On the Power of Membrane Division in P Systems[1]

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

Yasuhiro SUZUKI, Hiroshi TANAKA
Medical Research Institute (Bio-Informatics)
Tokyo Medical and Dental University
1-5-45 Yushima Tokyo, 113-0034 Japan
E-mail: {suzuki.com@mri.tmd.ac.jp, tanaka@tmd.ac.jp}

Takashi YOKOMORI
Department of Mathematics, School of Education
Waseda University, 1-6-1 Nishi-waseda, Shinjuku-ku
Tokyo 169-8050, Japan
E-mail: yokomori@mn.waseda.ac.jp

Abstract. First, we consider P systems with active membranes, hence with the possibility that the membranes can be divided, with non-cooperating evolution rules (the objects always evolve separately). These systems are known to be able to solve NP-complete problems in linear time. Here we give a normal form theorem for such systems: their computational universality is preserved even if only the elementary membranes are divided. The possibility of solving SAT in linear time is preserved only when non-elementary membranes may also be divided under the influence of objects in their region.

Second, we consider a slight generalization, namely, we allow that a membrane can produce by division both a copy of itself and a copy of a membrane with a different label; again, only elementary membranes may be divided. In this case, we prove that the hierarchy on the maximal number of membranes present in the system collapses: three membrane at a time are sufficient in order to characterize the recursively enumerable sets of vectors of natural numbers. This result is optimal, two membranes are shown not to be sufficient.

Third, we consider P systems with cooperating rules (several objects may evolve together). Making use of this powerful feature, we show that many NP-complete problems can be solved in linear time in a quite uniform way (by systems which are very similar to each other), using only elementary membranes division (and not further ingredients, such as electrical charges). The degree of cooperation is minimal: two objects at a time.

[1]Research supported by "Research for Future" Program no. JSPS-RFTF 96I00101, from the Japan Society for the Promotion of Science, and by Grant-in-Aid for Exploratory Research, No 11878055 and No 11837005, from the Ministry of Education, Science, Sports, and Culture of Japan.

1 Introduction

P systems are a class of distributed parallel computing models introduced in [8], inspired from the way the alive cells process chemical compounds, energy, and information. In short, in the *regions* delimited by a *membrane structure* (see Figure 1 for an illustration of this notion), one places *multisets* of *objects*, which evolve according to *evolution rules* associated with the regions; a *computation* consists of transitions among system *configurations*; the *result* of a halting computation is the vector of the multiplicities of objects present in the final configuration in a specified *output membrane* or of objects which leave the external membrane of the system (the *skin* membrane) during a computation. That is, a P system of this form *computes* a set of vectors of natural numbers. Many variants characterize the family of recursively enumerable sets of vectors of natural numbers, which are exactly the Parikh sets associated with recursively enumerable languages. Details can be found in [8], [1], [11], [13], [17], etc. When membrane division is allowed, NP-complete problems can be solved in linear time, [12], [7].

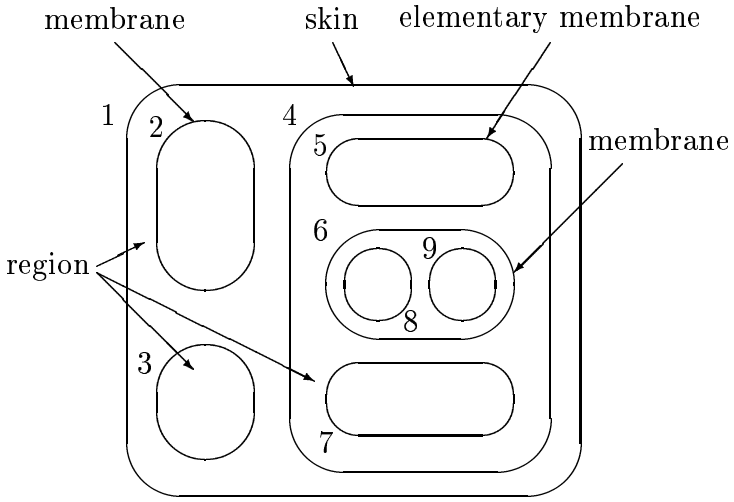

Figure 1: A membrane structure

The division of membranes is entailed in [12] by two means: as a result of the action of an object, and because of the existence of two lower membranes of opposite polarizations. In the first case, rules of the form $[_i a]_i^{\alpha_1} \rightarrow [_i b]_i^{\alpha_2} [_i c]_i^{\alpha_3}$ are used, where i is the label of the membrane, a, b, c are objects, and $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$ are *electrical charges* associated with the membranes. Note that membrane i is an elementary one, both before and after the division. In the second case, one uses rules of the form $[_i [_j \,]_j^{\alpha_1} [_k \,]_k^{\alpha_2}]_i^{\alpha_3} \rightarrow [_i [_j \,]_j^{\alpha_4}]_i^{\alpha_5} [_i [_k \,]_k^{\alpha_6}]_i^{\alpha_7}$, where i, j, k are labels of membranes and $\alpha_1, \ldots, \alpha_7$ are electrical charges such that $\{\alpha_1, \alpha_2\} = \{+, -\}$. Note that always a membrane is divided into two new membranes and that both these new membranes have the same label as the divided membrane.

While rules of the first type can be considered to have a connection with the biochemistry of alive cells, where the division takes place under the influence of the chemical compounds present in the cell, the second type of rules is far from biology, so it is a

natural question whether or not one can get rid of using such rules.

Pleasantly enough, we find that rules of the second type are not necessary in order to obtain the computational universality of the systems. The capability of solving SAT in linear time is preserved only if we allow that also non-elementary membranes can be divided under the influence of objects placed in their regions (but not under the influence of inner membranes of opposite polarizations). When a non-elementary membrane is divided, all membranes present in it are duplicated and added to each of the two new membranes obtained by division. Of course, this is a rather powerful operation (and non-realistic from a biochemical point of view).

In the case of universality, one does not obtain a bound on the number of membranes simultaneously present in the system during the computation. Such a bound – even a rather low one: three – can be found if a slight generalization is introduced in the division process: when dividing a membrane i, one of the resulting membranes should be labeled by i, but the second one may have a different label. This corresponds to the case investigated in [18], [19], where, however, the division is provoked by the "concentration" of certain objects in the membranes. From the proof of our result, one actually can see that also in our case one can consider that the reason for division is the concentration of certain objects in the divided membranes.

In all these cases one uses only non-cooperative rules, that is, rules where the objects evolve independently to each other.

In an attempt to get rid also of the duplication of the membranes present in a dividing membrane, in the last section of the paper we consider systems with cooperating rules, where two or more objects evolve by a common rule. This is also a very powerful feature, leading in an easy way to computational universality (see [8], [3]). We do not examine its power from the *competence* point of view, but from the *performance* point of view and we find rather encouraging results from a "practical" point of view: many NP-complete problems (we consider here ten problems, five from logic and five from graph theory) can be solved in linear time by P systems with cooperative rules, with elementary membrane division (without using electrical charges for membranes) and, more important, in a very uniform way. All systems have the same sub-systems: generating in n steps a set of 2^n elements (candidate solutions to our problem), checking whether or not at least a solution exists, communicating outside the system the result of this checking. On the one hand, this strategy has the same general shape as the so-called *computing by carving*, [10], on the other hand, the uniformity of the way of solving the ten problems reminds the idea of a *problem class* for a given decidability tool, as investigated already in [21] (where several NP-complete problems were proved to be solvable in linear time by CCC programs, with CCC meaning "Computing by Conformational Change"). Finding such classes of problems can be important from a practical point of view: if the solution of one of the problems in the class would be implemented, then the changes necessary to solve any other problem in the class will be minimal.

