On the Computational Efficiency of Polarizationless Recognizer P Systems with Strong Division and Dissolution

Claudio Zandron[1], Alberto Leporati[1], Claudio Ferretti[1], Giancarlo Mauri[1], Mario J. Pérez-Jiménez[2]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano – Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
E-mails: {zandron,leporati,ferretti,mauri}@disco.unimib.it

[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: marper@us.es

Summary. Recognizer P systems with active membranes have proven to be very powerful computing devices, being able to solve **NP**-complete decision problems in a polynomial time. However such solutions usually exploit many powerful features, such as electrical charges (polarizations) associated to membranes, evolution rules, communication rules, and strong or weak forms of division rules. In this paper we contribute to the study of the computational power of polarizationless recognizer P systems with active membranes. Precisely, we show that such systems are able to solve in polynomial time the **NP**-complete decision problem 3-SAT by using only dissolution rules and a form of strong division for non–elementary membranes, working in the maximal parallel way.

1 Introduction

Membrane systems (also known as *P systems*) have been introduced in [11] as a parallel, nondeterministic, synchronous and distributed model of computation inspired by the structure and functioning of living cells. The basic model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain some *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. Usually, the rules are applied in a nondeterministic and maximally parallel way; moreover, all the objects that may evolve are forced to evolve. A *computation* starts from an initial configuration of the system and terminates

when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane*, or emitted from the skin of the system. An interesting subclass of membrane system is constituted by *recognizer* P systems, in which: (1) all computations halt, (2) only two possible outputs exist (usually named `yes` and `no`), and (3) the result produced by the system depends only upon its input, and is not influenced by the particular sequence of computation steps taken to produce it. For a systematic introduction on P systems we refer the reader to [14], whereas the latest information can be found in [24].

Since the introduction of membrane systems, many investigations have been performed on their computational properties: in particular, many variants have been proposed in order to study the contribution of various ingredients (associated with the membranes and/or with the rules of the system) to the achievement of the computational power of these systems. In this respect, it is known [19, 23, 5] that the class of all decision problems which can be solved in polynomial time by a family of recognizer P systems that use only basic rules, that is, evolution, communication and rules involving membrane dissolution, coincides with the standard complexity class **P**. Hence, in order to efficiently solve **NP**-complete problems by means of P systems it seems necessary to be able to construct an exponential workspace, expressed by the number of membranes, in polynomial time. In particular, two features have proven to be of paramount importance in establishing whether a membrane system is able to solve **NP**-complete decision problems in polynomial time: membrane division and dissolution. The former is inspired from the biological process called *mitosis*: using *division rules* we can duplicate a given membrane that contains one specified symbol, possibly rewriting this symbol in a different way in each of the cells produced by the process. All the other symbols, as well as the rules, which are contained in the original cell are copied unaltered into each of the resulting cells. As for the membranes eventually contained in the original cell, we can make the following distinctions. If no membrane occurs, the we say that the division is *elementary*; if at least one membrane occurs, then the division is non elementary, and we have to specify how the membranes are distributed to the resulting membranes. If all the membranes are copied to each of the resulting membranes, then we have a *weak* (non-elementary) division; if, on the other hand, we can choose what membranes are copied into each of the resulting membranes, then we have *strong* (non-elementary) division. Membrane dissolution is performed by rules that simply dissolve the surrounding membrane when a specified symbol occurs.

Recognizer P systems with active membranes (using division rules and, eventually, polarizations associated to membranes) have thus been successfully used to efficiently solve **NP**-complete problems. The first solutions were given in the so called *semi-uniform* setting [13, 23, 7, 9], which means that we assume the existence of a deterministic Turing machine that, for every instance of the problem, produces in polynomial time a description of the P system that solves such an instance. The solution is computed in a *confluent* manner, meaning that the instance given in input is positive if and only if every computation of the P system

associated with it is an accepting computation. Another way to solve **NP**-complete problems by means of P systems is by considering the *uniform* setting, in which all the instances of the problem are given in input — encoded in an appropriate way — to the same P system and then solved by it. Sometimes, a uniform solution to a decision problem Q is provided by defining a family $\{\Pi_Q(n)\}_{n\in\mathbb{N}}$ of P systems such that for every $n \in \mathbb{N}$ the system $\Pi_Q(n)$ reads in input an encoding of any possible instance of size n, and solves it. P systems with active membranes have thus been successfully used to design uniform polynomial-time solutions to some well-known **NP**-complete problems, such as SAT [20], SUBSET SUM [17], KNAPSACK [18], PARTITION [6] and the COMMON ALGORITHMIC PROBLEM [21].

All the papers mentioned above deal with P systems with three polarizations that use only division rules for elementary membranes (in [22] also division for non–elementary membranes is permitted, and in this way a semi–uniform solution to the **PSPACE**-complete problem QSAT is provided), and working in the *maximal parallel* way. As shown in [1], the number of polarizations can be decreased to two without loss of efficiency.

