Trading Polarization for Bi-stable Catalysts in P Systems with Active Membranes

Mario J. Pérez-Jiménez and Francisco José Romero-Campero

Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
University of Sevilla,
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{Mario.Perez, Francisco-Jose.Romero}@cs.us.es

Abstract. In the last time, several efforts have been made in order to remove polarizations of membranes from P systems with active membranes; the present paper is a contribution in this respect. In order to compensate the loss of power represented by avoiding polarizations, we use bi-stable catalysts. Polarizationless systems with active membranes which use bi-stable catalysts are proven to be computationally complete and able to solve efficiently NP-complete problems. In this paper we present a solution to SAT in linear time. In order to illustrate the presented solution, we also provide a simulation with CLIPS.

1 Introduction

In membrane computing, P systems with active membranes are specially suitable to solve efficiently NP-complete problems, because of the fact that they provide membrane division, inspired from cell division. By using this operation, one can create an exponential number of membranes (working space) in linear time; in this way, we trade space for time to solve NP-complete problems (this has been reported for SAT, VALIDITY, Subset Sum, Knapsack, etc.).

One important feature of P systems with active membranes is the polarization of membranes; each membrane has an "electrical charge", positive (+), negative (−) or neutral (0). However, the electrical charges are not very realistic from a biological point of view. Because of this, several efforts are being made in order to remove the polarizations without losing the universality and the efficiency.

This paper goes into this direction of research: we remove the polarization of the membranes but on the other hand we use bi-stable catalysts. This variant of P systems with active membranes is proven to be computationally complete and able to solve NP-complete problems like SAT in linear time.

The paper is organized as follows: Section 2 introduces bi-stable catalytic P systems with active membranes without charges as generating devices and as recognizer devices. In Section 3 the complexity classes for P systems are briefly recalled. Sections 4, 5, and 6 present a cellular solution in linear time to the SAT problem within the framework of this variant of P systems. In Section 7

the programming language CLIPS is used to exhibit a simulation of the designed solution in order to illustrate how it works. Conclusions are given in Section 8.

2 Bi-stable Catalytic P Systems with Active Membranes Without Polarizations

Definition 1. *A bi-stable catalytic P system with active membranes and without polarizations is a tuple*

$$\Pi = (\Gamma, K, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_p, R),$$

where:

1. $p \geq 1$ *is the initial degree of the system;*
2. Γ *is the alphabet of symbol-objects;*
3. K *is a subset of* Γ, $K \subseteq \Gamma$, *such that if* $c \in K$ *then* $\overline{c} \in K$ *(the elements of* K *are called bi-stable catalysts);*
4. H *is a finite set of labels for membranes;*
5. μ *is a membrane structure consisting of* p *membranes labelled (not necessarily in a one-to-one manner) with elements of* H*;*
6. $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *are strings over* Γ*, describing the initial multisets of objects associated with the regions of* μ*;*
7. R *is a finite set of evolution rules, of the following forms:*
   (a) $[\, a \rightarrow \omega \,]_h$, *for* $h \in H$, $a \in \Gamma - K, \omega \in (\Gamma - K)^*$. *This is an object evolution rule, associated with a membrane labelled with* h *but not directly involving the membrane.*
   (b) $[\, ca \rightarrow c\omega \,]_h$, $[\, ca \rightarrow \overline{c}\omega \,]_h$, $[\, \overline{c}a \rightarrow \overline{c}\omega \,]_h$, $[\, \overline{c}a \rightarrow c\omega \,]_h$, *for* $h \in H$, $c \in K$ *and* $a \in \Gamma - K, \omega \in (\Gamma - K)^*$ (bi-stable catalytic evolution rules). *Such a rule is an object evolution rule involving bi-stable catalysts, associated with a membrane labelled with* h *but not directly involving the membrane.*
   (c) $a\,[\ \,]_h \rightarrow [\, b \,]_h$, *for* $h \in H$, $a, b \in \Gamma - K$ ("send in" communication rules). *An object from the region immediately outside a membrane labelled with* h *is introduced in this membrane, possibly transformed into another object.*
   (d) $[\, a \,]_h \rightarrow b\,[\ \,]_h$, *for* $h \in H$, $a, b \in \Gamma - K$ ("send out" communication rules). *An object is sent out from membrane labelled with* h *to the region immediately outside, possibly transformed into another object.*
   (e) $[\, a \,]_h \rightarrow b$, *for* $h \in H$, $a, b \in \Gamma - K$ (dissolving rules). *A membrane labelled with* h *is dissolved in reaction with an object. The skin is never dissolved.*
   (f) $[\, a \,]_h \rightarrow [\, b \,]_h\, [\, c \,]_h$, *for* $h \in H$, $a, b, c \in \Gamma - K$ (division rules for elementary membranes). *An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects.*

