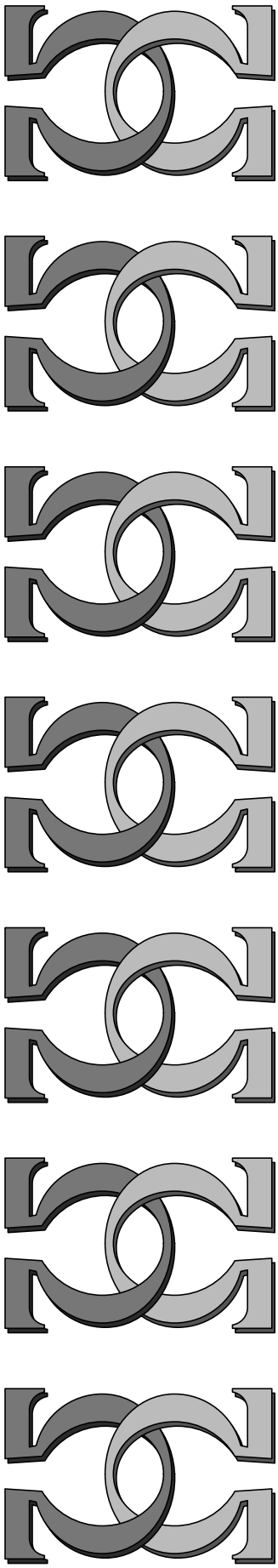

CDMTCS
Research
Report
Series

P Systems with Active Membranes: Attacking NP Complete Problems

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy, Bucureşti, Romania

CDMTCS-102
May 1999

Centre for Discrete Mathematics and
Theoretical Computer Science

P Systems with Active Membranes: Attacking NP Complete Problems*

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro

Abstract

P systems are parallel Molecular Computing models based on processing multisets of objects in cell-like membrane structures. Various variants were already shown to be computationally universal, equal in power to Turing machines. In this paper one proposes a class of P systems whose membranes are the main active components, in the sense that they directly mediate the evolution and the communication of objects. Moreover, the membranes can multiply themselves by division. We prove that this variant is not only computationally universal, but also able to solve NP complete problems in polynomial (actually, linear) time. We exemplify this assertion with the well-known SAT problem.

1 Introduction: The Basic Variants of P Systems

The P systems are a class of distributed parallel computing devices of a biochemical type, introduced in [5], which can be seen as a general computing architecture where various types of objects can be processed by various operations.

Very shortly, in the basic model one considers a *membrane structure* consisting of several cell-like membranes which are hierarchically embedded in a main membrane, called the *skin* membrane. The membranes delimit *regions*, where we place *objects*.

The objects evolve according to given *evolution rules*, which are associated with the regions. A rule is applied to objects in the region with which it is associated and can modify the objects, send them outside the current membrane or to an inner membrane, and can also dissolve the membrane. When such an action takes place, all the objects of the dissolved membrane remain free in the membrane placed immediately outside, but the evolution rules of the dissolved membrane are removed. The skin membrane is never dissolved. Note that the membranes are both separators and channels of communication, but they are passive participants to the process, the whole functioning of the system is governed by the evolution rules.

*Research supported by "Research for Future" Program no. JSPS-RFTF 96I00101, from the Japan Society for the Promotion of Science.

The application of evolution rules is done in a maximally parallel manner: at each step, all objects which can evolve should evolve.

Starting from an initial configuration and using the evolution rules, we get a *computation*. A computation is considered completed when it halts, no further rule can be applied to the objects present in the last configuration. There are two possible ways of assigning a result to a computation: by considering the multiplicity of objects present in a designated membrane in a halting configuration, or by concatenating the symbols which leave the system, in the order they are sent out of the skin membrane (if several symbols are expelled at the same time, then any ordering of them is accepted). Thus, in the first case we compute vectors of natural numbers, while in the second case we generate a language.

Many variants are considered in [2], [5], [6], [8], [9], [10], [11], [12]. In all of these variants the number of membranes can only decrease during a computation, by dissolving membranes as a result of applying evolution rules to the objects present in the system.

A natural possibility is to let the number of membranes also to increase during a computation, for instance, by division, as it is well-known in biology. Actually, the membranes from biochemistry are not at all passive, like those in the models briefly described above. For example, the passing of a chemical compound through a membrane is often done by a direct interaction with the membrane itself (with the so-called *protein channels* or *protein gates* present in the membrane); during this interaction, the chemical compound which passes through membrane can be modified, while the membrane itself can in this way be modified (at least locally).