2 P Systems with Active Membranes

We refer to [16] for the elements of formal language theory we use here. We only specify that for a string $x \in V^*$ and a symbol $a \in V$, we denote by $|x|$ the length of x and by $|x|_a$ the number of occurrences of the symbol a in the string x. The families of context-free, context-sensitive, and recursively enumerable languages are denoted by CF, CS, RE, respectively. For $w \in V^*, V = \{a_1, \ldots, a_n\}$, we denote by $\Psi_V(w)$ the Parikh vector of w, that is, $\Psi_V(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$; this is extended to languages in the natural way. For a family FL of languages, we denote by $PsFL$ the family of Parikh sets of vectors associated with languages in FL.

A membrane structure will be represented by a string of matching labeled parentheses. For instance, the membrane structure in Figure 1 is represented by

$$[_1\, [_2\,]_2\, [_3\,]_3\, [_4\, [_5\,]_5\, [_6\, [_8\,]_8\, [_9\,]_9\,]_6\, [_7\,]_7\,]_4\,]_1.$$

If a membrane with label i has an *electrical charge* $\alpha \in \{+, -, 0\}$, then we indicate this by writing $[_i\,]_i^\alpha$.

A multiset over an alphabet V is represented by a string over V (and by all its permutations) and each string precisely identifies a multiset; the Parikh vector associated with the string indicates the multiplicities of each element of V in the corresponding multiset. Thus, when speaking of a "multiset" $w \in V^*$ we understand the multiset identified by w.

We now recall from [12] the definition of *P systems with active membranes*. Such a system is a construct

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

where:

(i) $m \geq 1$;

(ii) V is an alphabet;

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

(iv) 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;

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

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

(a) $[_h a \to v]_h^\alpha$,
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 membranes, but not directly involving 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) $a[_h\,]_h^{\alpha_1} \to [_h b]_h^{\alpha_2}$,
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) $[_h a\,]_h^{\alpha_1} \to [_h\,]_h^{\alpha_2} b$,
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) $[_h a\,]_h^{\alpha} \to b$,
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; all objects different from a are duplicated in the two new membranes);

(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;
(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; all membranes of neutral polarization are duplicated and then are part of the contents of both copies of the membrane h_0).

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). Also, it is important to note that rules of type (e) refer to elementary membranes.

A system as above is said to be *non-cooperative*. If we also allow rules of type (a) of the general form $[_h u \to v]_h^{\alpha}$, for $h \in H, \alpha \in \{+, -, 0\}, u \in V^+, v \in V^*$, then we say that we have a *cooperative* system. Note that such rules, with more than one object in their "left hand side" are only allowed for the type (a), not for the other types (although

the cooperation can be easily extended to all types of rules, here we do not need such a powerful feature). In all sections below, excepting Section 6, we use only non-cooperative rules.

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

1. All the rules are applied in parallel: in one 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 chosen (there is no priority relation among rules), but any object which can evolve by a rule of any form, must do it.

2. If a membrane is dissolved, then all the objects in its region are left free in the surrounding region. 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 associated 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.

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 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 result of a complete computation consists of the vector describing the multiplicity of all objects which are sent out of the system during the computation; a non-halting computation provides no output. The set of all vectors of natural numbers produced in this way (we say *computed*) by a system Π is denoted by $N(\Pi)$. The family of all such sets $N(\Pi)$ of vectors of natural numbers is denoted by NPA.

In [12] one proves that $PsRE = NPA$ and that SAT (the satisfiability of propositional formulas in the conjunctive normal form) can be solved in linear time by systems as above (the time, consisting of parallel steps in our system, for solving a formula with n variables and m clauses is $2n + 2m + 1$). In the proofs of both these results from [12] rules of type (f) are used, but, as we will see immediately, this is not necessary.

3 A Restricted Variant and Its Power

Let us denote by NPA_{rd} the family of vectors of natural numbers $N(\Pi)$ computed by (non-cooperative) systems Π which do not use division rules of type (f) (the subscript *rd* stands for "restricted division"). Somehow surprisingly, but entirely pleasantly, this restriction does not decreases the power of P systems with active membranes.

In the universality proofs which follow we need the notion of a *matrix grammar with appearance checking*. Such a grammar 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 (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.)

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}. If $F = \emptyset$, then G is said to be without appearance checking. The family of languages generated by such grammars is denoted by MAT.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$. Further details about matrix grammars can be found in [2] and in [16].

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 $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
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, in the last step of a derivation.

According to Lemma 1.3.7 in [2], for each matrix grammar G there is an equivalent matrix grammar G' in the binary normal form; if G is without appearance checking, then also G' is without appearance checking.

Theorem 1. $PsRE = NPA_{rd}$.

Proof. The inclusion $NPA_{rd} \subseteq PsRE$ follows from Church-Turing thesis or can be proved directly, in a straightforward way (but involving a long construction). Hence, we only prove the inclusion $PsRE \subseteq NPA_{rd}$. 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 the obtained grammar by G'. 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 $p = n_1 + n_2 + 1$ initial membranes)

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

with

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

and the following set R of rules:

1. For each matrix $m_i = (X \to Y, A \to x)$, $1 \leq i \leq n_1$, we introduce the rules:

$$
\begin{array}{l}
X[_i\ ]_i^0 \to [_i Y]_i^+, \\
[_i Y \to Y]_i^+, \\
A[_i\ ]_i^+ \to [_i A]_i^0, \\
[_i A]_i^0 \to [_i\ ]_i^0 \langle x \rangle, \\
[_i Y]_i^0 \to [_i\ ]_i^0 Y', \\
[_p \langle x \rangle \to x]_p^0, \\
[_p Y' \to Y]_p^0.
\end{array}
$$

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:

$$
\begin{array}{l}
X[_i\ ]_i^0 \to [_i Y]_i^0, \\
[_i Y]_i^0 \to [_i Y']_i^+ [_i Y']_i^0, \\
A[_i\ ]_i^+ \to [_i \dagger\ ]_i^+, \\
[_i Y']_i^0 \to [_i\ ]_i^0 Y, \\
[_i Y' \to Y]_i^+, \\
[_i Y]_i^+ \to g.
\end{array}
$$

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

$$[_p a]_p^0 \to [_p\ ]_p^0 a,$$

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

$$[_p \alpha \to \alpha]_p^0.$$

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 $w \in (N_2 \cup T)^*$ (also objects $g, \dagger$ can appear, but g never evolves while $\dagger$ evolves forever, so we ignore them: g does not matter, $\dagger$ leads to a non-halting computation which is of no further interest). Initially, we have $w = A$, for $(S \to XA)$ being the starting matrix of G.

As long as symbols α from $N_1 \cup N_2$ are present, the computation is not finished, the rule $[_p \alpha \to \alpha]_p^0$ can be used. That is, we have to remove all nonterminal symbols from membrane p and this is done by simulating a terminal 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 $m_i : (X \to Y, A \to x)$ of 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 that both Y and A can send out of this membrane 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 $m_i : (X \to Y, A \to \#)$. Inside membrane i we have the symbol Y, which entails the division of the membrane. We get two copies of membrane i, one with positive and one with neutral charge; in both membranes we have changed Y to Y'. In the positively charged membrane, Y' is replaced with Y, but, at the same step, if any copy of A exists outside membrane $[_i\ ]_i^+$, then the rule $A[_i\ ]_i^+ \to [_i\ \dagger\ ]_i^+$ should be used (the use of the rule $[_iY' \to Y]_i^+$ does not count as a use of the membrane – see again the principles which govern the use of rules – hence if the rule $A[_i\ ]_i^+ \to [_i\ \dagger\ ]_i^+$ *can* be applied, then it *must* be applied). If this will happen, then the computation will never stop: at the next step, the membrane is dissolved by the rule $[_iY]_i^+ \to g$ and the trap-object $\dagger$ will be left free in membrane p where it will evolve forever. During these steps, the object Y' will leave the membrane $[_i\ ]_i^0$. Consequently, we correctly simulate the use of the matrix m_i in the appearance checking manner; we return to a neutral membrane i and to a computation which can be halted if and only if the symbol A is not present.