Note that, in contrast to [2], the bi-stable catalysts are not always flip-flop-ing from non-barred to barred versions and back, but also rules of the form $ca \rightarrow cw$

and $\bar{c}a \to \bar{c}w$ are allowed. The case when the catalysts appear only in rules of the form $ca \to \bar{c}w$ and $\bar{c}a \to cw$ is called *restricted*.

These rules are applied according to the following principles:

- All the rules are applied in parallel and in a maximal manner. In one step, one object of a membrane can be used by only one rule (chosen in a non deterministic way), but any object which can evolve by one rule of any form, should evolve.
- If a membrane is dissolved, its content (multiset and internal membranes) is left free in the surrounding region.
- 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) and (b), then we suppose that first the evolution rules of types (a) and (b) are used, and then the division is produced. Of course, this process takes only one step.
- The rules associated with membranes labelled with h are used for all copies of this membrane. At one step, a membrane labelled with h can be the subject of *only one* rule of types (c)-(f).

2.1 Bi-stable Catalytic P Systems with Active Membranes Without Polarizations, as Generating Devices

As a generating device, the result (output) of a halting configuration of a bi-stable catalytic P system is the cardinality of the multiset associated with the environment in the last configuration. In these P systems a non halting computation yields no output.

Definition 2. *We denote by $N(\Pi)$ the set of all outputs of halting computations with respect to a bi-stable catalytic P system Π.*

Theorem 1. *Restricted bi-stable catalytic P systems with active membranes without polarization, using rules of type (b) and (d), are computationally complete.*

Proof. Let L be a recursively enumerable language. Let G be a matrix grammar with appearance checking such that $L(G) = L$. We can consider that $G = (N, \{a\}, S, M, F)$ is given in Z-binary normal form, in the standard notation. That is,

- $N = N_1 \cup N_2 \cup \{S, Z, \sharp\}$, with these three sets mutually disjoint.
- The matrices in M are in one of the following forms:
  1. $(S \to XA)$, where $X \in N_1, A \in N_2$,
  2. $(X \to Y, A \to x)$, where $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
  3. $(X \to Y, A \to \sharp)$, where $X \in N_1, Y \in N_1 \cup \{Z\}, A \in N_2$,
  4. $(Z \to \lambda)$.
- $F = \{A \to \sharp \mid \exists m \in M(m = (X \to Y, A \to \sharp)\}$.
  Moreover, if the special symbol Z appears in a sentential form w, then we have $w = Zw'$, with $w' \in (T \cup \{\sharp\})^*$ (that is, no nonterminal from N_2 is present).

- The matrices in M will be ordered as follows:

$$\begin{array}{ll} m_0: & (S \to X_{init}A_{init}), \text{ with } X_{init} \in N_1, A_{init} \in N_2, \\ \left.\begin{array}{l} m_1: \\ \vdots \\ m_k: \end{array}\right\} & (X \to \alpha, A \to x), \text{ with } x \in N_1, \alpha \in N_1 \cup \{\lambda\}, A \in N_2, \\ \left.\begin{array}{l} m_{k+1}: \\ \vdots \\ m_n: \end{array}\right\} & \begin{array}{l}(X \to Y, A \to \sharp). \\ (X \to Z, A \to \sharp)\end{array} \text{ , with } X, Y \in N_1, A \in N_2 \\ m_{n+1}: & (Z \to \lambda) \end{array}$$

We construct the system

$$\Pi = (\Gamma, K, \{1\}, [\ ]_1, \mathcal{M}_1, R),$$

where:

- $\Gamma = N \cup K \cup \{a, \sharp\} \cup \{X', \overline{X}, \overline{X}' \mid X \in N_1\}$,
- $K = \{c_i, \overline{c}_i \mid 0 \le i \le n\}$,
- $\mathcal{M}_1 = \{X_{init}, A_{init}, E, c_0, c_1, \ldots, c_n\}$,
- The set R consists of the following rules:

(1.)