We will here make use of these observations and we will consider P systems where the central role in the computation is played by the membranes: evolution rules are associated both with objects and membranes, while the communication through membranes is performed with the direct participation of the membranes; moreover, the membranes can not only be dissolved, but they also can multiply by *division*. An elementary membrane can be divided by means of an interaction with an object from that membrane. Each membrane is supposed to have an "electrical polarization" (we will say *charge*), one of the three possible: *positive, negative*, or *neutral*. If in a membrane we have two immediately lower membranes of opposite polarizations, one *positive* and one *negative*, then that membrane can also divide in such a way that the two membranes of opposite charge are separated; all membranes of neutral charge and all objects are duplicated and a copy of each of them is introduced in each of the two new membranes. The skin is never divided.

In this way, the number of membranes can grow, even exponentially. As expected, by making use of this increased parallelism we can compute faster. We prove that this is the case, indeed: the SAT (satisfiability of propositional formulas in the conjunctive normal form) problem, one of the basic NP complete problems, can be solved in this framework in linear time (the time units are steps of a computation in a P system as sketched above, where we perform in parallel, in all membranes of the system, applications of evolution rules or division of membranes). Moreover, the model is shown to be computationally universal: any recursively enumerable set of (vectors of) natural numbers can be generated by our systems.

These two features – the computational universality and the possibility of solving NP complete problems in linear time – looks very promising for considering P systems

with active membranes as a computing model. However, it is still prematurely to say whether or not such a model can be implemented, in biochemical media or in electronic media. Anyway, we advocate that together with DNA Computing (see details in [7]), Computing with Membranes is an appealling area of Natural Computing which deserves further investigations.

2 P Systems with Active Membranes

We directly define the variant of P systems we introduce in this paper; we start by fixing a few notions and notations.

A *membrane structure* is represented by a Venn diagram and is identified by a string of correctly matching parentheses, with a unique external pair of parentheses; this external pair of parentheses corresponds to the external membrane, called *the skin*. A membrane without any other membrane inside is said to be *elementary*. For instance, the structure in Figure 1 contains 8 membranes; membranes 3, 5, 6, and 8 are elementary. The string of parentheses identifying this structure is

$$\mu = [_1[_2[_5\ ]_5[_6\ ]_6]_2[_3\ ]_3[_4[_7[_8\ ]_8]_7]_4]_1.$$

All membranes are labeled; here we have used the numbers from 1 to 8. (We sometimes say that the number of membranes is the *degree* of the membrane structure, while the height of the tree associated in the usual way with the structure is its *depth*. In the example above we have a membrane structure of degree 8 and of depth 4.)

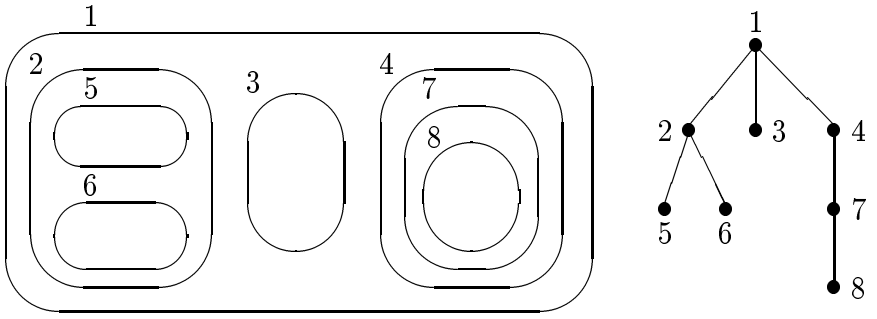

Figure 1. A membrane structure and its associated tree.

In what follows, the membranes can be marked with $+$ or $-$, and this is interpreted as an "electrical charge", or with 0, and this means "neutral charge". We will write $[_i\ ]_i^+, [_i\ ]_i^-, [_i\ ]_i^0$ in the three cases, respectively.

The membranes delimit *regions*, precisely identified by the membranes (the region of a membrane is delimited by the membrane and all membranes placed immediately inside it, if any such a membrane exists). In these regions we place *objects*, which are represented by symbols of an alphabet. Several copies of the same object can be present in a region, so we work with *multisets* of objects. A multiset over an alphabet V is represented by a string over V: the number of occurrences of a symbol $a \in V$ in a string $x \in V^*$ (V^* is

the set of all strings over V; the empty string is denoted by λ) is denoted by $|x|_a$ and it represents the multiplicity of the object a in the multiset represented by x.

Let us now pass to defining our systems.

A P system with active membranes is a construct

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

where:

(i) $m \geq 1$;

(ii) V is an alphabet (the *total alphabet* of the system);

(iii) $T \subseteq V$ (the *terminal* alphabet);

(iv) H is a finite set of *labels* for membranes;

(v) μ is a *membrane structure*, consisting of m membranes, labeled (not necessarily in a one-to-one manner) with elements of H; all membranes in μ are supposed to be neutral;

(vi) $w_1, \ldots, w_m$ are strings over V, describing the *multisets of objects* placed in the m regions of μ;

(vii) R is a finite set of *developmental rules*, of the following forms:

