On P Systems with Active Membranes

Andrei Păun
Department of Computer Science, University of Western Ontario
London, Ontario, Canada N6A 5B7
E-mail: apaun@csd.uwo.ca

Abstract. The paper deals with the vivid area of computing with membranes (P systems). We improve here two recent results about the so-called P systems with active membranes. First, we show that the Hamiltonian Path Problem can be solved in polynomial time by P systems with active membranes where the membranes are only divided into two new membranes (a result of this type was obtained by Krishna and Rama, [4], but making use of the possibility of dividing a membrane in an arbitrary number of new membranes). We also show that HPP can be solved in polynomial time also by a variant of P systems, with the possibility of dividing non-elementary membranes under the influence of objects present in them. Then, we show that membrane division (and even membrane dissolving) is not necessary in order to show that such systems are computationally complete.

1 Introduction

In the framework of Molecular Computing, a new model for distributed parallel computing has been recently introduced in [6], cell-like devices called P systems, which try to model the work of the alive cells.

Each cell is delimited by its membrane and can contain objects and other cells (the objects and the other cells are floating in this membrane). The objects are represented by symbols, their multiplicity is taken into account (that is, we work with multisets), and they can evolve by using given evolution rules (which are specific to each membrane).

The objects can pass from one cell to another one (if the cells are adjacent), they can dissolve a membrane (and in this case all other objects go into the immediately higher cell), or leave the biggest membrane (the skin membrane), which means that that specific object leaves the system.

A sequence of transitions among configurations of the system is interpreted as a computation. The result of a halting computation is the vector describing the multiplicities of objects which have left the system during the computation.

Many variants of P systems have been considered and investigated in [2], [3], [5], [8], etc. In [7] one considers also the possibility of letting the number of membranes increase during a computation by using some division rules (we will define them later as rules of types (e), (e'), (e_I), (f) and (f_I)). That is modelling the division of a cell from nature.

In the initial variant ([7]) the division rules were defined such that from the division of a membrane we obtain always two membranes (we call it a 2-bounded division); a generalisation was defined in [4] where a membrane can divide in several (but a finite number) of membranes (we call it a k-bounded division).

Using the 2-bounded division it was shown that the NP-complete problem SAT can be solved in linear time [7]. By using the k-bounded division, in [4] was shown that also the Hamiltonian path problem (HPP) can be solved in linear time [4]. Remember that HPP was the problem dealt with in the first experiment in DNA Computing, [1].

In [4], the open problem is formulated whether or not a system with k-bounded division can be simulated by a system with 2-bounded division, with a reasonable slowdown (polynomial, at most). We do not attack here this problem, but we directly shown that HPP can also be solved in polynomial time by using P systems with a 2-bounded division. Thus, the extension from 2-bounded to k-bounded division is not necessary from this point of view.

Actually, we give two results of this type, one in the framework of the definition in [7] (which is generalized in [4]), and one with the variant of evolution rules as considered in [9] (elementary or non-elementary membranes are divided under the influence of objects, but not under the influence of lower membranes).

The P systems with active membrane were shown in [7] to be computationally complete: any set of vectors of natural numbers which can be computed by a Turing machine can be computed by such a system. The result was improved in [9]: rules of type (f) (non-elementary membrane division under the influence of lower membranes) are not necessary. We improve here this result: no membrane division operation is necessary in order to obtain the computability completeness.

2 Basic Definitions

We refer the reader to [10] for basics of formal language theory and to [6], [7] for basic notions, notations and results about P systems; here we directly introduce the class of systems we will investigate in the subsequent sections. We only mention that the membrane structures are represented as strings of matching parentheses, with the parentheses labeled by elements in a given set, and with each membrane electrically polarized, that is, marked with one of the symbols $+, -, 0$. For example, the string $[_1[_2\,]_2^+[_3[_4\,]_4^-]_3^0]_1^0$ is a membrane structure as illustrated in Figure 1, with four membranes arranged in a hierarchy of depth three with membrane 1 neutral, membrane 2 positive, membrane 3 neutral and membrane 4 negative.

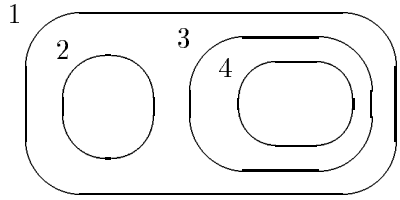

Figure 1: A membrane structure.

A *P system with active membranes and 2-bounded membrane division* is a construct

$$\Pi = (V, T, H, \mu, w_1, \ldots, w_m, R),$$

where: $m \geq 1$; V is an alphabet (the *total alphabet* of the system); $T \subseteq V$ (the *terminal* alphabet); H is a finite set of *labels* for membranes; μ is a *membrane structure*, consisting of m membranes, labeled with elements of H, and having a neutral charge (initially, all membranes are marked with

0); $w_1, \ldots, w_m$ are strings over V, describing the *multisets of objects* placed in the m regions of μ; R is a finite set of *rules*, of the following forms:

(a) $[_h a \to v]_h^\alpha$, for $h \in H,\ a \in V,\ v \in V^*,\ \alpha \in \{+, -, 0\}$ (object evolution rules),