Since by using all these features (membrane division, dissolution and polarizations) we can solve **NP**-complete problems, we have a model of computation which is considered too powerful from the point of view of traditional complexity theory. Hence a research direction of a clear interest is to selectively remove one or more of these features and see whether the computation power changes, that is, investigating for what combinations of features we are still able to obtain polynomial time solutions to computationally hard problems and what features, once removed, only allow to obtain polynomial time solutions to tractable problems, in the classical sense. In this direction, in [15] a conjecture was formulated by Gh. Păun about the computational power of polarizationless P systems with active membranes and working in the maximally parallel mode, stating that such systems can only solve decision problems that are in **P** (by using only elementary division), and some partial answers were given in [8]. Also, in [4] the computational power of recognizer P systems with active membranes but without electrical charges and dissolution rules was investigated, establishing that they characterize the complexity class **P**.

In this paper we continue this research line, showing that polarizationless P systems with active membranes that use strong division for non–elementary membranes and dissolution rules, working in the maximal parallel way, are able to solve in polynomial time the **NP**-complete problem 3-SAT. This result provides further partial answers to Păun's conjecture, establishing that neither evolution nor communication rules, and no electrical charges are needed to solve **NP**-complete problems, provided that we can use strong division rules for non–elementary membranes (as well as dissolution rules, otherwise we would fall in the case considered in [4]).

The paper is organized as follows. In Sections 2 and 3 we recall the definition of polarizationless recognizer P systems with active membranes, thus establishing our model of computation, and we recall the definition of the **NP**-complete decision problem 3-SAT. In Section 4 we show how the systems we are considering are able

to solve the 3-SAT problem. Finally, Section 5 contains the conclusions and some directions for further research.

2 Polarizationless recognizer P systems with active membranes

Usually, P systems with active membranes are defined in the literature with three electrical charges (also called *polarizations*) associated with membranes (even thought two charges suffice, as proved in [1]) to control the application of the rules, which can be of the following types: *evolution rules*, by which single objects evolve to a multiset of objects, *communication rules*, by which an object is introduced in or expelled from a membrane, and possibly changed to another object while performing this operation, *dissolution rules*, by which a membrane is dissolved under the influence of an object, that can also be modified during this operation, and *membrane division rules* (both for elementary and non–elementary membranes, or only for elementary membranes). However, in this paper we will consider *polarizationless* P systems with active membranes, that is, P systems in which no electrical charge is associated with any membrane.

Formally, a P system with polarizationless active membranes of the initial degree $n \geq 1$ is a tuple of the form $\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_n, R, h_0)$, where:

1. Γ is the alphabet of objects;
2. H is a finite set of labels for membranes;
3. μ is a membrane structure, consisting of n membranes being labelled with elements of H;
4. $\mathcal{M}_1, \ldots, \mathcal{M}_n$ are strings over Γ, describing the multisets of objects placed in the n initial regions of μ;
5. R is a finite set of developmental rules, of the following forms:
   (a) $[\, a \to v \,]_h$, for $h \in H$, $a \in \Gamma$, $v \in \Gamma^*$ (object evolution rules);
   (b) $a[\ \ ]_h \to [\, b\,]_h$, for $h \in H$, $a, b \in \Gamma$ (in communication rules);
   (c) $[\, a\,]_h \to b[\ \ ]_h$, for $h \in H$, $a, b \in \Gamma$ (out communication rules);
   (d) $[\, a\,]_h \to b$, for $h \in H$, $a, b \in \Gamma$ (dissolution rules);
   (e) $[\, a\,]_h \to [\, b\,]_h [\, c\,]_h$, for $h \in H$, $a, b, c \in \Gamma$ (weak division rules for elementary or non–elementary membranes);
   (f) $h_0 \in H$ or $h_0 = env$ indicates the output region (in the latter case, usually h_0 does not appear in the description of the system).

We can also consider rules of the form $[\,[\ \ ]_{h_1}[\ \ ]_{h_2}\,]_{h_3} \to [\,[\ \ ]_{h_1}\,]_{h_3}[\,[\ \ ]_{h_2}\,]_{h_3}$, where h_1, h_2, h_3 are labels from H: if the membrane with label h_3 contains other membranes than those with labels h_1, h_2, these membranes and their contents are duplicated and placed in both new copies of the membrane h_3; all membranes and objects placed inside membranes h_1, h_2, as well as the objects from membrane h_3 placed outside membranes h_1 and h_2, are reproduced in the new copies of membrane h_3. These rules are called *strong division rules for non–elementary membranes*.