(a) $[_h a \to v]^{\alpha}_h$,
for $h \in H, \alpha \in \{+, -, 0\}, a \in V, v \in V^*$
(object evolution rules, associated with membranes and depending on the label and the charge of the membrane, but not directly implying the membranes, in the sense that the membranes are neither taking part to the application of these rules nor are they modified by them);

(b) $a[_h\,]^{\alpha_1}_h \to [_h b]^{\alpha_2}_h$,
for $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a, b \in V$
(communication rules; an object is introduced in the membrane, maybe modified during this process; also the polarization of the membrane can be modified, but not its label);

(c) $[_h a\,]^{\alpha_1}_h \to [_h\,]^{\alpha_2}_h b$,
for $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a, b \in V$
(communication rules; an object is sent out of the membrane, maybe modified during this process; also the polarization of the membrane can be modified, but not its label);

(d) $[_h a\,]^{\alpha}_h \to b$,
for $h \in H, \alpha \in \{+, -, 0\}, a, b \in V$
(dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

(e) $[_h a\,]_h^{\alpha_1} \to [_h b\,]_h^{\alpha_2} [_h c\,]_h^{\alpha_3}$,
for $h \in H, \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}, a, b, c \in V$
(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);

(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 should have neutral charge in order to may apply this rule
(division of non-elementary membranes; this is possible only if a membrane contains two immediately lower 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).

Note that in all rules of types (a) – (e) only one object is specified (that is, the objects do not directly interact) and that, with the exception of rules of type (a), always single objects are transformed into single objects (the two objects produced by a division rule of type (e) are placed in two different regions).

These rules are applied according to the following *principles*:

1. All the rules are applied in parallel: in a step, the rules of type (a) are applied to all objects to which they can be applied, all other rules are applied to all membranes to which they can be applied; an object can be used by only one rule, non-deterministically choosen (there is no priority relation among rules), but any object which can evolve by a rule of any form, should evolve.

2. If a membrane is dissolved, then all the objects in its region are left free in the region immediately above it. Because all rules are associated with membranes, the rules of a dissolved membrane are no longer available at the next steps. The skin membrane is never dissolved.

3. All objects and membranes not specified in a rule and which do not evolve are passed unchanged to the next step. For instance, if a membrane with the 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 two resulting membranes h. Similarly, when dividing a membrane h by means of a rule of type (f), the neutral membranes are reproduced in each of the two new membranes with the label h, unchanged if no rule is applied to them (in particular, the contents of these neutral membranes is reproduced unchanged in these copies, providing that no rule is applied to their objects).

4. If at the same time a membrane h is divided by a rule of type (e) and there are objects in this membrane which evolve by means of rules of type (a), then in the

new copies of the membrane we introduce the result of the evolution; that is, we may suppose that first the evolution rules of type (a) are used, changing the objects, and then the division is produced, so that in the two new membranes with label h we introduce copies of the changed objects. Of course, this process takes only one step. The same assertions apply to the division by means of a rule of type (f): always we assume that 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.

5. The rules associated with a membrane h are used for all copies of this membrane, irrespective whether or not the membrane is an initial one or it is obtained by division. At one step, a membrane h can be the subject of only one rule of types (b) – (f).

6. The skin membrane can never divide. As any other membrane, the skin membrane can be "electrically charged".

The membrane structure of the system at a given time, together with all multisets of objects asociated with the regions of this membrane structure is the *configuration* of the system at that time. The $(m+1)$-tuple $(\mu, w_1, \ldots, w_m)$ is the *initial configuration*. We can pass from a configuration to another one by using the rules from R according to the principles given above. We say that we have a (direct) *transition* among configurations. We do not define formally a transition. In the next section we will consider in detail an example which can enlighten the idea; also the proof of the computational universality theorem from Section 4 will offer a series of explanations.

A sequence of transitions which starts from the initial configuration is called a *computation* with respect to Π. A computation is *complete* if it cannot be continued: there is no rule which can be applied to objects and membranes in the last configuration.

Note that during a computation the number of membranes (hence the degree of the system) can increase and decrease but the labels of these membranes are always among the labels of membranes present in the initial configuration (by division we only produce membranes with the same label as the label of the divided membrane).

During a computation, objects can leave the skin membrane (by means of rules of type (c)). The terminal symbols which leave the skin membrane are collected in the order of their expelling from the system, so a string is associated to a complete computation; when several terminal symbols leave the system at the same time, then any ordering of them is accepted (thus, with a complete computation we possibly associate a set of strings, due to this "local commutativity" of symbols which are observed outside the system at the same time). In this way, a language is associated with Π, denoted by $L(\Pi)$, consisting of all strings which are associated with all complete computations in Π.

Two facts are worth emphasizing: (1) the symbols not in T which leave the skin membrane as well as all symbols from T which remain in the system at the end of a halting computation are not considered in the generated strings; (2) if a computation goes for ever, then it provides no output, it does not contribute to the language $L(\Pi)$.