$$\left.\begin{array}{l} [\ c_iX \to \overline{c}_iY'\ ]_1 \\ [\ \overline{c}_iA \to c_ix\ ]_1 \\ [\ \overline{c}_iE \to c_i\sharp\ ]_1 \\ [\ c_0Y' \to \overline{c}_0Y\ ]_1 \\ [\ \overline{c}_0Y' \to c_0Y\ ]_1 \end{array}\right\} \text{ for each } m_i : (X \to Y, A \to x), \text{ with } 1 \le i \le k\ .$$

These rules simulate the matrices m_i, for $i = 1, \ldots, k$. When we have in the skin region a multiset containing X and there exists in M a matrix $m_i : (X \to Y, A \to x)$, the rule $[\ c_iX \to \overline{c}_iY'\ ]_1$ is applicable. In order to simulate the second component of the grammar one of the rules $[\ c_0Y' \to \overline{c}_0Y\ ]_1$, $[\ \overline{c}_0Y' \to c_0Y\ ]_1$ (either c_0 or $\overline{c}_0$ is present, hence one of these rules can be used) provides a step in which if there exists an object A in the skin region, then the rule $[\ \overline{c}_iA \to c_ix\ ]_1$ can be applied; otherwise, if there is no such object, the rule $[\ \overline{c}_iE \to c_i\sharp\ ]_1$ produces the trap symbol $\sharp$ showing that we can not apply this matrix and so this is not a correct derivation.

(2.)

$$\left.\begin{array}{l} [\ c_iX \to \overline{c}_i\overline{Y}'\ ]_1 \\ [\ c_0\overline{Y}' \to \overline{c}_0\overline{Y}\ ]_1 \\ [\ \overline{c}_0\overline{Y}' \to c_0\overline{Y}\ ]_1 \\ [\ \overline{c}_iA \to c_i\sharp\ ]_1 \\ [\ \overline{c}_i\overline{Y} \to c_iY\ ]_1 \end{array}\right\} \text{ for each } m_i : (X \to Y, A \to \sharp), \text{ with } k+1 \le i \le n$$

These rules simulate the matrices m_i, for $i = k+1, \ldots, n$. When we have in the skin region a multiset containing X and there exists in M a matrix $m_i : (X \to Y, A \to \sharp)$, the rule $[\ c_i X \to \overline{c}_i \overline{Y}'\ ]_1$ is applicable. In order to simulate the second component of the grammar, one of the rules $[\ c_0 \overline{Y}' \to \overline{c}_0 \overline{Y}\ ]_1$,$[\ \overline{c}_0 \overline{Y}' \to c_0 \overline{Y}\ ]_1$ provides a step in which if there exists an object A in the skin region, the rule $[\ \overline{c}_i A \to c_i \sharp\ ]_1$ is applied; otherwise, if there is no such object, then the rule $[\ \overline{c}_i \overline{Y} \to c_i Y\ ]_1$ completes the simulation of this matrix.

(3.)

$[\ a\ ]_1 \to a[\ ]_1$, $[\ c_0\sharp \to \overline{c_0}\sharp\ ]_1$, $[\ \overline{c}_0\sharp \to c_0\sharp\ ]_1$,

The first of these two last rules, $[a]_1 \to a[\ ]_1$, sends out to the environment the object a. In a halting configuration of the system the multiplicity of the object a in the environment represents the length of the word generated by G. If the computation of the system simulates a non terminal derivation in G, then the rules $[\ c_0\sharp \to \overline{c}_0\sharp\ ]_1$, $[\ \overline{c}_0\sharp \to c_0\sharp\ ]_1$ produce a non halting computation.

From the above remarks it is easy to prove that the equality $length(L(G)) = N(\Pi)$ holds, where $length(L(G))$ is the length set of the language $L(G)$, that is, $length(L(G)) = \{|u| \mid u \in L(G)\}$. □

In the previous proof we do not actually use the fact that the membranes are *active* (for instance, we do not use membrane division); otherwise stated, the proof can be easily reformulated in terms of basic transition P systems, and this makes necessary the comparison of Theorem 1 with universality results known for such systems. First, the universality is known for systems with bi-stable catalysts already from [5], where, however, one uses two membranes (see also Theorem 3.4.7 from [2]; our results improves on this point, because we use only one membrane. Then, in [1] it is proven that two catalysts are sufficient to get universality in P systems without polarizations and without priorities, but the systems considered in [1] contain both catalytic and non-catalytic rules.