(b) $a[_h]_h^{\alpha_1} \longrightarrow [_h b]_h^{\alpha_2}$, where $a, b \in V,\ h \in H,\ \alpha_1, \alpha_2 \in \{+, -, 0\}$ (an object is introduced in the membrane h),

(c) $[_h a]_h^{\alpha_1} \to b[_h]_h^{\alpha_2}$, for $h \in H,\ \alpha_1, \alpha_2 \in \{+, -, 0\},\ a, b \in V$ (an object is sent out of the membrane h),

(d) $[_h a]_h^\alpha \to b$, for $h \in H,\ \alpha \in \{+, -, 0\},\ a, b \in V$ (the membrane h was dissolved),

(e) $[_h a\ ]_h^{\alpha_1} \to [_h b\ ]_h^{\alpha_2} [_h c\ ]_h^{\alpha_3}$,
for $h \in H,\ \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\},\ a, b, c \in V$
(2-division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, maybe of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects, all other objects are duplicated in the two new copies of the membrane);

(f) $[_{h_0}[_{h_1}\ ]_{h_1}^{\alpha_1} \ldots [_{h_k}\ ]_{h_k}^{\alpha_1}\ [_{h_{k+1}}\ ]_{h_{k+1}}^{\alpha_2} \ldots [_{h_n}\ ]_{h_n}^{\alpha_2}]_{h_0}^{\alpha_0}$
$\to [_{h_0}[_{h_1}\ ]_{h_1}^{\alpha_3} \ldots [_{h_k}\ ]_{h_k}^{\alpha_3}]_{h_0}^{\alpha_5}\ [_{h_0}[_{h_{k+1}}\ ]_{h_{k+1}}^{\alpha_4} \ldots [_{h_n}\ ]_{h_n}^{\alpha_4}]_{h_0}^{\alpha_6}$,
for $k \geq 1, n > k, h_i \in H, 0 \leq i \leq n$, and $\alpha_0, \ldots, \alpha_6 \in \{+, -, 0\}$ with $\{\alpha_1, \alpha_2\} = \{+, -\}$; if this membrane with the label h_0 contains other membranes than those with the labels $h_1, \ldots, h_n$ specified above, then they must have neutral charges in order to make this rule applicable; these membranes are duplicated and then are part of the contents of both copies of the membrane h_0;
(2-division of non-elementary membranes; this is possible only if a membrane directly contains two membranes of opposite polarization, + and −; the membranes of opposite polarizations are separated in the two new membranes, but their polarization can change; always, all membranes of opposite polarizations are separated by applying this rule).

We refer to [7] and [9] for a more precise definition of the way a P system as above works and we only describe here informally the way of passing from a configuration of the system to the next one.

The rules of types (a) to (d) are applied in a maximally parallel manner: in each time unit, any objects which can evolve, have to evolve. If a membrane with label h is divided by a rule of type (e), which involves an object a, then all other objects in membrane h which do not evolve are introduced in each of the resulting membranes h. Similarly, when using a rule of type (f), the membranes which are not specified in the rule, (that is, they are neutral), are reproduced in each of the resulting membranes with the label h, unchanged if no rule is applied to them (the contents of these membranes are reproduced unchanged in these copies, providing that no rule is applied to their objects).

When applying a rule of type (f) or (e) to a membrane, if there are objects in this membrane which evolve by a rule of type (a), changing the objects, then in the new copies of the membrane we introduce the results of evolution. The rules are applied "from bottom up", in one step, but first the rules of the innermost region and, then, level by level until the region of the skin membrane. The rules associated with a membrane h are used for all copies of this membrane; it doesn't matter whether or not the membrane is an initial one or it was obtained by division. At one step, a membrane h can be the subject of only one rule of types (b) – (f). The skin membrane can never

divide (or dissolve). As any other membrane, the skin membrane can be "electrically charged". During a computation, objects can leave the skin membrane (using rules of type (c)).

The result of a halting computation (no rule can be used in the last configuration) consists of the vector of natural numbers describing the multiplicities of terminal objects which have left the system during the computation. Because of the non-determinism of choosing the rules and the objects which evolve at each step, several results are obtained starting from an initial configuration. We denote by $N(\Pi)$ the set of all vectors of natural numbers computed as above by a P system Π.

The symbols not in T which leave the skin membrane as well as all the symbols from T which remain in the system at the end of a halting computation are not contributing to the generated vectors; if a computation goes forever, then it provides no output, it does not contribute to the set $N(\Pi)$.

Further variants for these systems have been defined. For example, in [9] one allows the division of non-elementary membranes by using rule $[_i a]_i^{\alpha_1} \to [_i b]_i^{\alpha_2} [_i c]_i^{\alpha_3}$ of type (e); we say that this is a rule of type (e′).