3 Solving SAT in Linear Time

In order to prove the usefulness of using active membranes (in particular, membrane division) and in order to illuminate the informal definition of a transition in a P system as given above, we will consider an example which is also very significant by itself: solving the SAT problem by a P system with active membranes.

The SAT (satisfiability of propositional formulas in the conjunctive normal form) is probably the most known NP complete problem. It asks whether or not for a given formula in the conjunctive normal form there is a truth-assignment of the variables for which the formula assumes the value *true*. A formula as above is of the form

$$\gamma = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

where each $C_i, 1 \leq i \leq n$, is a *clause* of the form of a disjunction

$$C_i = y_1 \vee y_2 \vee \ldots \vee y_r,$$

with each y_j being either a propositional variable, x_s, or its negation, $\sim x_s$. (Thus, we use the variables $x_1, x_2, \ldots$ and the three connectives $\vee, \wedge, \sim$: *or, and, negation.*)

For example, let us consider the propositional formula

$$\beta = (x_1 \vee x_2) \wedge (\sim x_1 \vee \sim x_2)$$

(it is also used in [4], where a linear time DNA Computing solution to SAT is proposed). We have two variables, x_1, x_2, and two clauses. It is easy to see that it is satisfiable: any of the following truth-assignments makes the formula *true*

$$(x_1 = \mathit{true}, x_2 = \mathit{false}), \quad (x_1 = \mathit{false}, x_2 = \mathit{true}).$$

We now pass to one of the main results of this paper:

Theorem 3.1 *The SAT problem can be solved by a P system with active membranes in a time which is linear in the number of variables and the number of clauses.*

Proof. Let us consider a propositional formula

$$\gamma = C_1 \wedge C_2 \wedge \ldots \wedge C_m,$$

with

$$C_i = y_{i,1} \vee \ldots \vee y_{i,p_i},$$

for some $m \geq 1, p_i \geq 1$, and $y_{i,j} \in \{x_k, \sim x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m, 1 \leq j \leq p_i$.

We construct the P system

$$\Pi = (V, T, H, \mu, w_0, w_1, \ldots, w_m, w_{m+1}, R)$$

with the components

$$\begin{aligned}
V &= \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \\
&\cup \{c_i \mid 0 \leq i \leq 2n+m-1\} \\
&\cup \{t\}, \\
T &= \{t\}, \\
H &= \{0, 1, \ldots, m+1\}, \\
\mu &= [_{m+1}[_m[_{m-1} \ldots [_1[_0 \,]_0^0]_1^0 \ldots]_{m-1}^0]_m^0]_{m+1}^0, \\
w_0 &= c_0 a_1 a_2 \ldots a_n, \\
w_i &= \lambda, \text{ for all } i = 1, 2, \ldots, m+1,
\end{aligned}$$

while the set R contains the following rules:

1. $[_0 c_i \to c_{i+1}]_0^\alpha$, for all $0 \leq i \leq 2n+m-2$ and $\alpha \in \{+, -, 0\}$

   (we count to $2n+m-1$, which is the time needed for producing all 2^n truth-assignments for the n variables, as well as 2^n membrane sub-structures which will examine the truth value of formula γ for each of these truth-assignments; this counting is done in the central membrane, irrespective which is its polarity);

2. $[_0 a_i]_0^0 \to [_0 t_i]_0^+ [_0 f_i]_0^-$, for all $1 \leq i \leq n$

   (in membrane 0, when it is "electrically neutral", we non-deterministically choose one variable x_i and both values *true* and *false* are associated with it, in the form of objects t_i, f_i, which are separated in two membranes with the label 0 which differ only by these objects t_i, f_i and by their charge);

3. $[_0 c_{2n+m-1}]_0^0 \to t$

   (after $2n+m-1$ steps, each copy of membrane 0 is dissolved and their contents is released in the upper membranes, those labeled with 1);

4. $[_j t_i]_j^0 \to t_i$, if x_i appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$, and

   $[_j f_i]_j^0 \to f_i$, if $\sim x_i$ appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$

   (a membrane with label j, $1 \leq j \leq m$, is dissolved if and only if clause C_j is satisfied by the current truth-assignment; if this is the case, then the truth values associated with the variables are released in the upper membrane, that associated with the next clause, C_{j+1}, otherwise these truth values remain blocked in membrane j and never used at the next steps by the membranes placed above; note that, as we will see immediately, after $2n+m-1$ steps we have 2^n membrane sub-structures of the form $[_m[_{m-1} \ldots [_1 \,]_1^0 \ldots]_{m-1}^0]_m^0$ working in parallel in the skin membrane);

5. $[_{m+1} t]_{m+1}^0 \to [_{m+1} \,]_{m+1}^+ t$

(together with the truth-assignments, we also have the object t, which can be passed from a level to the upper one only by dissolving membranes; this object reaches the skin membrane if only if all membranes in a sub-structure of the form $[_m[_{m-1}\dots[_1\ ]_1^0\dots]_{m-1}^0]_m^0$ are dissolved, which means that the associated truth-assignment has satisfied all the clauses, that is, the formula is satisfiable; therefore, t leaves the system if and only if the formula is satisfiable; when this rule is applied, the skin membrane gets a "positive charge", so the rule can be applied only once);