2.2 Recognizer Bi-stable Catalytic P Systems with Active Membranes Without Polarization

Definition 3. *A* P system with input *is a tuple* (Π, Σ, i_Π)*, where* Π *is a P system, with working alphabet* Γ*, with* p *membranes labelled* $1, \ldots, p$*, and initial multisets* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *associated with them,* Σ *is an (input) alphabet strictly contained in* Γ*, the initial multisets are over* $\Gamma - \Sigma$*, and* i_Π *is the label of a distinguished (input) membrane.*

The computations of a P system with an input in the form of a multiset m over Σ are defined in a natural way; they start from a configuration which is obtained by adding the multiset m to the initial configuration of the system.

Definition 4. *Let* (Π, Σ, i_Π) *be a P system with input. Let* Γ *be the working alphabet of* Π*,* μ *the membrane structure and* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *the initial multisets*

of Π. Let m be a multiset over Σ. The initial configuration of (Π, Σ, i_Π) with input m *is* $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots \mathcal{M}_p)$.

In the case of P systems with input and with external output, the concept of computation is introduced in a similar way as for standard P systems – see [2] – but with a small change. We consider that it is not possible to observe the internal processes inside the P system and we can only know if the computation has halted via some distinguished objects sent out of the skin. We can formalize these ideas in the following way.

Definition 5. *A* recognizer bi-stable catalytic P system *is a P system with input, (Π, Σ, i_Π), and with external output such that:*

1. *Π is a bi-stable catalytic P system.*
2. *The working alphabet of Π contains two distinguished objects YES, NO.*
3. *All its computations halt.*
4. *If $\mathcal{C}$ is a computation of Π, then either the object YES or the object NO (but not both) is sent to the environment, and only in the last step of the computation.*

We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object YES (respectively, NO) appears in the environment in the halting configuration of $\mathcal{C}$.

In what follows we will deal with recognizer bi-stable P systems with active membranes without polarizations. Let us denote by $\mathcal{BAM}$ the class of this variant of recognizer P systems.

3 The Complexity Class $\mathbf{PMC}_{\mathcal{F}}$

The first results about "solvability" of **NP**–complete problems in polynomial time (even linear) by cellular computing systems with membranes were obtained using variants of P systems that lack an input membrane. Thus, the constructive proofs of such results need to design one system for each instance of the problem.

This drawback can be easily avoided if we consider a P system with input. Then, the same system could solve different instances of the problem, provided that the corresponding input multisets are introduced in the input membrane.

Instead of looking for a single system that solves a problem, we prefer designing a family of P systems such that each element decides all the instances of "equivalent size" of the problem.

Definition 6. *Let $\mathcal{F}$ be a class of recognizer P systems. We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}^+}$, of systems from $\mathcal{F}$, and we denote this by $X \in \mathbf{PMC}_{\mathcal{F}}$, if the following is true:*

- *The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines; that is, there exists a deterministic Turing machine constructing $\Pi(n)$ from $n \in \mathbf{N}^+$ in polynomial time.*

- *There exists a pair* (g, h) *of polynomial-time computable functions* $g : L \to \bigcup_{n \in \mathbf{N}^+} I_{\Pi(n)}$ *and* $h : L \to \mathbf{N}^+$ *such that for every* $u \in L$ *we have* $g(u) \in I_{\Pi(h(u))}$, *and*
  - *the family* $\mathbf{\Pi}$ *is polynomially bounded with regard to* (X, g, h)*; that is, there exists a polynomial function* p*, such that for each* $u \in I_X$ *every computation of* $\Pi(h(u))$ *with input* $g(u)$ *is halting and, moreover, it performs at most* $p(|u|)$ *steps;*
  - *the family* $\mathbf{\Pi}$ *is sound with regard to* (X, g, h)*; that is, for each* $u \in I_X$*, if there exists an accepting computation of* $\Pi(h(u))$ *with input* $g(u)$*, then* $\theta_X(u) = 1$*;*
  - *the family* $\mathbf{\Pi}$ *is complete with regard to* (X, g, h)*; that is, for each* $u \in I_X$*, if* $\theta_X(u) = 1$*, then every computation of* $\Pi(h(u))$ *with input* $g(u)$ *is an accepting one.*

In the above definition we have imposed to every P system $\Pi(n)$ to be *confluent*, in the following sense: every computation with the *same* input produces the *same* output.