As usual, a computation starts in the *initial configuration*, which is given by the membrane structure μ and the strings (multisets) $\mathcal{M}_1, \ldots, \mathcal{M}_n$ of objects initially present in the n regions of μ. Using the *maximally parallel manner*, at each computation step (a global clock is assumed) in each region of the system we apply the rules in such a way that no further rule can be applied to the remaining objects or membranes. In each step, each object and each membrane can be involved in only one rule. The application of a maximal set of rules during a computation step produces a new configuration of the system. A *computation* is a sequence $C_0, C_1, \ldots$ of configurations such that C_0 is the initial configuration described above, and for all $i \geq 1$ the configuration C_i is obtained from C_{i-1} by applying a maximal set of rules as described above. Note that a computation may be finite or infinite; in the former case we require that the last element of the sequence is an *halting* configuration, that is, a configuration in which no rule can be applied anywhere in the system. A halting computation provides a result encoded by the objects present in region h_0 at the end of the computation; this is a region of the system if $h_0 \in H$ (and in this case, for a computation to be successful, exactly one membrane with label h_0 should be present in the halting configuration), or it is the environment if $h_0 = env$. An infinite computation produces no result.

A *recognizer* P system with active membranes is obtained from the definition given above by assuming that the system halts on every computation and produces one of two possible outputs, that are usually denoted by `yes` and `no`. A further requirement is that the system is *confluent*, that is, for any given input configuration, all the computations that can start with such a configuration end by producing the same output. In this way, we can say that a recognizer P system with active membranes recognizes the language which is composed by the strings that encode the initial configurations that produce `yes` as a result. By considering the trivial bijection existing between these languages and decision problems, we can also say that a recognizer P system solves the decision problem whose positive instances are associated with initial configurations of the system that produce the output `yes` in h_0.

We denote by $\mathcal{AM}^0$ the class of polarizationless recognizer P systems with active membranes, and we denote by $\mathcal{AM}^0(\alpha, \beta, \gamma, \delta)$, where $\alpha \in \{-d, +d\}$, $\beta \in \{-ne, +new, +nes\}$, $\gamma \in \{-ev, +ev\}$, and $\delta \in \{-comm, +comm\}$ the class of all recognizer P systems with polarizationless active membranes such that: (a) if $\alpha = +d$ (resp., $\alpha = -d$) then dissolution rules are permitted (resp., forbidden); (b) if $\beta \in \{+new, +nes\}$ (resp., $\beta = -ne$) then division rules for elementary and non–elementary membranes, weak or strong (resp., only division rules for elementary membranes) are permitted; (c) if $\gamma = +ev$ (resp., $\gamma = -ev$) then evolution rules are permitted (resp., forbidden); (d) if $\delta = +comm$ (resp., $\delta = -comm$) then communication rules are permitted (resp., forbidden).

The class of all decision problems which can be solved in uniform (resp., semi–uniform) way, and in polynomial time by a family $\mathcal{R}$ of recognizer membrane systems is denoted by $\mathbf{PMC}_{\mathcal{R}}$ (resp., $\mathbf{PMC}^*_{\mathcal{R}}$). The following inclusions directly follow from these definitions.

Proposition 1. *For all* $\alpha \in \{-d, +d\}$, $\beta \in \{-ne, +new, +nes\}$, $\gamma \in \{-ev, +ev\}$, $\delta \in \{-comm, +comm\}$ *and* $\varepsilon \in \{*, \lambda\}$*:*

1. $\mathbf{PMC}_{\mathcal{AM}^0(\alpha,\beta,\gamma,\delta)} \subseteq \mathbf{PMC}^{*}_{\mathcal{AM}^0(\alpha,\beta,\gamma,\delta)}$
2. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(-d,\beta,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(+d,\beta,\gamma,\delta)}$
3. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,-ne,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,+new,\gamma,\delta)}$
4. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,-ne,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,+nes,\gamma,\delta)}$
5. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,-ev,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,+ev,\delta)}$
6. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,\gamma,-comm)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,\gamma,+comm)}$

where $\varepsilon = *$ *(resp.,* $\varepsilon = \lambda$*, the empty string) means that the complexity classes are associated with semi–uniform (resp., uniform) solutions.*

Also, using this notation, Păun's conjecture (problem **F** in [15]) can be restated as follows:

$$\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(+d,-ne,+ev,+comm)} = \mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,-ne,+ev,+comm)}$$

As stated in the Introduction, results in [4] and [8] proved the following theorem, considering a reachability problem (is the state in which the symbol `yes` is expelled to the environment reachable?) defined on the so called *dependency graph*. We refer the reader to [4] and [8] for further details on the proofs.

Theorem 1. *For all* $\beta \in \{-ne, +new, +nes\}$*,*

$$\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(-d,\beta,+ev,+comm)} = \mathbf{PMC}^{*}_{\mathcal{AM}^0(-d,\beta,+ev,+comm)}$$

This result holds for systems working in the maximal parallel manner; in [8] also systems working with *minimal parallelism* were considered, but in this paper we will not address them.