Moreover, in [4] one considers rules (e_I) and (f_I) that generalize the rules (e) and (f) allowing one membrane to divide in a number of other membranes (in the initial variant they were allowed to divide in two membranes):

(e_I) $[_h a]_h^\alpha \to [_h a_1]_h^{\alpha_1} [_h a_2]_h^{\alpha_2} \ldots [_h a_n]_h^{\alpha_n}$,
for $\alpha, \alpha_i \in \{+, -, 0\}$, $a \in V$, $a_i \in V^*$, $i = 1, \ldots, n$, $h \in H$ (n-division rules for elementary membranes),

(f_I) $[_{h_0} [_{h_1}]_{h_1}^{\alpha_1} \ldots [_{h_k}]_{h_k}^{\alpha_k} [_{h_{k+1}}]_{h_{k+1}}^{\alpha_{k+1}} \ldots [_{h_n}]_{h_n}^{\alpha_n}]_{h_0}^{\alpha_0}$
$\to [_{h_0} [_{h_1}]_{h_1}^{\beta_1}]_{h_0}^{\beta_0} \ldots [_{h_0} [_{h_k}]_{h_k}^{\beta_k}]_{h_0}^{\beta_0} [_{h_0} [_{h_{k+1}}]_{h_{k+1}}^{\beta_{k+1}}]_{h_0}^{\beta_0} \ldots [_{h_0} [_{h_n}]_{h_n}^{\beta_n}]_{h_0}^{\beta_0}$,
for $k \geq 1$, $n > k$, $h_i \in H$, $0 \leq i \leq n$, and there is i, $1 \leq i \leq n-1$, such that $\{\alpha_i, \alpha_{i+1}\} = \{+, -\}$; moreover, $\beta_j \in \{+, -, 0\}$, $1 \leq j \leq n$ (n-division of non-elementary membranes).

One can notice that in the rules of type (e_I) we have a double generalisation, one dealing with the number of membranes produced (from 2 to n) and also in each membrane produced we have a string $a_i \in V^*$ (in the initial model we had only one symbol).

By using P systems with rules of types (e_I) and (f_I) instead of types (e) and (f) (it is immediate to see that such systems are more powerful than the initial ones), in [4] one shows that the Hamiltonian path problem can be solved in linear time. In that paper, it is also conjectured that the systems with rules (a)...(d), (e_I), and (f_I) can be simulated with systems with rules (a)...(f). We do not have an answer to this problem, but we directly show that HPP can be solved in polynomial time by using systems with rules of the forms (a)...(f).

3 Solving HPP by P Systems with Active Membranes

We will continue in this section the work from [7], namely, we will solve the Hamiltonian Path Problem in polynomial time (specifically, in $O(n^3)$ steps).

Theorem 3.1 *HPP can be solved in polynomial time by P systems with rules of types* $(a) \ldots (f)$.

Proof. Consider a graph $G = (V, E)$, where $V = \{0, 1, \ldots, n-2, n-1\}$, $2 \leq n$, is the set of vertices and E is the set of edges (without loss of the generality we may assume that no edge of

the form $\{i, i\}$ exists). We will construct a P system $\Pi = (V, H, \mu, w_0, w_1, \ldots, w_{n-1}, w_n, R)$, taking G as input and producing as the output the letter t at the moment $2n^3 + 2n$ if and only if there is a Hamiltonian path in G starting in node 0.

The components of the P system Π are as follows:

$$
\begin{aligned}
V &= \{i, b_i, d_i, u_i \mid 0 \leq i \leq n-1\} \cup \{c_i \mid 0 \leq i \leq 2n-2\} \cup \{t_i \mid 0 \leq i \leq 2n^2-2\} \\
&\quad \cup \{u_{j_1 \ldots j_k} \mid 2 \leq k \leq n-2,\ 0 \leq j_1, \ldots, j_k \leq n-1\} \cup \{z, z_0, z_1, z_2, a, b, d, e, e', u', x, y, t, \#\}, \\
H &= \{0, 1, \ldots, n-1, n\}, \\
\mu &= [_n[_{n-1} \cdots [_1[_0\,]_0^0]_1^0 \cdots]_{n-1}^0]_n^0, \\
w_0 &= u_0 c_0, w_n = \lambda, \\
w_i &= e, \text{ for all } i = 1, 2, \ldots, n-1,
\end{aligned}
$$

and the set R contains the following rules:

0) $[_i e]_i^0 \to [_i\,]_i^+ e'$, for $i = 1, \ldots, n-1$,

1) $[_0 c_i \to c_{i+1}]_0^0$, for $0 \leq i < 2n-2$,

1′) $[_0 c_{2n-2}]_0^0 \to t$,

2) $[_0 u_i \to i b_i d]_0^0$, for $1 \leq i \leq n-1$,

2′) $[_0 u_0 \to b_0 d]_0^0$,

3) $[_0 d]_0^0 \to d_0 [_0]_0^+$,

4) $[_1 d_i \to d_{i+1}]_1^+$, for $0 \leq i \leq n-2$,

5) $[_0 b_i]_0^+ \to [_0 u_{j_1}]_0^+ [_0 u_{j_2 \ldots j_k}]_0^+$,
$[_0 u_{j_2 \ldots j_k}]_0^+ \to [_0 u_{j_2}]_0^+ [_0 u_{j_3 \ldots j_k}]_0^+$,
. . . . . .
$[_0 u_{j_{k-1} j_k}]_0^+ \to [_0 u_{j_{k-1}}]_0^+ [_0 u_{j_k}]_0^+$,
where $j_1, \ldots, j_k$ are the nodes adjacent with the node i in the graph,

