}]_c^- \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)),\, 1 \le t < l \tag{25}$$

$$q_{i,\sigma,l}[\ ]_\sigma^0 \to [q_{i,\sigma,l}]_\sigma^+ \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{26}$$

$$[q_{i,\sigma,l}]_\sigma^+ \to q'_{i,\sigma}[\ ]_\sigma^+ \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{27}$$

The state-object $q'_{i,\sigma}$ now contains all the information needed to compute the transition function δ of the Turing machine M. Suppose $\delta(q,\sigma) = (r,v,d)$ for some $d \in \{-1,0,+1\}$. Then $q'_{i,\sigma}$ sets the charge of membrane v to $-$ and waits for $m+1$ steps, thus allowing σ_i to move to membrane $(m-1)$ of v by using the rules (31), (32) and (2):

$$q'_{i,\sigma}[\ ]_v^+ \to [q'_{i,\sigma}]_v^+ \quad \text{for } \sigma, v \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{28}$$

$$[q'_{i,\sigma}]_v^+ \to q'_{i,\sigma,1}[\ ]_v^- \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{29}$$

$$[q'_{i,\sigma,1}]_c^- \to q'_{i,\sigma,1}[\ ]_c^0 \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{30}$$

$$\sigma_i[\ ]_c^0 \to [\sigma_i]_c^0 \quad \text{for } \sigma \in \Sigma',\, bin(0) \le i < bin(s(n)) \tag{31}$$

$$\sigma_i[\ ]_v^- \to [\sigma_i]_v^- \quad \text{for } \sigma, v \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{32}$$

$$[q'_{i,\sigma,t} \to q'_{i,\sigma,t+1}]_h^0 \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)),\ 1 \le t \le m \tag{33}$$

The object $q'_{i,\sigma,m+1}$ is used to change the charges of membranes c and v to $+$, thus preparing the system for the next step of the simulation:

$$q'_{i,\sigma,m+1}[\ ]_c^0 \to [q'_{i,\sigma,m+1}]_c^+ \quad \text{for } \sigma \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{34}$$

$$q'_{i,\sigma,m+1}[\ ]_v^- \to [q'_{i,\sigma,m+1}]_v^- \quad \text{for } \sigma, v \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{35}$$

$$[q'_{i,\sigma,m+1}]_v^- \to q''_{i,\sigma}[\ ]_v^+ \quad \text{for } \sigma, v \in \Sigma',\, q \in Q,\, bin(0) \le i < bin(s(n)) \tag{36}$$

Finally, the state-object $q''_{i,\sigma}$ is rewritten to reflect the change of state and head position, thus producing a configuration of Π_x corresponding to the new configuration of M, as described in Section 3:

$$[q''_{i,\sigma} \to r_{i+d}]_c^+ \quad \text{for } bin(0) \le i < bin(s(n)) \tag{37}$$

The P system Π_x is now ready to simulate the next step of M. If $q \in Q$ is a final state of M, we assume that $\delta(q, \sigma)$ is undefined for all $\sigma \in \Sigma'$; thus we introduce the following rules which halt the P system with the same result (acceptance or rejection) as M:

$$[q_i]_c^+ \rightarrow [\ ]_c^+ yes \quad \text{for} \quad bin(0) \leq i < bin(s(n)), \text{ if } q \text{ is an accepting state} \tag{38}$$

$$[yes]_h^0 \rightarrow [\ ]_h^0 yes \quad \text{for} \quad bin(0) \leq i < bin(s(n)), \text{ if } q \text{ is an accepting state} \tag{39}$$

$$[q_i]_c^+ \rightarrow [\ ]_c^+ no \quad \text{for} \quad bin(0) \leq i < bin(s(n)), \text{ if } q \text{ is a rejecting state} \tag{40}$$

$$[no]_h^0 \rightarrow [\ ]_h^0 no \quad \text{for} \quad bin(0) \leq i < bin(s(n)), \text{ if } q \text{ is a rejecting state} \tag{41}$$

The simulation directly leads to the following result.