3 The 3-SAT problem

Let us now consider the **NP**-complete decision problem 3-SAT [3, p. 46]. The instances of 3-SAT depend upon two parameters: the number n of variables, and the number m of 3-clauses. We recall that a *clause* is a disjunction of literals, occurrences of x_i or $\neg x_i$, built on a given set $X = \{x_1, x_2, \ldots, x_n\}$ of boolean variables. A *3-clause* is a clause that contains exactly three literals. In what follows we will require that no repetitions of the same literal may occur in any clause. Without loss of generality we can also avoid the clauses in which both the literals x_i and $\neg x_i$, for any $1 \leq i \leq n$, occur. An *assignment* of the variables $x_1, x_2, \ldots, x_n$ is a mapping $a : X \to \{0, 1\}$ that associates to each variable a truth value. The number of all possible assignments to the variables of X is 2^n. We say that an assignment *satisfies* the clause C if, assigned the truth values to all the variables which occur in C, the evaluation of C (considered as a boolean formula) gives 1 (*true*) as a result.

We can now formally state the 3-SAT problem as follows.

Problem 1. NAME: 3-SAT.

- INSTANCE: a set $C = \{C_1, C_2, \ldots, C_m\}$ of 3-clauses, built on a finite set $\{x_1, x_2, \ldots, x_n\}$ of boolean variables.
- QUESTION: is there an assignment of the variables $x_1, x_2, \ldots, x_n$ that satisfies all the clauses in C?

In what follows we will sometimes equivalently say that an instance of 3-SAT is a propositional formula $\gamma_{n,m} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, expressed in the conjunctive normal form as a conjunction of m clauses, where each clause is a disjunction of three literals built using the boolean variables $x_1, x_2, \ldots, x_n$. With a little abuse of notation, from now on we will denote by 3-SAT(n, m) the set of instances of 3-SAT which have n variables and m clauses.

The reason for which we are here interested into 3-SAT (rather that with the more generic problem SAT, see [3, p. 39], where we put no upper bound on the number of literals that may appear in each clause) is that the number of possible 3-clauses which can be built using n boolean variables is $2n \cdot (2n-2) \cdot (2n-4) \in \Theta(n^3)$, a polynomial quantity with respect to n. This quantity is obtained by looking at a 3-clause as a triple, and observing that each component of the triple may contain one of the $2n$ possible literals, with the constraints that we do not allow neither the repetition of literals in the clauses, nor the use of the same variable two or three times in a clause. On the other hand, an instance of SAT may have a number of clauses which is exponential in n, since for every $i \in \{1, 2, \ldots, n\}$ either variable x_i or its negation (or none of them) can appear in a clause, yielding to 3^n possible combinations.

4 Solving 3-SAT with strong division and dissolution rules

In this section we propose a semi–uniform family $\{\Pi_{3SAT}(\gamma_{n,m})\}_{\gamma_{n,m} \in 3SAT(n,m)}$ of polarizationless recognizer P systems with active membranes that solves the **NP**-complete decision problem 3-SAT by using only membrane dissolution rules and a form of strong division rules for non–elementary membranes. Precisely, for every instance $\gamma_{n,m}$ of 3-SAT(n, m) we show how to build the system $\Pi_{3SAT}(\gamma_{n,m})$ that solves such an instance. Our result can be summarized by the statement of the following theorem.

Theorem 2. *3-SAT* $\in \mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,+nes,-ev,-comm)}$.

Proof. Let $\gamma_{n,m} = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an instance of 3-SAT(n, m), built using the boolean variables $x_1, x_2, \ldots, x_n$, and let $\Pi_{3SAT}(\gamma_{n,m})$ be the recognizer P system associated (in the semi–uniform framework) to $\gamma_{n,m}$, whose initial configuration is illustrated in Figure 1. The system is composed by m outer membranes (not counting the skin membrane) which are associated with the clauses of $\gamma_{n,m}$. Precisely, the membrane immediately contained in the skin is associated with clause C_m and contains membrane C_{m-1}, which is associated with the namesake clause;