6. $[_{i+1}[_i\ ]_i^+[_i\ ]_i^-]_{i+1}^0 \to [_{i+1}[_i\ ]_i^0]_{i+1}^+[_{i+1}[_i\ ]_i^0]_{i+1}^-$, for all $0 \leq i \leq m-2$, and

   $[_m[_{m-1}\ ]_{m-1}^+[_{m-1}\ ]_{m-1}^-]_m^0 \to [_m[_{m-1}\ ]_{m-1}^0]_m^0[_m[_{m-1}\ ]_{m-1}^0]_m^0$

   (division rules for membranes labeled with $0, 1, \dots, m$; the opposite polarization introduced when dividing a membrane 0 is propagated from lower levels to upper levels of the membrane structure and the membranes are continuously divided until dividing also membrane m – which will get neutral charge).

¿From the previous explanations one can easily see that

$$L(\Pi) = \begin{cases} \{t\}, & \text{if formula } \gamma \text{ is satisfiable,} \\ \emptyset, & \text{otherwise.} \end{cases}$$

Therefore, we get the answer to the question whether or not γ is satisfiable by examining the emptiness of the language $L(\Pi)$ (by checking whether or not any object leaves the system Π during the computation). This is achieved in $2n + 2m + 1$ steps: in $2n + m - 1$ steps we create the 2^n membrane sub-structures of the form $[_m[_{m-1}\dots[_1\ ]_1^0\dots]_{m-1}^0]_m^0$ (as well as the 2^n different truth-assignments), then we dissolve all membranes 0 (one further step) and we check the satisfiability of each clause for each truth-assignment, in parallel in the 2^n sub-structures (this takes further m steps); one more step is necessary in order to send out of the system one copy of the object t, if any copy of t has reached the skin membrane. If no copy of t leaves the system at this step, then the formula is not satisfiable. □

Note that we have used rules of all types (a) – (f), excepting the type (b), and that also in the case of rules of type (a) we have object-to-object rules (and not object-to-multiset).

We illustrate the previous construction and the work of the system Π obtaineed in this way for the propositional formula $\beta = (x_1 \vee x_2) \wedge (\sim x_1 \vee \sim x_2)$, mentioned before the theorem.

Thus, $n = 2, m = 2$. The initial configuration of the system is

$$[_3[_2[_1[_0 c_0 a_1 a_2]_0^0]_1^0]_2^0]_3^0.$$

The computation proceeds as follows (we specify the current configuration at each step):

Step 0: $[_3[_2[_1[_0 c_0 a_1 a_2]_0^0]_1^0]_2^0]_3^0$;

Step 1: $[_3[_2[_1[_0 c_1 t_1 a_2]_0^+[_0 c_1 f_1 a_2]_0^-]_1^0]_2^0]_3^0$

(in parallel, the rule $[_0 c_0 \to c_1]_0^0$ and the division rule $[_0 a_1]_0^0 \to [_0 t_1]_0^+ [_0 f_1]_0^-$ were used; membrane 1 must immediately divide, because of the two copies of membrane 0 with opposite polarizations);

Step 2: $[_3[_2[_1[_0 c_2 t_1 a_2]_0^0]_1^+[_1[_0 c_2 f_1 a_2]_0^0]_1^-]_2^0]_3^0$

(the counter c_1 is replaced with c_2 and membrane 1 is divided; because the two membranes with label 0 are not of neutral polarity, no new truth value is introduced at this step; at the next step, membrane 2 must divide);

Step 3: $[_3[_2[_1[_0 c_3 t_1 t_2]_0^+[_0 c_3 t_1 f_2]_0^-]_1^0]_2^0[_2[_1[_0 c_3 f_1 t_2]_0^+[_0 c_3 f_1 f_2]_0^-]_1^0]_2^0]_3^0$

(simultaneously, the counter c_2 is replaced by c_3, each membrane 0 is divided again, producing membranes of opposite polarity, and membrane 2 is also divided, because of the existence of the two membranes 1 with opposite polarity; the generation of the truth-assignments is completed, but we still have to divide membranes, because of the existence of membranes of opposite polarity);

Step 4: $[_3[_2[_1[_0 c_4 t_1 t_2]_0^0]_1^+[_1[_0 c_4 t_1 f_2]_0^0]_1^-]_2^0[_2[_1[_0 c_4 f_1 t_2]_0^0]_1^+[_1[_0 c_4 f_1 f_2]_0^0]_1^-]_2^0]_3^0$

(we divide the two membranes 1, in parallel, and we increase the counter; no other rule can be applied);

Step 5: $[_3[_2[_1[_0 c_5 t_1 t_2]_0^0]_1^0]_2^0[_2[_1[_0 c_5 t_1 f_2]_0^0]_1^0]_2^0[_2[_1[_0 c_5 f_1 t_2]_0^0]_1^0]_2^0[_2[_1[_0 c_5 f_1 f_2]_0^0]_1^0]_2^0]_3^0$

(we increase again the counter and we divide the two membranes 2).