Theorem 1. *Let M be a single-tape deterministic Turing machine working in polynomial space $s(n)$ and time $t(n)$. Then there exists an $(\mathbf{L}, \mathbf{L})$-uniform family $\mathbf{\Pi}$ of P systems Π_x with active membranes using object evolution and communication rules that simulates M in space $O(log\ n)$ and time $O(t(n)s(n))$.*

Proof. For each $x \in \Sigma^n$, the P system Π_x can be built from 1^n and x in logarithmic space as it is described in Definition 2; thus, the family $\mathbf{\Pi}$ is $(\mathbf{L}, \mathbf{L})$-uniform. Each P system Π_x uses only a logarithmic number of membranes and a constant number of objects per configuration; thus, Π_x works in space $O(log\ n)$. Simulating one of the $t(n)$ steps of M requires $O(s(n))$ time, an upper bound to the subscripts of objects used to introduce delays during the simulation; thus, the total time is $O(t(n)s(n))$.

5 Conclusion

In this paper we provided a simulation of the polynomial space Turing machines by using logarithmic space P systems with active membranes and binary representations for the positions on the tape. A similar approach is presented in [9]. There are important differences in terms of technical details and efficient representation; in comparison to [9], we improve the simulation by reducing the number of membranes (by $|\Sigma'| - 1$) and the number of rules (by $|\Sigma'| \cdot |Q| \cdot s(n) \cdot (5 - |\Sigma'|) + |\Sigma'| \cdot |\Sigma'| s(n) \cdot (2 \cdot m + 1) + |Q| \cdot s(n) - |\Sigma'| \cdot s(n) \cdot (m + 3)$). In particular, for the running example, the number of rules is reduced by $14 \cdot |Q| + 84$. A different approach is presented in [8] where it is claimed that a constant space is sufficient. However, in order to obtain the constant space space, input objects (from Δ) are allowed to create other objects (from Γ) leading to a different and more powerful formalism than the one used by us in this paper, and making such an approach not so interesting because of these unrealistic (powerful) input objects.

References

1. B. Aman, G.Ciobanu. Describing the Immune System Using Enhanced Mobile Membranes. *Electronic Notes in Theoretical Computer Science* **194**, 5–18 (2008).
2. B. Aman, G.Ciobanu. Turing Completeness Using Three Mobile Membranes. *Lecture Notes in Computer Science* **5715**, 42–55 (2009).
3. B. Aman, G. Ciobanu. *Mobility in Process Calculi and Natural Computing.* Natural Computing Series, Springer (2011).
4. D. Besozzi, G. Ciobanu. A P System Description of the Sodium-Potassium Pump. *Lecture Notes in Computer Science* **3365**, 210–223 (2004).
5. C. Bonchiş, G. Ciobanu, C. Izbaşa. Encodings and Arithmetic Operations in Membrane Computing. *Lecture Notes in Computer Science Volume* **3959**, 621–630 (2006).
6. M. Cavaliere. Evolution-Communication P Systems. *Lecture Notes in Computer Science* **2597**, 134–145 (2003).
7. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.). *Applications of Membrane Computing*. Springer (2006).
8. A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron. Constant-Space P Systems with Active Membranes. *Fundamenta Informaticae* **134**(1–2), 111–128 (2014).
9. A. Leporati, G. Mauri, A.E. Porreca, C. Zandron. A Gap in the Space Hierarchy of P Systems With Active Membranes. *Journal of Automata, Languages and Combinatorics* **19** (1-4), 173–184 (2014).
10. N. Murphy, D. Woods. The Computational Power of Membrane Systems Under Tight Uniformity Conditions. *Natural Computing* **10**, 613-632 (2011).
11. Gh. Păun. P Systems With Active Membranes: Attacking NP-complete Problems. *Journal of Automata, Languages and Combinatorics* **6**, 75-90 (2001).
12. Gh. Păun, G. Rozenberg, A. Salomaa (Eds.). *The Oxford Handbook of Membrane Computing*, Oxford University Press (2010).
13. M.J. Pérez-Jiménez, A. Riscos-Núñez, A. Romero-Jiménez, D. Woods. Complexity-Membrane Division, Membrane Creation. In [12], 302–336.
14. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron, Sublinear-Space P Systems with Active Membranes. *Lecture Notes in Computer Science* **7762**, 342–357 (2013).