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.

At any step, each terminal of G can be sent out of the system by rules $[_pa]_p^0 \to [_p\ ]_p^0 a$. In conclusion, $N(\Pi) = \Psi_T(L(G))$, which ends the proof. □

It is worth noting that in the previous construction we have only membranes with two polarizations, + and 0. However, we have a number of membranes simultaneously present in the system which depends on the number of matrices in G. Bounding this number is a problem of interest, which we will deal with in Section 5.

4 Solving SAT in Linear Time

The main reason of considering membrane division is to provide an enlarged parallelism, in the aim of solving complex problems in a feasible time. This was shown in [12] to be possible for systems which can use (non-cooperative) rules of all the six forms, but, as we have already announced, the restriction not to use rules of type (f) does not lose this possibility, providing that we compensate this by allowing to rules of type (e) to be used also for non-elementary membranes. That is, rules of the form

(e′) $[_ia]_i^{\alpha_1} \to [_ib]_i^{\alpha_2}[_ic]_i^{\alpha_3}$

are allowed, even for i being non-elementary; the contents of membrane $[_i\ ]_i^{\alpha_1}$, objects and membranes included, is copied to both $[_i\ ]_i^{\alpha_2}$ and $[_i\ ]_i^{\alpha_3}$ – with the exception of object a, which is replaced by b, c in the two new membranes, respectively.

In order to show that SAT can still be solved in linear time in this framework, some changes in the proof from [12] are sufficient.

Theorem 2. *The SAT problem can be solved by a (non-cooperative) P system with active membranes, without using dividing rules of the type (f), but using rules of type (e′), in a time which is linear in the number of variables and the number of clauses.*

Proof. Let us consider n variables $x_1, \ldots, x_n, n \geq 1$, and 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 $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m, 1 \leq j \leq p_i$.

We construct the P system

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

with the components

$$\begin{aligned}
V &= \{a_i, t_i, t_i', f_i, f_i' \mid 1 \leq i \leq n\} \cup \{c_i \mid 1 \leq i \leq n+1\} \cup \{c, g, t\},\\
H &= \{0, 1, \ldots, m+1\},\\
\mu &= [_{m+1}[_0[_1[_2 \ldots [_m\,]_m^0 \ldots]_2^{0}]_1^{+}]_0^{0}]_{m+1}^{0},\\
w_0 &= c_1 a_1 a_2 \ldots a_n,\\
w_i &= \lambda, \text{ for all } i = 1, 2, \ldots, m-1 \text{ and for } i = m+1,\\
w_m &= t,
\end{aligned}$$

(note that the only membrane with a positive electrical charge is that with label 1) while the set R contains the following rules:

1. $[_0 c_i \to c_{i+1}]_0^0$, for all $1 \leq i \leq n+1$
   (we count from 1 to n, which is the time needed for producing all 2^n truth-assignments for the n variables);

2. $[_0 a_i]_0^0 \to [_0 t_i]_0^0 [_0 f_i]_0^0$, for all $1 \leq i \leq n$
   (in membrane 0 – which is not an elementary one! – we nondeterministically 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; this operation is done in parallel with increasing the subscript of object c_j, which is present in both the new membranes; it is also worth noting that all other objects and membranes from membrane 0 are duplicated, copies of each of them will be placed in both copies of membrane 0 produced by the present rule; in particular, this is true for all membranes with labels $1, 2, \ldots, m$ associated with the m clauses);

3. $c_{n+1}[_1\,]_1^+ \to [_1 c]_1^0$
   (after n steps, all truth-assignments are generated; they are placed in 2^n copies of membrane 0, together with copies of the membrane sub-structure $[_1[_2 \ldots [_m t]_m^0 \ldots]_2^0]_1^+$; the present rule will change the polarity of membrane 1, which makes possible the checking of the truth values of all clauses, for all truth-assignments, in parallel, by means of the rules in the next group);

4. $t_i[_j\,]^0_j \to [_j t'_i]^+_j$, if x_i appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$, and
$f_i[_j\,]^0_j \to [_j f'_i]^+_j$, if $\neg x_i$ appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$
(the polarization of a membrane with label j, $1 \leq j \leq m$, is changed into + if and only if clause C_j is satisfied by the current truth-assignment);

5. $[_j t'_i]^+_j \to t_i$, and
$[_j f'_i]_j \to f_i$, for all $1 \leq j \leq m, 1 \leq i \leq n$
(the objects t'_i, f'_i introduced in a membrane j by the previous rules if clause j was satisfied will entail the dissolvation of this membrane; note that the truth-assignment is not changed, t_i and f_i are still present in membrane 0, and that after changing the polarization of membrane j to + there is no other applicable rule but the dissolving one; in this way, we can proceed with checking the truth of the subsequent clause);

6. $[_0 t]^0_0 \to [_0\,]^0_0 t$,
$[_{m+1} t]^0_{m+1} \to [_{m+1}\,]^+_{m+1} t$
(if at least one truth-assignment has satisfied all clauses, then all membranes $1, 2, \ldots, m$ from one of the copies of membrane 0 were dissolved and the object t can leave both membrane 0 and, at the next step, the skin membrane; in this way, the skin membrane gets a positive polarization, hence no further object can leave the system).

From the previous explanations it is easy to see that formula γ is satisfiable if and only if the object t leave the system, and this precisely happens at the step number $n+2m+3$: in n steps we generate all the 2^n truth-assignments, in one further step we change the polarization of membrane 1, in $2m$ steps we check the truth value of the m clauses, in two final steps we send out a copy of t providing that this is possible. Thus, in order to check the satisfiability of formula γ we have to watch the system at step $n + 2m + 3$. □

The reader might try to exemplify the previous construction for a simple formula. Of course, the speed-up is obtained by making use of an exponential space, which is obtained in a natural way by the features embedded in a P system as above. This has nothing to do with any possible implementation of such a system, especially in a biochemical framework. For instance, dividing membrane 0 and at that step replicating all the membranes which exist inside it and placing copies of all these membranes in the new copies of membrane 0 is a totally unfeasible operation. It is an important *open problem* whether or not SAT (or another NP-complete problem) can be solved in a linear time (a polynomial time would be good enough) by P systems using rules of type (e) and not (e′) as in the previous proof. In some sense, a positive answer is given in Section 6, but at the price of using cooperative rules.

5 A Slight Generalization

The fact that when dividing a membrane we always produce two new membranes with the same label as the former membrane could be biologically motivated, but it is rather re-

strictive from a mathematical point of view. In this section we relax a little this restriction and instead of rules of type (e) we allow rules of the form

(e'') $[_h a\,]_h^{\alpha_1} \to [_h b\,]_h^{\alpha_2}[_j c\,]_j^{\alpha_3}$,
for $h, j \in H$, h is an elementary membrane, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, and $a, b, c \in V$.

That is, only one of the new membranes must have the same label as the former membrane, the other one may have any other label from the list provided in the system. Note that we return to dividing only elementary membranes.

We denote by $NPA''_{rd}(k)$ the family of sets $N(\Pi)$ of vectors of natural numbers which can be computed by (non-cooperative) P systems with rules of types (a) – (d), (e''), by computations such that at most k membranes are present in any current configuration, $k \geq 1$. It might be surprising to note that the, at the first sight, very innocent generalization we have introduced in the membrane division operation is sufficient in order to get a characterization of $PsRE$ by systems which never have more than three membranes present in a configuration and, moreover, do not use membrane polarization.

Theorem 3. *$PsRE = NPA''_{rd}(k)$, for all $k \geq 3$.*

Proof. We only have to prove the inclusion $PsRE \subseteq NPA''_{rd}(3)$. To this aim, we proceed as in the proof of Theorem 1.

Let $G = (N, T, S, M, F)$ be a matrix grammar with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$. 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 the obtained grammar by G'. 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 $(X \to Y, A \to \#)$, $X, Y \in N_1, A \in N_2$. Let $p = n_1 + n_2 + 1$.