The membrane structure (and the contents of membranes with label 0) is represented in Figure 2. One sees that for each truth-assignment we have a sub-structure of the form $[_2[_1[_0\ ]_0]_1]_2$. All membranes have neutral polarity.

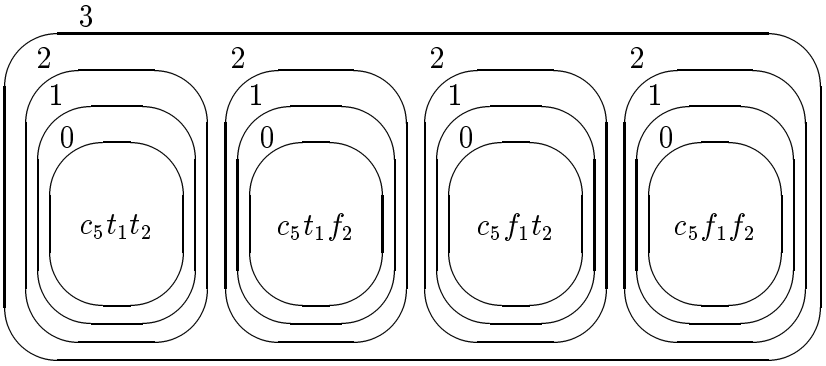

Figure 2. The membrane structure in the example, after Step 5.

Step 6: $[_3[_2[_1 t t_1 t_2]_1^0]_2^0[_2[_1 t t_1 f_2]_1^0]_2^0[_2[_1 t f_1 t_2]_1^0]_2^0[_2[_1 t f_1 f_2]_1^0]_2^0]_3^0$

(the counter has reached its maximal value; it can be transformed in t while dissolving all membranes 0);

Step 7: $[_3[_2 tt_1t_2]_2^0[_2 tt_1f_2]_2^0[_2 tf_1t_2]_2^0[_2[_1 tf_1f_2]_1^0]_2^0]_3^0$

(clause 1 is satisfied by the first three truth-assignments; the corresponding membranes with label 1 are dissolved; the last truth-assignment does not satisfy the first clause, its associated membrane 1 remains unchanged and the truth values in it will be of no use from now on);

Step 8: $[_3[_2 tt_1t_2]_2^0 tt_1f_2tf_1t_2[_2[_1 tf_1f_2]_1^0]_2^0]_3^0$

(clause 2 is satisfied by the second and the third truth-assignments, so the corresponding membranes 2 are dissolved; two copies of t are left free in the skin membrane, corresponding to the two truth-assignments which satisfy the formula);

Step 9: One of the two copies of t will be sent out of the system, so we know that the formula was satisfied. This is the last step of the computation, because no further rule can be applied (the skin membrane is now "positively charged", so the second copy of t cannot leave it).

4 The Computational Universality

We now investigate the computability power of P systems with active membranes.

Let us denote by LPA the family of languages $L(\Pi)$, generated by P systems with active membranes as (informally) defined in Section 2.

Because we work here with multisets of symbols, the "main information" contained by the strings in a language $L(\Pi)$ is given by the Parikh vectors associated with these strings, not by the ordering of symbols in the strings.

For a string $w \in V^*$, for $V = \{a_1, \ldots, a_n\}$, we define the *Parikh vector* of w (with respect to V, the ordering of elements from V being relevant) by

$$\Psi_V(w) = (|w|_{a_1}, \ldots, |w|_{a_n}).$$

The definition is extended in the usual way to languages, $\Psi_V(L) = \{\Psi_V(w) \mid w \in L\}, L \subseteq V^*$.

For a family FL of languages (over a given alphabet V) we denote by $PsFL$ the family of Parikh images of languages in FL.

By $PsRE$ we denote the family of recursively enumerable sets of vectors of natural numbers; this is equal to the family of Parikh sets of recursively enumerable languages. (For elements of formal language theory we refer to [13].)

In the proof of the main theorem below we will use the notion of a matrix grammar.

A *matrix grammar with appearance checking* is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (we say that N is the nonterminal

alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied; we say that these rules are applied in the *appearance checking* mode.) If $F = \emptyset$, then the grammar is said to be without appearance checking (and F is no longer mentioned).

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When we use only grammars without appearance checking, then the obtained family is denoted by MAT.

It is known that $MAT \subset MAT_{ac} = RE$ and that each one-letter language in the family MAT is regular, [3]. Further details about matrix grammars can be found in [1] and in [13].