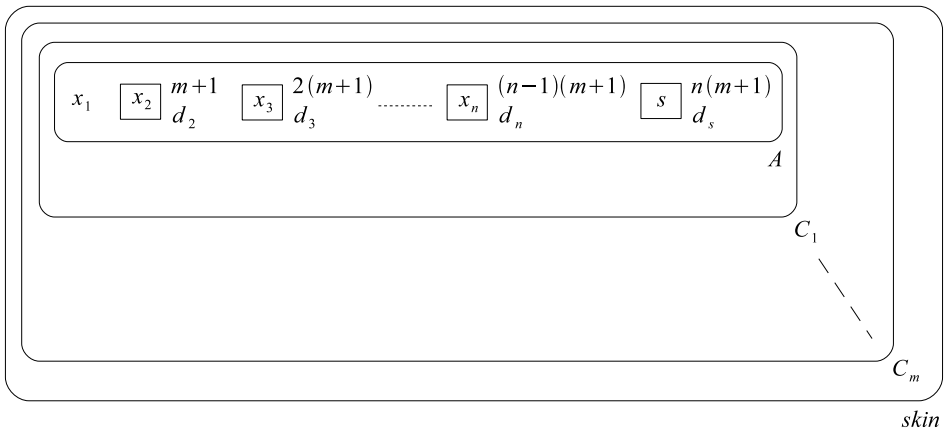

Fig. 1. Initial configuration of the system $\Pi_{3SAT}(\gamma_{n,m})$ that solves the instance $\gamma_{n,m}$ of 3-SAT(n, m)

on its turn, membrane C_{m-1} contains a membrane labelled with C_{m-2}, and so on, until we reach membrane C_1 that contains a membrane labelled with A, that will be used to generate all the possible assignments to $x_1, x_2, \ldots, x_n$. Membrane A contains the object x_1 (that represents the namesake variable) as well as n hierarchies of nested membranes. As depicted in Figure 2, the notation $\boxed{x_i}^{k}_{d_i}$ that we

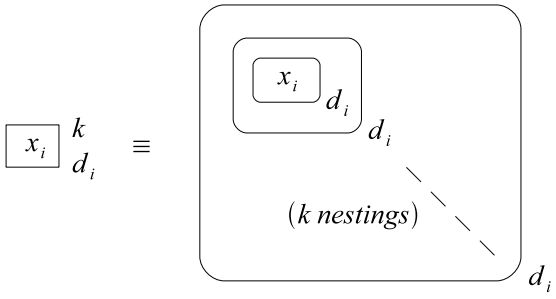

Fig. 2. The hierarchies of nested membranes used in the system depicted in Figure 1 to perform the correct sequence of membrane divisions

have adopted in Figure 1 indicates that symbol x_i is surrounded by k membranes, nested one into the other, all labelled by d_i. In this way, we can operate on membrane A through a rule which is activated by x_1 and, in the meanwhile, dissolve one membrane in each of the subsystems contained in A. After $m + 1$ steps x_2 emerges and activates another rule of A, and so on, until symbol s emerges and starts another phase of computation.

The computation of the system is composed by two phases: the *generation stage* and the *verification stage*. During the generation stage, 2^n copies of the subsystem contained into the skin of the initial configuration depicted in Figure 1 are produced, where in each copy membrane A contains an encoding of one of the possible assignments to $x_1, x_2, \ldots, x_n$. Such a phase is performed by the following rules:

1. $[\,[\,]_A[\,]_A\,]_{C_1} \rightarrow [\,[\,]_A\,]_{C_1}\,[\,[\,]_A\,]_{C_1}$
2. $[\,[\,]_{C_{i-1}}[\,]_{C_{i-1}}\,]_{C_i} \rightarrow [\,[\,]_{C_{i-1}}\,]_{C_i}\,[\,[\,]_{C_{i-1}}\,]_{C_i}$ for all $i = 2, 3, \ldots, m$
3. $[\,x_j\,]_A \rightarrow [\,t_j\,]_A\,[\,f_j\,]_A$ for all $j = 1, 2, \ldots, n$
4. $[\,x_j\,]_{d_j} \rightarrow x_j$ for all $j = 1, 2, \ldots, n$
5. $[\,s\,]_{d_s} \rightarrow s$
6. $[\,s\,]_A \rightarrow \texttt{yes}$
7. $[\,t_j\,]_{C_i} \rightarrow t_j$ if $x_j \in C_i$, $[\,f_j\,]_{C_i} \rightarrow f_j$ if $\neg x_j \in C_i$, where $1 \leq i \leq m$

Rules 1 and 2 are strong division rules for non–elementary membranes: whenever a membrane C_i contains two membranes at their immediately inner level, it divides and each of the resulting copies contains one of the previous inner membranes. Rule 3 is used to generate the assignments: when the symbol x_j, for $j \in \{1, 2, \ldots, n\}$, occurs in membrane A then A divides; in one of the resulting copies the symbol x_j is rewritten to t_j, indicating the fact that we are assigning the value TRUE to the boolean variable x_j. Similarly, in the other copy of A the symbol x_j is rewritten to f_j, indicating that the boolean value FALSE is assigned to x_j. In order to control the order of application of division rules during the generation phase, only one symbol x_j occurs in membrane A every $m + 1$ computation steps. In this way we first divide membrane A, assigning the two boolean values TRUE and FALSE to x_j as described above; then, rule 1 can be applied, thus duplicating membrane C_1.