We construct the P system (with only two initial membranes)

$$\Pi = (V, H, [_p[_0\,]_0]_p, w_0, w_p, R),$$

with

$$\begin{aligned}
V &= N_1 \cup N_2 \cup T \cup \{g, Z, \dagger\} \cup \{X' \mid X \in N_1\} \\
&\quad \cup \{\langle x \rangle \mid x \in (N_2 \cup T)^*, (X \to Y, A \to x) \text{ is a matrix in } G'\}, \\
H &= \{0, 1, 2, \ldots, p\}, \\
w_0 &= \lambda, \\
w_p &= XA, \text{ for } (S \to XA) \text{ being the initial matrix of } G,
\end{aligned}$$

and the following set R of rules:

1. For each matrix $m_i = (X \to Y, A \to x)$, $1 \leq i \leq n_1$, we introduce the rules:

$$\begin{aligned}
&X[_0\,]_0 \to [_i X]_i, \\
&[_0 X]_0 \to [_0 g]_0[_i Y']_i, \\
&[_i Y' \to Y']_i, \\
&A[_i\,]_i \to [_i A]_i,
\end{aligned}$$

$[_i A]_i \to \langle x \rangle$,
$[_p \langle x \rangle \to x]_p$,
$[_p Y' \to Y]_p$.

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:

   $X[_0 \,]_0 \to [_0 X]_0$,
   $[_0 X]_0 0 \to [_0 Y']_0 [_i Y']_i$,
   $A[_i \,]_i \to [_i \dagger \,]_i$,
   $[_i Y' \to Y]_i$,
   $[_i Y]_i \to g$,
   $[_0 Y']_0 \to [_0 \,]_0 Y$.

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

   $[_p a]_1 \to [_1 \,]_p a$,

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

   $[_p \alpha \to \alpha]_p$.

This system works in a way which is rather similar to the work of the system in the proof of Theorem 1. Instead of having copies of membranes i associated with the matrices in G provided from the beginning, we create such membranes by dividing membrane 0. A copy of membrane 0 is reproduced, it takes its part during simulating a matrix of G, then it returns to its initial state, and waits for being used in a subsequent simulation. The copy of membrane i is dissolved immediately after simulating the corresponding matrix m_i.

We leave to the reader the routine task to check that the system works as desired, that is, $N(\Pi) = \Psi_T(L(G))$. Because at any step of a computation we have at most three membranes present in the current configuration, we have $\Psi_T(L(G)) \in NPA''_{rd}(3)$, which concludes the proof. □

The previous result is optimal from the point of view of the number of membranes simultaneously present in a configuration. Indeed, if we do not allow having more than two membranes in the system, then the dividing operation is not at all used: the skin membrane cannot be divided, hence we have at any time at most one further membrane inside. Thus, instead of $NPA''_{rd}(k), k = 1, 2$, we write $NPA(k)$.

In order to clarify the relations between these two families and $NPA''_{rd}(3)$ we will use the Parikh images of ET0L languages.

An *ET0L system* is a construct $G = (V, T, w, P_1, \ldots, P_m)$, $m \geq 1$, where V is an alphabet, $T \subseteq V$, $w \in V^*$, and $P_i, 1 \leq i \leq m$, are finite sets (*tables*) of context-free rules over V such that for each $a \in V$ there is at least one rule $a \to x$ in each set P_i (we say that each of these tables is *complete*). In a derivation step, all the symbols present in the current sentential form are rewritten using *one* table. The language generated by G, denoted by $L(G)$, consists of all strings over T which can be generated in this way,

starting from w. An ET0L system with only one table is called an *E0L system*. Details can be found, e.g., in [15].

We denote by $E0L$ and $ET0L$ the families of languages generated by E0L and ET0L systems, respectively.

The following inclusions are known:

$$PsCF \subset PsE0L \subset PsET0L \subset PsCS.$$

For instance, $\{(2^n) \mid n \geq 1\} \in PsE0L - PsCF$, $\{(2^n 3^m) \mid n, m \geq 1\} \in PsET0L - PsE0L$ (according to Exercise II.4.4 in [15], we have $\{a^{2^n 3^m} \mid n, m \geq 1\} \in ET0L - E0L$), and $\{(n) \mid n \text{ prime}\} \in PsCS - PsET0L$ (see Exercise VI.2.6 in [15]).

Theorem 4. (i) $NPA(1) \subseteq NPA(2) \subseteq PsET0L \subset NPA''_{rd}(3) = PsRE$.
(ii) *If* $M \in PsE0L, M \subseteq \mathbf{N}^k$, *then* $M \times \{1\} \in NPA(1)$.
(iii) $NPA(1) - PsE0L \neq \emptyset$.

Proof. (i) The inclusions among NPA families follow from the definitions.

The inclusion $NPA(2) \subseteq PsET0L$ can be proved as in [3] and [6]. Because we do not have membrane division possibilities, we can at most dissolve the inner membrane. Evolution rules (including rules which move objects in and out of membranes, as well as dissolving rules) can be simulated in an ET0L system, by "colouring" the symbols with the label of the membrane where they are placed. For details, the reader is referred to the two paper cited above, where much more general variants of P systems are shown to lead to vector sets in $PsET0L$.

The strictness of the inclusion $PsET0L \subset PsRE$ follows from the fact mentioned above, that $PsET0L \subset PsCS$.

(ii) Let us consider an E0L system $G = (V, T, w, P)$ and construct the P system

$$\begin{aligned}
\Pi &= (V \cup \{c\}, \{1\}, [\,_1\,]^0_1, cw, R), \\
R &= \{[\,_1 a \to x]^0_1 \mid a \to x \in P\} \\
&\quad \cup \{[\,_1 c \to c]^0_1,\ [\,_1 c]^0_1 \to [\,_1\,]^+_1 c\} \\
&\quad \cup \{[\,_1 a]^+_1 \to [\ ]^+_1 a \mid a \in T\} \\
&\quad \cup \{[\,_1 \alpha \to \alpha]^+_1 \mid \alpha \in V - T\}.
\end{aligned}$$

As long as membrane 1 has the charge 0, we can simulate the rules of G (note that in E0L systems and in P systems alike the rules are used in parallel). In any moment, the new symbol c can leave the system and change the polarization of the skin membrane to $+$. Now, all terminal symbols can leave the system, while all non-terminal symbols will evolve forever. That is, the computation stops only if the change of polarization of the skin membrane was done after obtaining a terminal string in the E0L system G. Consequently, $N(\Pi) = \Psi_T(L(G)) \times \{1\}$ (we assume that c is placed after the symbols of T when counting the multiplicities of symbols sent out of the system).

(iii) Let us consider the P system

$$\Pi = (\{a, a', b, c\}, \{1\}, [\,_1\,]^0_1, abc, R),$$

$$R = \{[_1 a \to aa]_1^0,\ [_1 b \to b]_1^0,\ [_1 b]_1^0 \to [_1\ ]_1^+ b,$$
$$[_1 a \to a'a'a']_1^+,\ [_1 a' \to a'a'a']_1^+,\ [_1 c \to c]_1^+,\ [_1 c]_1^+ \to [_1\ ]_1^0 c,$$
$$[_1 a']_1^0 \to [_1\ ]_1^0 a\}.$$

As long as membrane 1 has polarization 0, the object a is doubled and b is just reproduced; at any moment, b can leave the system and change the charge of the skin membrane to $+$. In such a circumstance, a is replaced by a' and tripled at every step, while c is just reproduced. At any moment, also c can leave the system and change again the polarization of the skin membrane, returning it to 0. From now on, only rules which send all copies of a' outside the system, changed into a, can be used. Consequently, $N(\Pi) = \{(2^n 3^m, 1, 1) \mid n, m \geq 1\}$. The family $E0L$ is closed under erasing morphisms; if $N(\Pi)$ would be in $PsE0L$, then also $\{(2^n 3^m) \mid n, m \geq 1\}$ would be in $PsE0L$, which is not true. Consequently, $N(\Pi) \notin PsE0L$. □

From the previous result we find that $NPA(2) \subset NPA''_{rd}(3)$ is a strict inclusion. We do not know whether or not also the inclusion $NPA(1) \subseteq NPA(2)$ is proper.