5′) $[_0 b_i]_0^+ \to [_0 u_j]_0^+$, if j is the only node adjacent with i in G,

6) $[_{j+1}[_j]_j^+ \cdots [_j]_j^+ [_j]_j^-]_{j+1}^{\alpha} \to [_{j+1}[_j]_j^+ \cdots [_j]_j^+]_{j+1}^+ [_{j+1}[_j]_j^+]_{j+1}^-$, for $0 \leq j \leq n-2, \alpha \in \{0, +\}$,

7) $d_{n-1}[_0]_0^+ \to [_0 u']_0^-$,

8) $[_0 u' \to a t_0]_0^-$,

9) $[_0 t_i \to t_{i+1}]_0^-$, for $0 \leq i \leq 2n^2 - 2$,

10) $[_0 a]_0^+ \to b[_0]_0^-$,

10′) $[_j b]_j^+ \to b[_j]_j^0$, for $1 \leq j \leq n-2$,

11) $[_{n-1} b]_{n-1}^- \to a[_{n-1}]_{n-1}^0$,

12) $a[_j]_j^+ \to [_j a]_j^0$, for $1 \leq j \leq n-1$,

13) $a[_0]_0^+ \to [_0 a]_0^-$,

14) $[_0 t_{2n^2-2}]_0^- \to x[_0]_0^-$,

15) $[_j x]_j^0 \to x[_j]_j^0$, for $1 \leq j \leq n-1$,

16) $[_n x \to y^{n-1} z_0 z]^0_n$,

17) $y[_j]^0_j \to [_j y]^+_j$, for $1 \leq j \leq n-1$,

18) $y[_0]^-_0 \to [_0 y]^0_0$,

19) $[_n z_0 \to z_1]^+_n$,

20) $[_n z_1 \to z_2]^+_n$,

21) $[_n z]^0_n \to \#[_n]^+_n$,

22) $[_n \alpha \to \#]^+_n$, for $\alpha \in \{y, z\}$

23) $[_n z_2]^+_n \to \#[_n]^0_n$,

23′) $[_n z \to \#]^+_n$,

24) $[_i i]^0_i \to i$, for $1 \leq i \leq n-1$,

25) $[_n t]^0_n \to t[_n]^+_n$.

We will explain now how this system works. We start with the symbols u_0 and c_0 in the membrane 0. At the first step, these symbols are changed into db_0 and c_1, respectively, by rules 2′ and 1; in parallel, all membranes $1, 2, \ldots, n-1$ will get a positive charge by means of rules of type 0. At the next step we will change the polarization of the membrane 0 using a rule of type 3; in parallel with this rule we apply again a rule of type 1.

In this moment membrane 0 has the polarization +, so no other rules of types 1, 2, 2', 3 can be applied.

We start by using the rules of types 4 and 5 in parallel; notice that the rules of type 4 simulate a timer (it will finish the work after the last rule of type 5 was applied). The rules of type 5 create many 0 membranes, in each one having (in the end) an object u_j, where $\{i, j\}$ is an edge in the initial graph. A node can have at most $n-1$ different nodes adjacent with it, so the number of rules of type 5 that can be used for a specific node i is at most $n-1$. One can observe also that using the special characters $u_{j_1 \ldots j_k}$ forces us to follow only one path. The computation can only continue using the rule of type 7, d_{n-1} enters one of the 0 membranes and changes the polarization of that particular membrane from + to − (and it is introduced in the membrane as u'). So, the membrane structure is now:

$$\mu = [_n[_{n-1} \cdots [_2[_1[_0\,]^+_0[_0\,]^+_0 \cdots [_0\,]^+_0[_0\,]^-_0]^0_1]^0_2 \cdots]^0_{n-1}]^0_n.$$

Now, u' generates two symbols, a and t_0 (rule 8), t_0 is a timer and a will change the sign of another membrane of type 0; in parallel, also a rule of type 6 is applied at this step (because we have a negative membrane). Because of the form of rule 6 we will produce two membranes, one positive and one negative, so another rule of type 6 will be applied at the next step and so on. Because the sign of membrane 0 is positive again we can only apply a rule of type 10 and the membrane is negative again (we cannot apply a rule of type 6 to this negative membrane because this is the only membrane of type 0 into membrane 1). Because membrane 0 is negatively charged, the timer starts working (rule 9). Now the rules of type 10′) are applied in parallel with the rules of type 6, so after $n-1$ steps the membrane structure will be as follows (the symbol b changes the membranes to neutral polarization when exiting them):

$$\mu = [_n[_{n-1} \cdots [_2[_1[_0\,]^+_0[_0\,]^+_0 \cdots [_0\,]^+_0]^+_1]^+_2 \cdots]^+_{n-1}[_{n-1}[_{n-2} \cdots [_2[_1[_0]^-_0]^0_1]^0_2 \cdots]^0_{n-2}]^-_{n-1}]^0_n$$

and in this moment the symbol b is in the membrane $n-1$ that is negative. So next we apply the rule of type 11 (notice that rules of type 6 cannot apply because the membrane n cannot divide), so we get:

$$\mu = [_n[_{n-1}\cdots[_2[_1[_0\,]_0^+[_0\,]_0^+\cdots[_0\,]_0^+]_1^+]_2^+\cdots]_{n-1}^+[_{n-1}[_{n-2}\cdots[_2[_1[_0]_0^-]_1^0]_2^0\cdots]_{n-2}^0]_{n-1}^0]_n^0$$