In the subsequent $m - 1$ computation steps, membranes $C_2, C_3, \ldots, C_m$ are duplicated exactly in this order thanks to rules 2. Figure 3 depicts the first steps of this process for an instance containing $n = 2$ variables and $m = 2$ clauses (note that this example is conceived only for illustrative purposes, since at least three boolean variables are needed to build valid 3-clauses).

The rules are applied in the maximal parallel manner. In particular, at every computation step one membrane labelled with d_j, for each $j \in \{1, 2, \ldots, n\}$ such that membrane d_j still occurs in the system, is dissolved. In this way, a symbol x_j emerges in membrane A just after the assignment to x_{j-1} and all the subsequent duplications of membranes $C_1, C_2, \ldots, C_m$ have been performed. By using the same mechanism, symbol s emerges in membrane A after $n(m + 1)$ steps, that is, after all the assignments to $x_1, x_2, \ldots, x_n$ and all the duplications of membranes $C_1, C_2, \ldots, C_m$ have been performed. In practice, the construct composed by $n(m+1)$ nested membranes, all labelled with d_s, together with the symbol s into the innermost membrane and the dissolution rule $[\,s\,]_{d_s} \rightarrow s$, implement a counter whose initial value is nm and which is decremented each time the dissolution rule is applied.

When the symbol s appears in A then $n(m + 1)$ computation steps have been performed, that is, the generation stage has ended and the verification stage can

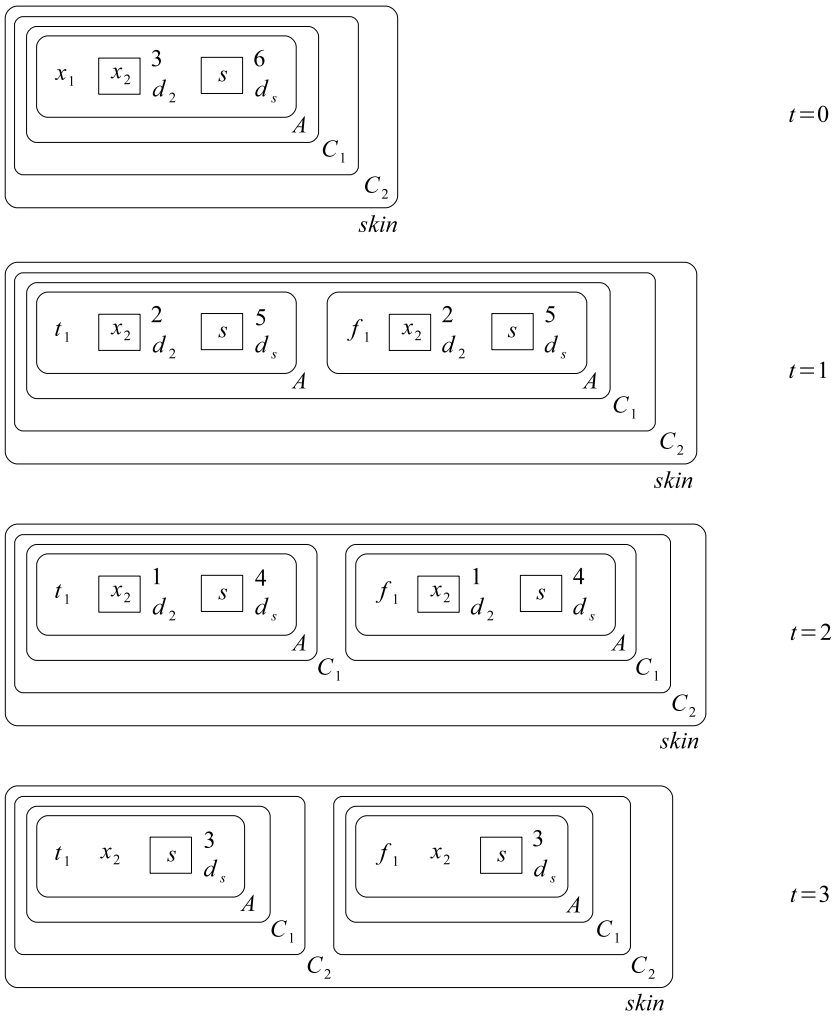

Fig. 3. First steps of the generation stage of a system designed to work on two clauses, built using two boolean variables. For reasons of space, also in this figure we have used the abbreviation depicted in Figure 2

start. All the copies of membrane A are dissolved by executing rule 6 (which also changes s to `yes`), so that all the objects t_j and f_j that represent the truth values of $x_1, x_2, \ldots, x_n$ can reach the corresponding membrane C_1 and activate its rules. These rules, of type 7, depend upon the instance $\gamma_{n,m}$ of 3-SAT(n, m) we are solving. For example, assume that the first clause of $\gamma_{n,m}$ is $C_1 = x_1 \vee \neg x_3 \vee x_4$. Then, membranes C_1 will contain the following dissolution rules:

$$[\, t_1 \,]_{C_1} \to t_1$$
$$[\, f_3 \,]_{C_1} \to f_3$$
$$[\, t_4 \,]_{C_1} \to t_4$$