Note that in the the previous proof we have again used two polarizations of membranes, $+$ and 0.

The fact that in the previous proof we have not obtained the inclusion $PsE0L \subseteq NPA(1)$ is due to the restriction that we can change the polarization of a membrane only when passing objects through it. Moreover, when sending an object out of a membrane, we can at most change the name of the object, but we cannot "destroy" the object. If rules of the form $[_1 c]_1^0 \to [_1\ ]_1^+$ would be allowed, then the additional component of the vectors in $\Psi_T(L(G))$ will not be necessary.

6 Using Cooperative Rules

Working with non-cooperative rules is natural from a mathematical point of view, because it is natural to look for *minimalistic* models, but from a biochemical point of view this is not only non-necessary, but also non-realistic: most chemical reactions involve two or more chemical compounds (and also produce two or more compounds). "Reactions" of the form $a \to bc$ corresponds to breaking a molecule a into two smaller molecules, b and c, but many reactions are of the form $ab \to cd$, where a and b interact in producing c and d.

The cooperative rules are very powerful in what it concerns the computing capacity of P systems, in the sense that by using such rules it is very easy to obtain computational universality ([8], [3], [1]). We are not interested here in the computing capacity, but in the computing efficiency. Not entirely surpringly, we find that using such rules we can solve many NP-complete problems in a rather uniform manner, that is, by P systems which are very similar to each other.

The general structure (and functioning) of these systems is as follows:

1. Always we start with two membranes, always the central one is divided into an exponential number of copies.

2. In a central "parallel engine" one generates, making use of the membrane division, a "data pool" of an exponential size; due to the parallelism, this takes, however, only a linear time. In parallel with this process, a "timer" is simultaneously tick-ing, in general, for synchronization reasons.

3. After finishing the generation of the "data pool", one checks whether or not any solution to our problem exists; this is the step where we need cooperative rules.

4. A message is sent out of the system at a precise moment telling whether or not the problem has a solution. In all cases, the last two steps are done in a constant number of time units, again making use of the parallelism.

Figure 2 pictorially suggests the general shape of the systems which follow.

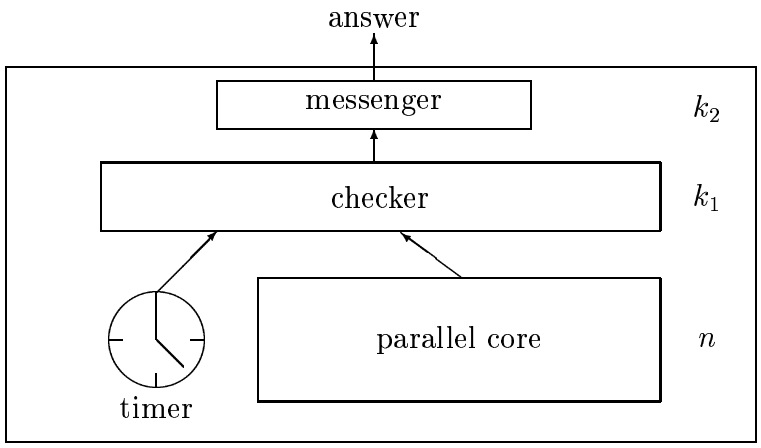

Figure 2: The shape of P systems solving NP-complete problems

Many other decidability problems can be approached in the same way, but we just illustrate here the idea with five problems from logic and five from graph theory. It would be of interest to note that in the case of logic we do not use membrane dissolving rules; actually, we only use rules of three forms, (a) (cooperative), (c), and (e). This makes particularly interesting such problems in the case of any attempt to implement P systems for solving them.

Because in all cases we have the initial membrane structure $[_1[_0\ ]_0]_1$, and an empty multiset in the skin membrane, we do no longer specify these items, but we only give the other components of the systems, the set V of objects, the initial multiset w_0, and the set of evolution rules. For the formulation of the problems, we follow [4].

PROPOSITIONAL LOGIC

[LO1] **Satisfiability (SAT)**

INSTANCE: A collection C of clauses $C_i = y_{i,1} \vee \ldots \vee y_{i,p_i}$, $1 \leq i \leq m$, for some $m \geq 1, p_i \geq 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}, n \geq 1$, for each $1 \leq i \leq m, 1 \leq j \leq p_i$.

QUESTION: Is there a satisfying truth-assignment for C?

P SYSTEM:

$$
\begin{aligned}
V &= \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \\
&\cup \{c_i \mid 1 \leq i \leq n+1\} \\
&\cup \{k_i \mid 1 \leq i \leq m\} \cup \{yes\}, \\
w_0 &= c_1 a_1 \ldots a_n, \\
R &= \{[_0 a_i]_0 \to [_0 t_i]_0 [_0 f_i]_0 \mid 1 \leq i \leq n\} \ (\textit{parallel engine}), \\
&\cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \leq i \leq n\} \ (\textit{timer}) \\
&\cup \{[_0 c_{n+1} t_i \to k_1 t_i]_0 \mid x_i \text{ appears in } C_1, 1 \leq i \leq n\} \\
&\cup \{[_0 c_{n+1} f_i \to k_1 f_i]_0 \mid \neg x_i \text{ appears in } C_1, 1 \leq i \leq n\} \\
&\cup \{[_0 k_j t_i \to k_{j+1} t_i]_0 \mid x_i \text{ appears in } C_{j+1}, 1 \leq i \leq n, 1 \leq j \leq m-1\} \\
&\cup \{[_0 k_j f_i \to k_{j+1} f_i]_0 \mid \neg x_i \text{ appears in } C_{j+1}, 1 \leq i \leq n, 1 \leq j \leq m-1\} \\
&\quad (\textit{truth-checking rules}) \\
&\cup \{[_0 k_m]_0 \to [_0\,]_0 yes,\ [_1 yes]_1 \to [_1\,]_1 yes\} \ (\textit{messenger rules}).
\end{aligned}
$$

In n steps, we generate all 2^n truths-assignments of the n variables; at the same time, the counter c_i arrives at c_{n+1} and starts checking the truth of clauses; if a clause C_j is valid, then the object k_j is introduced. Checking all clauses takes m steps. If in any copy of the membrane 0 we can obtain the object k_m, then the "message" *yes* is sent to membrane 1 and from here out of the system. This means that the set C of clauses is satisfiable if and only if at step $n+m+2$ we get the object *yes* outside the system.

Note that for each clause we have as many checking rules as many literals we have in the clause.

[LO2] **3-Satisfiability (3-SAT)**

INSTANCE: As in the case of SAT, with each clause containing exactly three literals (that is, $p_i = 3$ for all $1 \leq i \leq m$).

QUESTION: Is there a satisfying truth assignment for C?

P SYSTEM: Exactly as above, but with only three checking-rules for each clause (hence, the answer is obtained at step $n+5$).

[LO3] **Not-all-equal 3SAT**

INSTANCE: As in the case of 3-SAT.

QUESTION: Is there a truth-assignment such that each clause in C has at least one true literal and at least one false literal?