(the only change is in the polarization of the membrane $n-1$ and we have now the symbol a in the membrane n). We continue with rules of type 12; that is, the symbol a passes through membranes $n-1$ to 1 changing their polarization from + to 0. In $n-1$ steps we get:

$$\mu = [_n[_{n-1}\cdots[_2[_1[_0\,]_0^+[_0\,]_0^+\cdots[_0\,]_0^+]_1^0]_2^0\cdots]_{n-1}^0[_{n-1}[_{n-2}\cdots[_2[_1[_0]_0^-]_1^0]_2^0\cdots]_{n-2}^0]_{n-1}^0]_n^0$$

at the next step the symbol a enters one of the 0 membranes (rule 13) and changes the polarization of that membrane to $-$. Then the rule of type 6 is applied again; the symbol a exits the membranes $0, 1, \ldots, n-1$ and so on until we will have in each membrane 1 only one membrane 0; in that moment we cannot apply a rule of type 6, so the polarization of the 0 membrane remains $-$, so the symbol a remains in that membrane and this "branch" of the computation halts. We continue only after the timer t_i reaches the value t_{2n^2-2}, and at that moment we can apply the rule of type 14 that produces an x in membrane 1, then, by using rules of type 15, x exits the membranes $1, 2, \ldots, n-1$. In the membrane n, x produces $n-1$ copies of y, a copy of z_0 and a copy of z (rule 16). Next the y-s enter the membranes $n-1$ changing their polarities using rules of type 16). Notice that only one copy of y can enter one membrane, so there may be some number of y-s left in the membrane n at the next step, but they cannot enter any membrane $n-1$ because these membranes had their polarities changed at the previous step (and y can enter a membrane only if it is neutral). At the same time, z leaves the membrane n, changing its polarity in positive (rule 21), so at the next step all the remaining y-s will transform in the trap symbol # (rule 22) and at the next step z_2 leaves the membrane n making it neutral again (rule 23). After $n-1$ steps the y-s are present in all the membranes 1, so at the next step rule 18 will be applied (y enters the membranes 0 changing their polarizations to neutral). This change of polarization is very important, because we can apply again the rules of types 1, 2 and/or $2'$, so another step begins and the computation continues as described previously.

In the end, c_i will reach the value c_{2n-2}, which means that we cycled $n-1$ times, so we have paths of length n in the graph, so at the next step, when membranes 0 becomes neutral, we apply rules 2 and/or $2'$ and, at the same time, the rule $1'$; the rules of types 2, $2'$ introduce i instead of u_i and rules $1'$ dissolve all the 0 membranes at the same time (producing the symbol t). Then we will try to apply $n-1$ times the rules of type 24 (we dissolve the membrane i only if the symbol i is present; this means that we reached somewhere in that path the node i). If all the membranes $1, \ldots, n-1$ are dissolved somewhere then the symbol t reaches the membrane n and then can exit the system using rule 25 (we have found a path starting with node 0 that passed through all the nodes of the graph). If no copy of t reaches the membrane n, then we know that there is no Hamiltonian path starting with 0 in the graph G.

Following the steps of computation one can easily see that the symbol t exits the system at the moment $(n+3+2n^2-2+n+1)(n-1)+n+1+1+n-1+1 = 2n^3+2n$.

So, exactly after $2n^3+2n$ computation steps, the symbol t will exit the system if and only if there is a Hamiltonian path in the graph starting with the node 0. □

4 Solving HPP by P Systems of a Restricted Form

In this section we will show that the Hamiltonian path problem can be also solved using a restricted variant of the P systems with active membranes. Specifically, instead of using all types of rules: (a), ..., (f), we will prove that we can use only rules (a), ..., (d), (e′).

Theorem 4.1 *HPP can be solved in polynomial time by P systems with rules of types (a), ..., (d), (e').*

Proof. We will use the same idea as that from the proof of Theorem 3.1: we will construct all the paths in the graph starting with the node 0 and of length n, and then we will verify that they go through all the nodes of the graph (in other words, they pass through a node exactly once, because the length of the path is n). The system outputs the symbol t if and only if a Hamiltonian path has been found.

As input we will have the graph $G = (V, E)$, where $V = \{0, 1, \ldots, n-2, n-1\}$, $2 \leq n$, is the set of vertices and E is the set of edges. The output of the P system will be the letter t at the moment $4n^2 + 2n - 3$ if and only if a Hamiltonian path starting with the node 0 exists in G.