In this way, a membrane labelled with C_1 is dissolved if and only if at least one of the objects t_j and f_j that encode the assignment satisfy the clause. If no object satisfy the clause then the computation in that subsystem halts; on the contrary, if the assignment under consideration satisfies C_1 then by dissolving membrane C_1 the objects t_j and f_j that encode the assignments are released to membrane C_2. Then, the rules that correspond to clause C_2 are executed; if the assignment satisfies also C_2 then the corresponding membrane is dissolved and the computation continues in membrane C_3, otherwise membrane C_2 is not dissolved and the computation halts in that subsystem. If an assignment satisfies all the clauses of $\gamma_{n,m}$ then it will dissolve all the membranes $C_1, C_2, \ldots, C_m$, and the objects that represent the assignment will reach the skin membrane, that we consider as the output membrane. Hence, the instance $\gamma_{n.m}$ of 3-SAT(n, m) solved by the system is positive if and only if in the halting configuration (in which no rule can be applied) at least one symbol occurs in the region enclosed by the skin membrane (equivalently, if at least one copy of symbol `yes` occurs in such a region).

As stated above, we have focused our attention on the 3-SAT problem because the number m of clauses is $O(n^3)$. It is apparent that the number of computation steps of the system $\Pi_{3SAT}(\gamma_{n,m})$ we have just described is $\Theta(n(m+1)+m) \subseteq O(n^4)$. The number of membranes in the initial configuration of the system is:

$$\sum_{i=1}^{n} i(m+1) + m + 2 = (m+1)\,\frac{n(n+1)}{2} + m + 2$$
$$\in \Theta(n^2 m) \subseteq O(n^5)$$

a polynomial quantity in n. The total number of rules is $2n + m + 3 \in O(n^3)$, and the initial number of objects is $n + 1 \in \Theta(n)$.

For the sake of completeness, please note that we could enlarge this system so to always produce exactly one output, being it either `yes` or `no`, just by adding an object b, initially put in the leaf membrane of a series of $n(m+1)+1$ membranes, a second outermost membrane (we can call it "external skin") enclosing our system and this new series of nested membranes, and finally adding a set of rules which first let b move toward the external skin and eventually change it to `no`. Notice that, in case of a positive answer, the object `yes` arrives to skin membrane one step before the object b does.

5 Conclusions and directions for future research

For every possible instance $\gamma_{n,m}$ of 3-SAT(n, m), having m clauses built on the boolean variables $x_1, x_2, \ldots, x_n$, we have shown how to build a polarizationless

recognizer P system $\Pi_{3SAT}(\gamma_{n,m})$ with active membranes that determines whether $\gamma_{n,m}$ is positive, that is, whether there exists an assignment to the variables $x_1, x_2, \ldots, x_n$ that satisfies all the clauses $C_1, C_2, \ldots, C_m$ of $\gamma_{n,m}$. The system works in the maximal parallel manner and, besides using no electrical charges associated with the membranes, it does not use neither evolution nor communication rules. However, it uses a form of strong division rules for non–elementary membranes, that allow to divide the content of the membrane which is being duplicated among the two resulting copies of the membrane. We can summarize the result exposed in this paper by saying that 3-SAT $\in \mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,+nes,-ev,-comm)}$.

A first related question that comes to our mind is the following: is the class $\mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,+nes,-ev,-comm)}$ closed under polynomial reductions? If so, any problem in **NP** could (at least, in principle) be transformed to 3-SAT by a polarizationless P system with active membranes that performs its computations without leaving this class. As a result, we could conclude that $\mathbf{NP} \subseteq \mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,+nes,-ev,-comm)}$. Another question of clear interest is: can we use only *weak* division rules? Stated otherwise: is 3-SAT (or some other **NP**-complete problem) in $\mathbf{PMC}^{*}_{\mathcal{AM}^0(+d,+new,-ev,-comm)}$?

Acknowledgements

The ideas exposed in this paper emerged during the Sixth Brainstorming Week on Membrane Computing, held in Seville from February 4 to February 8, 2008.

The work of the authors was partially supported by the project "Azioni Integrate Italia-Spagna - Theory and Practice of Membrane Computing" (Acción Integrada Hispano-Italiana HI 2005-0194).