P SYSTEM: Exactly as above, but with the following truth-checking rules:

$$
\begin{aligned}
&\{[_0 c_{n+1} t_{i_1} t_{i_2} \to k_1 t_{i_1} t_{i_2}]_0 \mid x_{i_1}, \neg x_{i_2} \text{ appear in } C_1, 1 \leq i_1, i_2 \leq n\} \\
\cup\ &\{[_0 c_{n+1} t_{i_1} f_{i_2} \to k_1 t_{i_1} f_{i_2}]_0 \mid x_{i_1}, x_{i_2} \text{ or } \neg x_{i_1}, \neg x_{i_2} \text{ appear in } C_1, 1 \leq i_1, i_2 \leq n\} \\
\cup\ &\{[_0 c_{n+1} f_{i_1} f_{i_2} \to k_1 f_{i_1} f_{i_2}]_0 \mid \neg x_{i_1}, x_{i_2} \text{ appear in } C_1, 1 \leq i_1, i_2 \leq n\} \\
\cup\ &\{[_0 k_j t_{i_1} t_{i_2} \to k_{j+1} t_{i_1} t_{i_2}]_0 \mid x_{i_1}, \neg x_{i_2} \text{ appear in } C_{j+1}, 1 \leq i_1, i_2 \leq n, 1 \leq j \leq m-1\} \\
\cup\ &\{[_0 k_j t_{i_1} f_{i_2} \to k_{j+1} t_{i_1} f_{i_2}]_0 \mid x_{i_1}, x_{i_2} \text{ or } \neg x_{i_1}, \neg x_{i_2} \text{ appear in } C_{j+1},
\end{aligned}
$$

$1 \leq i_1, i_2 \leq n, 1 \leq j \leq m-1\}$

$\cup \{[_0 k_j f_{i_1} f_{i_2} \rightarrow k_{j+1} f_{i_1} f_{i_2}]_0 \mid \neg x_{i_1}, x_{i_2}$ appear in $C_{j+1}, 1 \leq i_1, i_2 \leq n, 1 \leq j \leq m-1\}$.

For each clause we have three rules in each membrane 0.

The previous rules contain three objects in their left side, but it is easy to modify them in such a way to have only two objects in each case. For instance, instead of a rule $[_0 k_j t_{i_1} t_{i_2} \rightarrow k_{j+1} t_{i_1} t_{i_2}]_0$ we can use the rules

$$[_0 k_j t_{i_1} \rightarrow \langle k_j t_{i_1} \rangle]_0,$$
$$[_0 \langle k_j t_{i_1} \rangle t_{i_2} \rightarrow k_{j+1} t_{i_1} t_{i_2}]_0,$$

where $\langle k_j t_{i_1} \rangle$ is a new object. Instead of m steps, we need now $2m$ steps for checking the truth values of the m clauses (hence the answer is obtained at the step $n + 2m + 2$).

In the same way, with a linear slow-down, we can change all cooperative rules from the systems which follow, hence we will not state this observation again for each case.

[LO4] **One-in-three 3SAT**

INSTANCE: As in the case of 3-SAT.

QUESTION: Is there a truth-assignment such that each clause in C has exactly one true literal?

P SYSTEM: Exactly as above, but with the following truth-checking rules:

$\{[_0 c_{n+1} t_{i_1} t_{i_2} t_{i_3} \rightarrow k_1 t_{i_1} t_{i_2} t_{i_3}]_0 \mid x_{i_1}, \neg x_{i_2}, \neg x_{i_3}$ appear in C_1,
$1 \leq i_1, i_2, i_3 \leq n\}$

$\cup \{[_0 c_{n+1} t_{i_1} t_{i_2} f_{i_3} \rightarrow k_1 t_{i_1} t_{i_2} f_{i_3}]_0 \mid x_{i_1}, \neg x_{i_2}, x_{i_3}$ or $\neg x_{i_1}, \neg x_{i_2}, \neg x_{i_3}$ appear in C_1,
$1 \leq i_1, i_2, i_3 \leq n\}$

$\cup \{[_0 c_{n+1} t_{i_1} f_{i_2} f_{i_3} \rightarrow k_1 t_{i_1} f_{i_2} f_{i_3}]_0 \mid x_{i_1}, x_{i_2}, x_{i_3}$ or $\neg x_{i_1}, x_{i_2}, \neg x_{i_3}$ appear in C_1,
$1 \leq i_1, i_2, i_3 \leq n\}$

$\cup \{[_0 c_{n+1} f_{i_1} f_{i_2} f_{i_3} \rightarrow k_1 f_{i_1} f_{i_2} f_{i_3}]_0 \mid \neg x_{i_1}, x_{i_2}, x_{i_3}$ appear in C_1,
$1 \leq i_1, i_2, i_3 \leq n\}$

$\cup \{[_0 k_j t_{i_1} t_{i_2} t_{i_3} \rightarrow k_{j+1} t_{i_1} t_{i_2} t_{i_3}]_0 \mid x_{i_1}, \neg x_{i_2}, \neg x_{i_3}$ appear in C_{j+1},
$1 \leq i_1, i_2, i_3 \leq n\}$

$\cup \{[_0 k_j t_{i_1} t_{i_2} f_{i_3} \rightarrow k_{j+1} t_{i_1} t_{i_2} f_{i_3}]_0 \mid x_{i_1}, \neg x_{i_2}, x_{i_3}$ or $\neg x_{i_1}, \neg x_{i_2}, \neg x_{i_3}$ appear in C_{j+1},
$1 \leq i_1, i_2, i_3 \leq n, 1 \leq j \leq m-1\}$

$\cup \{[_0 k_j t_{i_1} f_{i_2} f_{i_3} \rightarrow k_{j+1} t_{i_1} f_{i_2} f_{i_3}]_0 \mid x_{i_1}, x_{i_2}, x_{i_3}$ or $\neg x_{i_1}, x_{i_2}, \neg x_{i_3}$ appear in C_{j+1},
$1 \leq i_1, i_2, i_3 \leq n, 1 \leq j \leq m-1\}$

$\cup \{[_0 k_j f_{i_1} f_{i_2} f_{i_3} \rightarrow k_{j+1} f_{i_1} f_{i_2} f_{i_3}]_0 \mid \neg x_{i_1}, x_{i_2}, x_{i_3}$ appear in C_{j+1},
$1 \leq i_1, i_2, i_3 \leq n, 1 \leq j \leq m-1\}$.

For each clause we have exactly one rule in each membrane 0 (we will have three rules if we reduce to two the number of objects which cooperate in each rule).

[LO5] **Minimum 2-Satisfiability**

INSTANCE: A set of n variables, a collection C of m clauses such that each clause contains exactly two literals ($p_i = 2$ for all $1 \leq i \leq m$), and a positive integer $k \leq m$.

QUESTION: Is there a truth-assigment that simultaneously satisfies at least k clauses in C?

P SYSTEM:

$$
\begin{aligned}
V &= \{a_i, t_i, t_i', f_i, f_i' \mid 1 \leq i \leq n\} \\
&\quad \cup \{c_i \mid 1 \leq i \leq n+3\} \\
&\quad \cup \{d_i \mid 1 \leq i \leq m\} \cup \{d, yes\}, \\
w_0 &= c_1 a_1 \ldots a_n, \\
R &= \{[_0 a_i]_0 \to [_0 t_i'][_0 f_i']_0 \mid 1 \leq i \leq n\} \\
&\quad \cup \{[_0 t_i' \to t_i^m]_0,\ [_0 f_i' \to f_i^m]_0 \mid 1 \leq i \leq n\} \\
&\quad \cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \leq i \leq n, \text{ and } i = n+2\} \\
&\quad \cup \{[_0 c_{n+1} \to c_{n+2} d_1 d_2 \ldots d_m]_0\} \\
&\quad \cup \{[_0 d_j t_i \to d t_i]_0 \mid x_i \text{ appears in } C_{j+1}, 1 \leq i \leq n, 1 \leq j \leq m\} \\
&\quad \cup \{[_0 d_j f_i \to d f_i]_0 \mid \neg x_i \text{ appears in } C_{j+1}, 1 \leq i \leq n, 1 \leq j \leq m\} \\
&\quad \cup \{[_0 c_{n+3} d^k \to yes]_0,\ [_0 yes]_0 \to [_0\ ]_0 yes,\ [_1 yes]_1 \to [_1\ ]_1 yes\}.
\end{aligned}
$$

The answer is obtained in $n+5$ steps ($n+k+4$ steps if we "reduce" the rule $[_0 c_{n+3} d^k \to yes]_0$ to rules with only two cooperating objects), because the checking of the truth of the m clauses is done in parallel for all of them, in only one step, due to the presence of m copies of each truth value t_i, f_i and m objects $d_1, \ldots, d_m$ in each membrane 0. (This trick, with checking all clauses in parallel, in only one step, can be applied also for SAT, but at the price of using a rule of the form $[_0 c_{n+3} d^m \to yes]_0$, checking that *all* clauses are satisfied; if we replace this rule with m rules with only two objects in their left hand sides, then we return to a system as that considered above for SAT.) The fact that each clause contains only two literals is only "visible" in what concerns the number of checking rules, not in the shape of the system.