The P system is $\Pi = (V, H, \mu, w_0, w_1, \ldots, w_{n-1}, w_n, R)$, with the following components:

$$
\begin{aligned}
V &= \{i, b_i, b_i', b_i'', u_i \mid 0 \leq i \leq n-1\} \cup \{c_i \mid 0 \leq i \leq 2n-2\} \\
&\cup \{u_{j_1 \ldots j_k} \mid 2 \leq k \leq n-2,\ 0 \leq j_1, \ldots, j_k \leq n-1\} \cup \{d_i \mid 0 \leq i \leq 2n+1\} \\
&\cup \{y, z, z_0, z_1, t, \#\}, \\
H &= \{0, 1, \ldots, n-1, n, n+1\}, \\
\mu &= [_{n+1}[_n[_{n-1} \cdots [_1[_0\ ]_0^0]_1^0 \cdots]_{n-1}^0]_n^0]_{n+1}^0, \\
w_0 &= u_0 c_0, \\
w_i &= \lambda, \text{ for all } i = 1, 2, \ldots, n+1,
\end{aligned}
$$

and the set R contains the following rules:

0) $[_0 c_i \to c_{i+1}]_0^0$, for $0 \leq i < 2n-2$,

0′) $[_0 c_{2n-2}]_0^0 \to t$,

1) $[_0 u_i \to i b_i'']_0^0$, for $1 \leq i \leq n-1$,

1′) $[_0 u_0 \to b_0'']_0^0$,

2) $[_0 b_i'']_0^0 \to b_i'[_0]_0^+$, for $0 \leq i \leq n-1$,

2′) $[_j b_i']_j^0 \to b_i'[_j]_j^0$, for $1 \leq j \leq n-1$, $0 \leq i \leq n-1$,

3) $[_n b_i' \to b_i d_0]_n^0$, for $0 \leq i \leq n-1$,

4) $[_n d_0]_n^0 \to [_n]_n^+$,

5) $[_{n+1} d_i \to d_{i+1}]_{n+1}^0$, for $0 \leq i \leq 2n$,

6) $[_n b_i]_n^+ \to [_n u_{j_1}]_n^+ [_n u_{j_2 \ldots j_k}]_n^+$,
$[_n u_{j_2 \ldots j_k}]_n^+ \to [_n u_{j_2}]_n^+ [_n u_{j_3 \ldots j_k}]_n^+$,
.....
$[_n u_{j_{k-1} j_k}]_n^+ \to [_n u_{j_{k-1}}]_n^+ [_n u_{j_k}]_n^+$,

where $j_1, \ldots, j_k$ are the nodes adjacent with the node i in the graph G,

6') $[_n b_i \to u_j]^+_n$, if j is the only node adjacent with i in G,

7) $u_i[_j]^\alpha_j \to [_j u_i]^\alpha_j$, for $0 \leq i, j \leq n-1, \alpha \in \{0, +\}$

8) $[_{n+1} d_{2n+1} \to y^{n-1} z_0 z]^0_{n+1}$,

9) $y[_i]^\alpha_i \to [_i y]^0_i$, for $\alpha \in \{+, 0\}, 0 \leq i \leq n$,

10) $[_{n+1} z]^0_{n+1} \to \#[_{n+1}]^+_{n+1}$,

11) $[_{n+1} y \to \#]^+_{n+1}$,

11') $[_{n+1} z \to \#]^+_{n+1}$,

12) $[_{n+1} z_0 \to z_1]^+_{n+1}$,

13) $[_{n+1} z_1]^+_{n+1} \to \#[_{n+1}]^0_{n+1}$,

13') $[_{n+1} z_1 \to \#]^0_{n+1}$,

14) $[_i i]^0_i \to i$, for $1 \leq i \leq n-1$,

15) $[_n t]^0_n \to t[_n]^+_n$,

16) $[_{n+1} t]^0_{n+1} \to t[_{n+1}]^+_{n+1}$.

The computation starts in membrane 0 with a rule of type 1 (or 1' for u_0) and a rule of type 0 (which increments the counter c_i) applied in parallel. At the next step b_i'' exits the membrane 0 making it positive (rule 2) and again rule 0 is applied. Because the membrane changed the polarization, no other rule can be applied in the membrane 0 while b_i' exits the membranes $1, 2, \ldots, n-1$ (rule 2').

Reaching membrane n, b_i' in transformed in $b_i d_0$ (rule 3), the next rule applied is rule 4: d_0 exits the membrane n making it positive. The membrane structure at this moment is:

$$\mu = [_{n+1}[_n[_{n-1} \cdots [_1[_0\,]^+_0]^0_1 \cdots]^0_{n-1}]^+_n]^0_{n+1}.$$

Now the rules of type 6 can start to be applied in the same time with rules of type 5 (the timer d_i), so in one step we get the membrane structure:

$$\mu = [_{n+1}[_n[_{n-1} \cdots [_1[_0]^+_0]^0_1 \cdots]^0_{n-1}]^+_n [_n[_{n-1} \cdots [_1[_0]^+_0]^0_1 \cdots]^0_{n-1}]^+_n]^0_{n+1}.$$