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M are of one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
3. $(X \to Y, A \to \#)$, with $X, Y \in N_1, A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2$, and $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is a trap-symbol, once introduced, it is never removed. A matrix of type 4 is used only once, at the last step of a derivation.

According to Lemma 1.3.7 in [1], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

Theorem 4.1 (Computational Universality) $PsRE = PsLPA$.

Proof. The inclusion $LPA \subseteq RE$ follows from Church-Turing thesis or can be proved directly, in a straightforward (but involving a long construction) way. This implies the inclusion $PsLPA \subseteq PsRE$. So, we only have to prove the inclusion $PsRE \subseteq PsLPA$. To this aim, we make use of the equality $PsRE = PsMAT_{ac}$. Let $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and matrices of the four forms mentioned above. Each matrix of the form $(X \to \lambda, A \to x)$, $X \in N_1, A \in N_2, x \in T^*$, is replaced by $(X \to Z, A \to x)$, where Z is a new symbol. We denote by G' the obtained grammar. 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 $(X \to Y, A \to \#)$, $X, Y \in N_1, A \in N_2$. (That is, we consider separately the

matrices having rules used in the appearance checking mode and the matrices not having such rules.)

We construct the P system (with $p = n_1 + n_2 + 2$ initial membranes)

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

with

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

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:

$$\begin{aligned}
&X[_i\,]^0_i \to [_iY]^+_i, \\
&A[_i\,]^+_i \to [_iA]^-_i, \\
&[_iA]^-_i \to [_i\,]^0_i\langle x \rangle, \\
&[_iY]^0_i \to [_i\,]^0_i Y,
\end{aligned}$$

   as well as the rules

$$[_{p-1}\langle x \rangle \to x]^0_{p-1}.$$

2. For each matrix $m_i = (X \to Y, A \to \#)$, $n_1 + 1 \le i \le n_1 + n_2$, we introduce the rules:

$$\begin{aligned}
&X[_i\,]^0_i \to [_iX]^0_i, \\
&[_iX]^0_i \to [_i\bar{X}]^+_i[_iY]^-_i, \\
&[_{p-1}[_i\,]^+_i[_i\,]^-_i]^0_{p-1} \to [_{p-1}[_i\,]^+_i]^+_{p-1}[_{p-1}[_i\,]^-_i]^0_{p-1}, \\
&A[_i\,]^+_i \to [_i\,\dagger\,]^0_i, \\
&[_iY]^-_i \to [_i\,]^0_i Y, \\
&[_i\dagger \to \dagger]^0_i.
\end{aligned}$$

3. We also consider the following rules, for all $a \in T$,

$$\begin{aligned}
&[_{p-1}a]^0_{p-1} \to [_{p-1}\,]^0_{p-1}a, \\
&[_pa]^0_p \to [_p\,]^0_p a,
\end{aligned}$$

   as well as the following rules for all $\alpha \in N_1 \cup N_2$

$$[_{p-1}\alpha \to \alpha]^0_{p-1}.$$

The system works as follows.

Membranes labeled with $1, 2, \ldots, n_1$ are associated with matries not used in the appearance checking mode; each matrix is simulated with the help of the associated membrane. In any moment, in membrane $p-1$ there is only one symbol from the set N_1. If this symbol enters a membrane with the label i, $1 \leq i \leq n_1$, then the membrane gets the "electrical charge" + (initially, all membranes are neutral). We can continue only by introducing in this membrane i also the corresponding symbol A (at that time, the membrane gets a negative "electrical charge"). The continuation is deterministic: we send out of membrane i the symbol $\langle x \rangle$ (and the membrane is again neutral), then we send out also the symbol Y; at the second step, the symbol $\langle x \rangle$ is replaced in membrane $p-1$ by the string x. In this way, in membrane $p-1$ we have simulated the use of the matrix $m_i = (X \to Y, A \to x)$. Note how the "electrical charge" of the membrane controls the process and that the symbol Y is available in membrane $p-1$ only after completing the simulation of the matrix.

Membranes with labels $n_1 + 1, \ldots, n_1 + n_2$ are associated with matrices used in the appearance checking mode. Let $i, n_1 + 1 \leq i \leq n_1 + n_2$, be such a membrane, associated with $m_i = (X \to Y, A \to \#)$. As above, the symbol X can enter membrane i (unchanged); having this symbol inside, this membrane is divided in two membranes, of opposite polarity. In the first membrane, that with positive "charge", we check whether or not any occurrence of A is present in the membrane $p-1$. This is done as follows. Because of this opposite polarity of membranes with label i, membrane $p-1$ is also divided, in a copy of positive "charge" and a neutral copy. In this way, all objects from the former membrane with the label $p-1$ are duplicated and introduced in each of these membranes. If the symbol A is present, then at the same step when membrane $p-1$ is divided, we introduce the trap-object † in the positively "charged" copy of membrane $p-1$. This object will evolve for ever, so the computation will never finish. Also in parallel with membrane $p-1$ division, we release the symbol Y in the copy of membrane $p-1$ which is neutral. Thus, the simulation of the matrix m_i is successful if and only if the computation will ever stop, that is, if and only if the symbol A was not present.