In the case of logic, we have chosen the first problems from the beginning of the respective section from [4], but for graph theory we have to select decision problems only; moreover, we present the problems in the order of their difficulty/similarity.

GRAPH THEORY

[GT1] **Vertex Cover**

INSTANCE: A directed graph $G = (V, E)$ with n vertices, $v_1, \ldots, v_n$, and a positive integer $k \leq card(V)$.

QUESTION: Is there a subset $V' \subseteq V$ with $card(V') \leq k$ such that for all $(v_i, v_j) \in E, 1 \leq i, j \leq n$, at least one of v_i, v_j is in V'?

P SYSTEM:

$$V = \{v_i, v_i', v_i'', v_i''' \mid 1 \leq i \leq n+1\}$$

$$\cup \{c_i \mid 1 \leq i \leq n+3\} \cup \{c, d, yes\},$$
$$w_0 = c_1 v_1 \ldots v_n v_{n+1},$$
$$R = \{[_0 v_i]_0 \to [_0 v_i''']_0 [_0 v_i'']_0 \mid 1 \leq i \leq n+1\}$$
$$\cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \leq i \leq n+2\}$$
$$\cup \{[_0 v_i''' \to v_i' c]_0 \mid 1 \leq i \leq n+1\}$$
$$\cup \{[_0 c_{n+3} v_i'' v_j'' \to d]_0 \mid (v_i, v_j) \in E, 1 \leq i, j \leq n\}$$
$$\cup \{[_0 c_{n+3} c^{k+1} \to d]_0,\ [_0 d]_0 \to d,$$
$$d[_0\ ]_0 \to [_0 yes]_0,\ [_0 yes]_0 \to [_0\ ]_0 yes,\ [_1 yes]_1 \to [_1\ ]_1 yes\}.$$

In $n+1$ steps we generate all possible partitions of $V \cup \{v_{n+1}\}$ in two subsets, by triple priming and double priming the objects; the tripled primed objects correspond to the set V' we look for. In step $n+2$ the triple primed objects introduce a single primed version as well as the object c, used for checking whether or not $card(V') \leq k$. This is done at step $n+3$ by the rule $[_0 c_{n+2} c^{k+1} \to d]_0$. Because we have considered an object v_{n+1}, plus its primed versions, which does not correspond to a vertex in V, we can have $k = card(V)$, hence $V' = V$; if also v_{n+1} is put in V', then we get $k+1$ copies of c; if v_{n+1} is not in V', then the rule $[_0 c_{n+3} c^{k+1} \to d]_0$ is not applicable. The object d will dissolve the copy of membrane 0 where it appears. The rules $[_0 c_{n+3} v_i'' v_j'' \to d]_0$ are present for all edges in E (hence not for $i, j = n+1$) and any of can be applied if and only if the negation of the property in the problem is fulfilled. In such a case, the membrane 0 which corresponds to that subset V' is dissolved. Note that at least one membrane is dissolved, for instance, for the case $V' = V \cup \{v_{n+1}\}$ (because in this case we get $card(V') \geq k+1$), hence at least one copy of d will be produced in membrane 1. If any membrane survives, then the rule $d[_0\ ]_0 \to [_0 d]_0$ can be applied. This means that a satisfactory set V' exists and the corresponding message is sent out of the system (at step $n+7$)

The Vertex Cover problem was solved in linear time also in [7], but by using P systems with active membranes with rules of all forms (a) – (f), moreover, allowing that a membrane can produce by division any number of copies of itself, not only two; however, in [7] one uses only non-cooperative rules.

[GT58] **k-Closure**

INSTANCE: The same as above.

QUESTION: Is there a subset $V' \subseteq V$ with $card(V') \leq k$ such that for all $(v_i, v_j) \in E, 1 \leq i, j \leq n$, either $v_i \in V'$ or $v_j \notin V'$?

P SYSTEM: Exactly the same as for Vertex Cover, with the checking rules $[_0 c_{n+2} v_i'' v_j'' \to d]_0$ replaced by:

$$\{[_0 c_{n+3} v_i'' v_j' \to d]_0,\ [_0 c_{n+3} v_i' v_j'' \to d]_0 \mid (v_i, v_j) \in E, 1 \leq i, j \leq n\}.$$

[GT20] **Independent Set**

INSTANCE: The same as above (but we may assume that $k < card(V)$, because the case when $k = card(V)$ corresponds to $E = \emptyset$, which is trivial).

QUESTION: Is there a subset $V' \subseteq V$ with $card(V') \geq k$ such that no two vertices in V' are joined by an edge in E?

P SYSTEM: Very similar to that for Vertex Cover, but we give it in full details just to let the reader observe the similarities.

$$
\begin{aligned}
V &= \{v_i, v_i', v_i'', v_i''' \mid 1 \le i \le n\} \\
&\quad \cup \{c_i \mid 1 \le i \le n+2\} \cup \{c, d, yes\}, \\
w_0 &= c_1 v_1 \dots v_n, \\
R &= \{[_0 v_i]_0 \to [_0 v_i']_0 [_0 v_i''']_0 \mid 1 \le i \le n\} \\
&\quad \cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \le i \le n+1\} \\
&\quad \cup \{[_0 v_i''' \to v_i'' c]_0 \mid 1 \le i \le n\} \\
&\quad \cup \{[_0 c_{n+2} v_i' v_j' \to d]_0 \mid (v_i, v_j) \in E, 1 \le i, j \le n\} \\
&\quad \cup \{[_0 c_{n+2} c^{n-k+1} \to d]_0,\ [_0 d]_0 \to d, \\
&\qquad d[_0\,]_0 \to [_0 yes]_0,\ [_0 yes]_0 \to [_0\,]_0 yes,\ [_1 yes]_1 \to [_1\,]_1 yes\}.
\end{aligned}
$$

Note that this time we need to check that we have *at least* k selected vertices, hence we dissolve a membrane 0 only when less that k vertices are marked with a prime.

[GT4] **Graph 3-Colorability**

INSTANCE: The same as above.

QUESTION: Does there exists a function $f : V \longrightarrow \{1, 2, 3\}$ such that $f(v_i) \neq f(v_j)$ whenever $(v_i, v_j) \in E$?

P SYSTEM:

$$
\begin{aligned}
V &= \{v_i^\alpha \mid 1 \le i \le n, \alpha \in \{(1), (2), (3), (23)\} \\
&\quad \cup \{c_i \mid 1 \le i \le 2n+2\} \cup \{d, yes\}, \\
w_0 &= c_1 v_1 \dots v_n, \\
R &= \{[_0 v_i]_0 \to [_0 v_i^{(1)}]_0 [_0 v_i^{(23)}]_0,\ [_0 v_i^{(23)}]_0 \to [_0 v_i^{(2)}]_0 [_0 v_i^{(3)}]_0 \mid 1 \le i \le n\} \\
&\quad \cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \le i \le 2n+1\} \\
&\quad \cup \{[_0 c_{2n+1} v_i^{(r)} v_j^{(r)} \to d]_0 \mid (v_i, v_j) \in E, 1 \le i, j \le n, 1 \le r \le 3\} \\
&\quad \cup \{[_0 d]_0 \to d,\ d[_0\,]_0 \to [_0 yes]_0,\ [_0 yes]_0 \to [_0\,]_0 yes,\ [_1 yes]_1 \to [_1\,]_1 yes\}.
\end{aligned}
$$

We leave to the reader the task to see how this system works (and, also, to see the differences and the similarities with the previous systems). We only mention that at least for constant functions f we dissolve a membrane 0, hence we always produce an object d.