Using rules of type 7 the u_i-s go "deeper" into the membrane structure until each of them arrives in a membrane of type 0; because of the way they are produced, no two u_i-s can be found in the same membrane. So, after $2n-2$ steps this "branch" of the computation stops (u_i-s have reached membranes 0), and after $2n+1$ steps we reach d_{2n+1}, so we can apply rule 8, generating z, z_0 in the membrane $n+1$ together with $n-1$ copies of y. The next step the y-s go into the membranes n changing their polarity from + to 0 (rule 9); at the same time, z exits the membrane $n+1$ and change its polarity from 0 to +. Thus, at the next step all the remaining y-s and z-s will transform into the trap symbol # (rules 11, 11'). Then, z_1 exits the membrane $n+1$ making it neutral again (rule 13) and in the end we apply the rule 13' to get rid of the remaining z_1-s. In $n+1$ steps the y-s will reach all the 0 membranes, making them neutral again, so another cycle of computation can start with rules 0 and 1. We perform $n-1$ such cycles; when c_i reaches the value c_{2n-2}, rule 0' is applied and it dissolves all the 0 membranes. In this way, only the rules of type 14 can be applied the next $n-1$ steps, and after that if t reaches membrane n we can apply rules 15 and 16.

One can see that the time required by the system to output t is $(4n+5)(n-1)+n+2 = 4n^2+2n-3$. □

5 Universality without Membrane Division

In this section we will show that P systems with active membranes using only rules of types (a), (b), and (c) are computationally universal. This improves the best result known in the literature, Theorem 1 in [9], where rules of types (a), (b), (c), and (e) were shown to be sufficient.

We denote with NPA_r the family of vectors of natural numbers $N(\Pi)$ computed by systems Π which use only rules of types (a), (b) and (c); in other words, we don't use dissolving or division rules. By $PsRE$ we denote the set family of recursively enumerable sets of vectors of natural numbers; clearly, this is the same with the family of Parikh images of recursively enumerable languages.

Theorem 5.1 $PsRE = NPA_r$.

Proof. We will only prove the inclusion $PsRE \subseteq NPA_r$ and to this aim we use the fact that each recursively enumerable language can be generated by a matrix grammar with appearance checking in the binary normal form.

Let $G = (N, T, S, M, F)$ be such a grammar, that is, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and with matrices in M either of the form $(X \to \alpha, A \to x)$, for $X \in N_1, \alpha \in N_1 \cup \{\lambda\}, A \in N_2, x \in (N_2 \cup T)^*$, or of the form $(X \to Y, A \to \#)$, for $X, Y \in N_1, A \in N_2$; moreover, a unique matrix of the form $(S \to XA)$ exists, with $X \in N_1, A \in N_2$. Each matrix of the form $(X \to \lambda, A \to x)$ is replaced by $(X \to Z, A \to x)$, where Z is a new symbol. Assume that we have n_1 matrices of the form $(X \to Y, A \to x)$, with $X \in N_1, Y \in N_1 \cup \{Z\}, x \in (N_2 \cup T)^*$, and n_2 matrices of the form $(X \to Y, A \to \#)$, $X, Y \in N_1, A \in N_2$.

We construct the P system (with $p = n_1 + n_2 + 1$ membranes) $\Pi = (V, H, \mu, w_1, \ldots, w_p, R)$, with

$$
\begin{aligned}
V &= N_1 \cup N_2 \cup T \cup \{Z, \#\} \cup \{X', X'' \mid X \in N_1\} \\
&\quad \cup \{\langle x \rangle \mid x \in (N_2 \cup T)^*, (X \to Y, A \to x) \text{ is a matrix in } G'\}, \\
H &= \{1, 2, \ldots, p\}, \\
\mu &= [_p [_1]_1^0 [_2]_2^0 \cdots [_{n_1}]_{n_1}^0 [_{n_1+1}]_{n_1+1}^0 \cdots [_{n_1+n_2}]_{n_1+n_2}^0]_p^0, \\
w_i &= \lambda, \text{ for all } i \in H - \{p\}, \\
w_p &= XA, \text{ for } (S \to XA) \text{ being the initial matrix of } G,
\end{aligned}
$$

and the following set R of rules:

1. For each matrix $m_i = (X \to Y, A \to x)$, $1 \le i \le n_1$, we introduce the rules:
   $X[_i]_i^0 \to [_i Y]_i^+$,
   $[_i Y \to Y]_i^+$,
   $A[_i]_i^+ \to [_i A]_i^0$,
   $[_i A]_i^0 \to \langle x \rangle [_i]_i^0$,
   $[_i Y]_i^0 \to Y' [_i]_i^0$,
   $[_p \langle x \rangle \to x]_p^0$,
   $[_p Y' \to Y]_p^0$.

2. For each matrix $m_i = (X \to Y, A \to \#)$, $n_1 + 1 \leq i \leq n_1 + n_2$, we introduce the rules:
$X[_i\ ]_i^0 \to [_i Y]_i^+$,
$[_i Y \to Y']_i^+$,
$[_i Y' \to Y'']_i^+$,
$A[_i\ ]_i^+ \to [_i A]_i^0$,
$[_i A]_i^0 \to \#[_i]_i^0$,
$[_i Y'']_i^+ \to Y[_i]_i^0$.
3. To finish the simulation we need the following rules:
$[_p Z]_p^0 \to Z[_p]_p^+$,
$[_p a]_p^+ \to a[_p\ ]_p^+$, for $a \in T$,
as well as the following "trap" rules:
$[_p A \to \#]_p^+$, for $A \in N_2$
$[_p \# \to \#]_p^\alpha$, for $\alpha \in \{0, +\}$.