The process can continue. Each terminal symbol present in the membrane with the label $p-1$ and of neutral polarity is sent to the skin membrane and from here it is sent out of the system. As long as any nonterminal symbol from $N_1 \cup N_2$ is present in the membrane with the label $p-1$ and of neutral polarity, the computation is not halting. (Note that the copies of membrane $p-1$ produced for simulating matrices in the appearance checking mode and used only for checking the non-appearance of symbols $A \in N_2$ have a positive charge, so the nonterminal symbols present in them do not evolve.) Consequently, we simulate in Π exactly the terminal derivations in G'. Because the symbol Z cannot evolve, we have the equality $\Psi_T(L(G)) = \Psi_T(L(\Pi))$. □

In the construction above we have used rules of all types (a) – (f), with the excception of dissolving rules of type (d).

Because the division of membranes is used only for simulating matrices which contain rules used in the appearance checking manner, from the previous construction we obtain the fact that the Parikh sets of languages in MAT are the same with the Parikh sets of languages generated by P systems which do not use membrane division. We denote by

$LPA(ndiv)$ this family of languages, hence we can write:

Corollary 4.1 $PsMAT \subseteq PsLPA(ndiv)$.

Actually, this is a proper inclusion, because of the following result:

Theorem 4.2 $LPA(ndiv) - MAT \neq \emptyset$.

Proof. We consider the P system

$$\Pi = (\{a, b\}, \{a\}, \{1\}, [_1\ ]_1^0, ab, R),$$
$$R = \{[_1 a \to aa]_1^0,\ [_1 b \to b]_1^0,\ [_1 b]_1^0 \to [_1 b]_1^+,\ [_1 a]_1^+ \to [_1\ ]_1^+ a\}.$$

As long as membrane 1 is neutral, the number of occurrences of object a is doubled. At the same time, the object b is "doing nothing". At any moment, object b can determine the change of the polarity of membrane 1 (it becomes positive). From now on, no other rule can be used than those which send out of the system all available copies of object a. Consequently, we have $L(\Pi) = \{a^{2^n} \mid n \geq 1\}$. This is not a language in the family MAT (it is a non-regular one-letter language). □

5 Final Remarks

The P systems were introduced as parallel computing models of a biochemical inspiration, but up to now their usefulness from the computational complexity point of view was not investigated. An explanation can be the fact that the parallelism inherent to a P system (all evolution rules which can be used, in all membranes, are used at the same time) seems not to be sufficient in order to obtain a significant speed-up of computations (in comparison, for instance, with a Turing machine).

The variant we have introduced here has an enhanced parallelism: the membranes themselves can be multiplied by division. In this way, the number of membranes (and membrane sub-structures, hence "processors" of our computing device) can increase exponentially. We have shown that this feature leads to easy solutions to intractable problems: SAT can be solved in linear (parallel) time. Moreover, this class of P systems is still computationally universal, any recursively enumerable set of vectors of natural numbers (in particular, each recursively enumerable set of natural numbers) can be generated by such a system.

Several natural problems appear in this framework. For instance, it can be of interest to find other problems which are known to be hard (even NP complete) and which can be solved in an easy way by P systems with active membranes. Then, a question of a practical importance is to try to implement a P system of this type, either in biochemical media or in electronic media. To this aim, it could be necessary to consider variants of P systems with membrane division possibilities which are more "realistic", whatever this would mean. This is, of course, a topic for an interdisciplinary research.

References

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

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

[3] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Informatica*, 31 (1994), 719–728.

[4] R. J. Lipton, Using DNA to solve NP-complete problems, *Science*, 268 (April 1995), 542–545

[5] Gh. Păun, Computing with membranes, submitted, 1998 (see also *TUCS Research Report* No. 208, November 1998, http://www.tucs.fi).

[6] Gh. Păun, Computing with membranes. A variant, submitted, 1999 (see also *CDMTCS Report* No. 098, 1999, of CS Department, Auckland Univ., New Zealand, www.cs.auckland.ac.nz/CDMTCS).

[7] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Heidelberg, 1998.

[8] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, submitted, 1999 (see also *TUCS Research Report* No. 218, December 1998, http://www.tucs.fi).

[9] Gh. Păun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, submitted, 1999.

[10] Gh. Păun, T. Yokomori, Membrane computing based on splicing, submitted, 1999.

[11] Gh. Păun, T. Yokomori, Simulating H systems by P systems, submitted, 1999.

[12] Gh. Păun, S. Yu, On synchronization in P systems, submitted, 1999 (see also *CS Department TR* No 539, Univ. of Western Ontario, London, Ontario, 1999, www.csd.uwo.ca/faculty/syu/TR539.html).

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