References

1. A. Alhazov, R. Freund. On efficiency of P systems with active membranes and two polarizations. In G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (eds.), *Membrane Computing, 5th International Workshop, WMC 2004*, Revised Selected and Invited Papers, LNCS 3365, Springer-Verlag, Berlin, 2005, pp. 81–94.
2. A. Alhazov, M.J. Pérez-Jiménez. Uniform solution to QSAT using polarizationless active membranes. In M.A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núñez, F.J. Romero-Campero (eds.), *Proceedings of the Fourth Brainstorming Week on Membrane Computing*, Volume I, Fénix Editora, Sevilla, 2006, pp. 29–40.
3. M.R. Garey, D.S. Johnson. *Computers and Intractability. A Guide to the Theory on NP–Completeness*. W.H. Freeman and Company, 1979.
4. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero. On the power of dissolution in P systems with active membranes. In R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa (eds.) *Membrane Computing, 6th International Workshop, WMC 2005*, Revised Selected and Invited Papers, LNCS 3850, Springer-Verlag, Berlin, 2006, pp. 224–240.

5. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, A. Romero-Jiménez. Characterizing tractability by cell-like membrane systems. In K.G. Subramanian, K. Rangarajan, M. Mukund (eds.), *Formal models, languages and applications*, World Scientific, Series in Machine Perception and Artificial Intelligence, Vol. 66, 2006, pp. 137–154.
6. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez. A fast P system for finding a balanced 2–partition. *Soft Computing*, 9(9):673–678, 2005.
7. S.N. Krishna, R. Rama. A variant of P systems with active membranes: Solving **NP**-complete problems. *Romanian Journal of Information Science and Technology*, 2(4):357–367, 1999.
8. G. Mauri, M.J. Pérez-Jiménez, C. Zandron. On a Păun conjecture in membrane systems. In J. Mira and J.R. Álvarez (eds.), *Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007*, Part I, LNCS 4527, Springer-Verlag, Berlin, 2007, pp. 180–192.
9. A. Obtulowicz. Deterministic P systems for solving SAT problem. *Romanian Journal of Information Science and Technology*, 4(1–2):551–558, 2001.
10. C.H. Papadimitriou. *Computational Complexity*, Addison-Wesley, 1994.
11. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 1(61):108–143, 2000. See also Turku Centre for Computer Science – TUCS Report No. 208, 1998. Available at: `http://www.tucs.fi/Publications/techreports/TR208.php`
12. Gh. Păun. Computing with membranes. An introduction. *Bulletin of the EATCS*, 67:139–152, February 1999.
13. Gh. Păun. P systems with active membranes: Attacking **NP**-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.
14. Gh. Păun. *Membrane Computing. An Introduction.* Springer–Verlag, Berlin, 2002.
15. Gh Păun. Further twenty six open problems in membrane computing. In M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan (eds.), *Proceedings of the Third Brainstorming Week on Membrane Computing*, Fénix Editora, Sevilla, 2005, pp. 249–262.
16. G. Păun, G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1), 2002, pp. 73–100.
17. M.J. Pérez-Jiménez, A. Riscos-Núñez. Solving the SUBSET SUM problem by active membranes. *New Generation Computing*, 23(4):367–384, 2005.
18. M.J. Pérez-Jiménez, A. Riscos-Núñez. A linear–time solution to the KNAPSACK problem using P systems with active membranes. In C. Martín-Vide, Gh. Păun, G. Rozenberg, A. Salomaa (eds.), *Membrane Computing, 4th International Workshop, WMC 2003*, Revised Selected and Invited Papers, LNCS 2933, Springer-Verlag, Berlin, 2004, pp. 250–268.
19. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini. The **P** versus **NP** problem through cellular computing with membranes. In N. Jonoska, Gh. Păun, G. Rozenberg (eds.), *Aspects of Molecular Computing*, LNCS 2950, Springer-Verlag, Berlin, 2004, pp. 338–352.
20. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini. A polynomial complexity class in P systems using membrane division. In E. Csuhaj-Varjú, C. Kintala, D. Wotschke, G. Vaszil (eds.), *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, 2003, pp. 284–294.

21. M.J. Pérez-Jiménez, F.J. Romero-Campero. Attacking the Common Algorithmic Problem by recognizer P systems. In M. Margenstern (ed.), *Machines, Computations and Universality*, LNCS 3354, Springer-Verlag, Berlin, 2005, pp. 304–315.
22. P. Sosik. The computational power of cell division. *Natural Computing*, 2(3):287–298, 2003.
23. C. Zandron, C. Ferretti, G. Mauri. Solving **NP**-complete problems using P systems with active membranes. In I. Antoniou, C.S. Calude, M.J. Dinneen (eds.), *Unconventional Models of Computation*, Springer-Verlag, Berlin, 2000, pp. 289–301.
24. The P systems Web page: `http://ppage.psystems.eu`