The system works as follows.

Assume that at a given moment in membrane p we have a multiset Xw, for $X \in N_1$ and $w \in (N_2 \cup T)^*$. Initially, we have $w = A$, for $(S \to XA)$ being the starting matrix of G.

As long as symbols X from N_1 are present, the computation is not finished, a rule $X[_i]_i^0 \to [_i Y]_i^+$ can be used. That is, we have to remove all nonterminal symbols from membrane p that are from N_1 and this is done by simulating a derivation in G, in the following way.

The unique copy of X will go to a membrane i. Assume that we have $1 \leq i \leq n_1$, hence corresponding to a matrix $m_i : (X \to Y, A \to x)$ from G. Inside membrane i we have Y and the membrane gets a positive charge. The rule $[_i Y \to Y]_i^+$ can be used forever. The way to avoid this is to also send to membrane i the symbol A. The arriving of the right symbol A changes the polarity of membrane i to 0, which allows both the symbols Y and A to be sent out of this membrane as the symbols Y' and $\langle x \rangle$, respectively. In membrane p, the first one is replaced by Y and the latter one is replaced by the multiset x. In this way, the matrix m_i was simulated. Note that even if Y' is sent out before $\langle x \rangle$, the symbol Y is available only at the next step, hence we can start simulating a new matrix only after completing the simulation of m_i.

Assume now that the symbol X was sent into a membrane i with $n_1 + 1 \leq i \leq n_1 + n_2$, that is, corresponding to a matrix $m_i : (X \to Y, A \to \#)$. The polarity of the membrane is changed (and x is changed into Y). If there is at least a copy of the nonterminal A in the membrane p, then it will enter the membrane i using the rule $A[_i]_i^+ \to [_i A]_i^0$ at the same time as Y becomes Y' using the rule $[_i Y \to Y']_i^+$. Notice that if the symbol A enters the membrane i, then the polarity of that membrane is changed to neutral. If this doesn't happen (the polarity of the membrane remains positive), then we can transform Y' in Y'' (using the rule $[_i Y' \to Y'']_i^+$) and Y'' can exit the membrane, by using the rule $[_i Y'']_i^+ \to Y[_i]_i^0$ (the membrane returns to the neutral polarity while outside it we have the symbol Y). If A has entered membrane i, then it will send out the trap-symbol $\#$ and the computation will continue forever by using the rule $[_p \# \to \#]_p^0$. Thus, we can continue correctly only if the appearance checking application of the matrix is correctly simulated.

These operations can be iterated, hence any derivation in G can be simulated by a computation in Π and, conversely, the computations in Π correspond to correct derivations in G.

When the object Z has been produced, the process stops – providing that no element of N_2 is still present. This means that the derivation in G is terminal. This is done because the symbol Z exits the system and changes the polarization of the p membrane to positive (rule $[_pZ]_p^0 \to Z[_p]_p^+$). In this moment the nonterminals A from N_2 are transformed in the trap symbol #: $[_pA \to \#]_p^+$, for $A \in N_2$, and then the computation never stops because of the rule $[_p\# \to \#]_p^\alpha$, for $\alpha \in \{0, +\}$.

The terminal symbols $a \in T$ can exit the system using the rule $[_pa]_p^+ \to a[_p\,]_p^+$, for $a \in T$ (the membrane p will become positive only after the symbol Z exited the system). □

One can see that in our construction we used only two (out of three possible) polarities for the membranes (neutral and positive).

It is worth noticing that a system can be constructed that outputs only terminal symbols and only when the computation was successful. To this aim, we can use one further membrane, $p+1$, which will contain the membrane p, and which, in the presence of the symbol Z will allow to leave the system only the terminal symbols. The details are left to the reader.

6 Final Remarks

The question whether or not the systems with n-bounded division can be simulated in linear/polynomial time by systems with 2-bounded division still remains *open*. It is also *open* whether or not HPP can be solved in linear time by P systems with 2-bounded division. It would be also of interest to bound the number of membranes in the universality result; in our proof, the number of membranes depends on the number of matrices in the matrix grammar we start with, but we believe that systems with a small number of membranes are still computationally complete.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science*, 226 (1994), 1021 – 1024.

[2] J. Dassow, Gh. Păun, On the power of membrane computing, *Journal of Universal Computer Science*, **5**, 2 (1999), 33–49.

[3] R. Freund, Generalized P systems, *Fundamentals of Computation Theory, FCT'99*, Iaşi, 1999, (G. Ciobanu, Gh. Păun, eds.), *LNCS* 1684, Springer, 1999, 281–292.

[4] S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.

[5] C. Martin-Vide, V. Mitrana, P systems with valuation, submitted, 2000.

[6] Gh. Păun, Computing with membranes, *J. of Computer and System Sciences*, 61 (2000).

[7] Gh. Păun, P systems with active membranes: Attacking NP complete problems, *J. of Automata, Languages and Combinatorics*, to appear.

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

[9] Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems, *Proc. Conf. on Words, Languages, and Combinatorics*, Kyoto, 2000.

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

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