[GT6] **Monochromatic Triangle**

INSTANCE: A graph $G = (V, E)$ with n vertices and m edges ($E = \{e_1, \dots, e_m\}$).

QUESTION: Is there a partition of E into two disjoint sets E_1, E_2 such that neither $G_1 = (V, E_1)$ nor $G_2 = (V, E_2)$ contains a triangle?

P SYSTEM: Again very similar to the previous ones, but this time we have to find a subset E' of E, such that, simultaneously, $card(E') \ge 1$ and $card(E') \le m - 1$.

$$
V = \{e_i, e_i', e_i'', \bar{e}_i', \bar{e}_i'' \mid 1 \le i \le n\}
$$

$$
\begin{aligned}
&\cup \{c_i \mid 1 \leq i \leq m+2\} \cup \{d, g, h, yes\},\\
w_0 &= c_1 e_1 \dots e_m,\\
R &= \{[_0 e_i]_0 \to [_0 \bar{e}_i']_0 [_0 \bar{e}_i'']_0 \mid 1 \leq i \leq m\}\\
&\cup \{[_0 c_i \to c_{i+1}]_0 \mid 1 \leq i \leq m+1\}\\
&\cup \{[_0 \bar{e}_i' \to e_i' g]_0,\ [_0 \bar{e}_i'' \to e_i'' h \mid 1 \leq i \leq m\}\\
&\cup \{[_0 c_{m+2} g^m \to d]_0,\ [_0 c_{m+2} h^m \to g]_0\}\\
&\cup \{[_0 c_{m+2} e_i' e_j' e_k' \to d]_0,\ [_0 c_{m+2} e_i'' e_j'' e_k'' \to d]_0 \mid 1 \leq i, j, k \leq m,\\
&\quad e_i = (v_{s_1}, v_{s_2}), e_j = (v_{s_2}, v_{s_3}), e_j = (v_{s_3}, v_{s_1}), 1 \leq s_1, s_2, s_3 \leq n\}\\
&\cup \{[_0 d]_0 \to d,\ d[_0\ ]_0 \to [_0 yes]_0,\ [_0 yes]_0 \to [_0\ ]_0 yes,\ [_1 yes]_1 \to [_1\ ]_1 yes\}.
\end{aligned}
$$

The similarity of all these systems encourages the speculation that a few "modules" of "bio-computers" based on the P systems architecture are sufficient in order to solve a large class of decidability problems. We believe that similar "computing blocks" can also be found for other types of problems, for instance, for problems whose answer is not of the YES-NO type, but a number. It remains as a *research topic* to find such classes of problems (and the corresponding types of P systems).

7 Final Remarks

We have improved the universality result from [12], from two points of view: only two polarizations were considered for membranes (+ and 0) and, more important from the biochemical interpretation, we have no longer used rules for membrane division where inner membranes of opposite polarizations force the division of an upper membrane. Then, by allowing that one of the membranes obtained by dividing a membrane with a given label may have a different label, we have found that in order to characterize the family of recursively enumerable sets of vectors of natural numbers we need at most three membranes to be simultaneously present in the configurations of a computation and that no polarization is needed. However, the type of these membranes is not restricted (in the proofs above, this number depends on the number of matrices in the starting grammar). It is an *open problem* whether or not also the number of types of membranes can be bounded.

In all previous results, the evolution rules are non-cooperative, always a single object enters the "reaction". In the last section, we have allowed cooperative rules, a feature which is known to be powerful as it concerns the computing power. We find and convincingly illustrate here that such rules are also very efficient from the time complexity point of view: ten NP-complete problems were shown to be solved in linear time by P systems with elementary membrane division (without electrical charges of membranes), using cooperative rules, and of a very uniform structure. This last remark is a quite encouraging finding; a small number of "computing modules" can be used, sligthly modified from a problem to another one, in order to solve any problem from a large class. We emphasize the fact that cooperative evolution rules are rather common in biochemistry, in some sense, more unrealistic can be considered the non-cooperative rules than the cooperative ones.

A problem of (mathematical) interest in the case of using cooperative rules is to also consider membrane polarization. Which kind of an advantage can we get in this way?

From a biochemical point of view, it is also natural to consider that at a given step a membrane can enter several operations of sending objects in and out. This would correspond to the fact that the alive cells have several proteins embedded in the membrane, and all of these "protein channels" can work simultaneously. Mathematically, this means that a constant g is associated with each membrane and at each step we can use at most g rules which send objects through this membrane. The study of this variant remains to be done. (For instance, an interesting question is whether or not the parameter g induces an infinite hierarchy of the computed vectors.)

Membrane division was considered in [18], [19] in a diferent style: certain objects are distinguished and a membrane is divided as soon as the ratio of the number of copies of these distinguished objects over the total number of objects in the membrane exceeds a given threshold. The contents of the former membrane is distributed in a random way among the two new membranes. The two new membranes can have any label, not necessarily the label of the starting membrane.

The reader can see in the proofs of Theorems 1 and 3 that the membranes are divided only when any object from N_1 is present in them. Moreover, in that moment no other object is present in the divided membrane. Thus, by considering as distinguished objects those from N_1 and as the threshold controlling the division any strictly positive subunitary number, we are exactly in the situation from [18], [19]. Of course, the difference remains in the way of defining the contents of the newly obtained membranes.

Another strategy of controling the work of P systems is proposed in [14]: taking into account the energy consumed or produced by each rule. This can be also used with respect to systems with active membranes. The investigation of such a case remains to be done.

References

[1] C. Calude, Gh. Păun, *Computing with Cells and Atoms*, Taylor and Francis, London, 2000 (Chapter 3: "Computing with Membranes").

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

[3] J. Dassow, Gh. Păun, On the power of membrane computing, *J. of Universal Computer Sci.*, 5, 2 (1999), 33–49 (www.iicm.edu/jucs).

[4] M. R. Garey, D. J. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Comp., San Francisco, 1979.

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

[6] M. Ito, C. Martin-Vide, Gh. Păun, A characterization of Parikh sets of ET0L languages in terms of P systems, 2000.

[7] 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.

[8] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, in press, and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).

[9] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, 67 (1999), 139–152.

[10] Gh. Păun, (DNA) Computing by carving, *Research Report CTS-97-17*, Center for Theoretical Study of the Czech Academy of Sciences, Prague, 1997, and (in a revised form) *Soft Computing*, 3, 1 (1999), 30–36

[11] Gh. Păun, Computing with membranes – A variant: P systems with polarized membranes, *Intern. J. of Foundations of Computer Science*, 11, 1 (2000), 167–182.

[12] Gh. Păun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages and Combinatorics*, 5 (2000), and *Auckland University, CDMTCS Report* No 102, 1999 (www.cs.auckland.ac.nz/CDMTCS).

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

[14] Gh. Păun, Y. Suzuki, H. Tanaka, P Systems with energy accounting, submitted, 2000.

[15] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

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

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

[18] Y. Suzuki, H. Tanaka, Order parameter for a symbolic chemical system, *Artificial Life VI*, MIT Press, 1998, 130–139.

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

[20] Y. Suzuki, H. Tanaka, Chemical evolution among artificial proto-cells, *Artificial Life VII*, MIT Press, 2000, to appear.

[21] T. Yokomori, Computation = Self-assembly + conformational change: Toward new computing paradigms, *DLT, 99*, Aachen, 1999, 21–30.