The class $\mathbf{PMC}_{\mathcal{F}}$ is closed under polynomial–time reduction and complement, as proven, for instance, in [10].

4 Solving SAT in Linear Time

The SAT problem is the following one: *Given a boolean formula in conjunctive normal form (CNF), to determine whether or not it is satisfiable; that is, whether there exits an assignment to its variables on which it evaluates true.*

We will address the resolution of this problem via a brute force algorithm within the framework of recognizer bi-stable catalytic P systems with active membranes without charges. Our strategy will consist in:

- *Generation stage*: Using membrane division we generate all possible assignments associated with the formula.
- *Evaluation stage*: In each membrane we evaluate the formula on the assignment produced in that membrane.
- *Checking stage*: In each membrane we check wether or not the formula evaluates true on the assignment from that membrane.
- *Output stage*: Send to the environment the right answer according to the previous stage.

Let us consider the function $\langle\ ,\ \rangle$ defined by $\langle n, m\rangle = ((n+m)(n+m+1)/2)+n$ for $\varphi = C_1 \wedge \cdots \wedge C_m$ a propositional formula in CNF and $Var(\varphi) = \{x_1, \ldots, x_n\}$. The function $\langle\ ,\ \rangle$ is polynomial-time computable (it is primitive recursive and bijective from $\mathbf{N}^2$ onto $\mathbf{N}$). Also, the inverse function of $\langle\ ,\ \rangle$ is polynomial.

The family presented here is

$$\mathbf{\Pi} = \{\ (\Pi(\langle n, m\rangle), \Sigma(n, m), i(n, m)) \mid (n, m) \in \mathbb{N}^2\ \}.$$

For each element of the family, the input alphabet is

$$\Sigma(n,m) = \{x_{i,j}, \overline{x}_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq n\},$$

the input membrane is $i(n,m) = 2$, and the P system

$$\Pi(\langle n,m\rangle) = (\Gamma(n,m), K(n,m), \{1,2\}, \mu, \mathcal{M}_1, \mathcal{M}_2, R)$$

is defined as follows:

- Bi-stable catalysts:

  $K(n,m) = \{t_j, \overline{t}_j, f_j, \overline{f}_j, s_i, \overline{s}_i, ans, \overline{ans} \mid 1 \leq i \leq m, 1 \leq j \leq n\}$.
- Working alphabet:

$$\begin{aligned}\Gamma(n,m) = \Sigma(n,m) &\cup K(n,m) \cup \{v_j, p_j, n_j \mid 1 \leq j \leq n\} \\ &\cup \{c_i, r_i \mid 1 \leq i \leq m\} \cup \{no_k \mid 1 \leq k \leq n+m+3\} \\ &\cup \{\sharp, yes, YES, NO\}.\end{aligned}$$

- Membrane structure: $\mu = [_1\, [_2\ ]_2\,]_1$ (we will say that every membrane with label 2 is an *internal membrane*).
- Initial Multisets:

  $\mathcal{M}_1 = \{no_1, ans\}$,
  $\mathcal{M}_2 = \{v_1, \ldots, v_n, \overline{t}_1, \ldots, \overline{t}_n, \overline{f}_1, \ldots, \overline{f}_n, \overline{s}_1, \ldots, \overline{s}_m, c_1\}$.
- The set R consists of the following rules:
  1. $[\ v_j\ ]_2 \ \rightarrow\ [\ p_j\ ]_2\, [\ n_j\ ]_2, \quad 1 \leq j \leq n.$

     The goal of these rules is to generate an internal membrane for each assignment to the variables of the formula. The new membrane where the object p_j appears represents the assignment where $x_j = true$ and the new membrane where the object n_j appears represents the assignment where $x_j = false$.
  2. $$\left.\begin{array}{l}[\ \overline{t}_j p_j \ \rightarrow\ t_j p_j\ ]_2 \\ [\ t_j\, x_{i\,j} \ \rightarrow\ t_j\, r_i\ ]_2 \\ [\ t_j\, \overline{x}_{i\,j} \ \rightarrow\ t_j\, \sharp\ ]_2\end{array}\right\} \text{ for } \ 1 \leq i \leq m,\ 1 \leq j \leq n.$$

     The object p_j activates the catalyst t_j which "erases" the objects $\overline{x}_{i,j}$ (these objects represent the literals $\neg x_j$), but reacts with the objects $x_{i,j}$ (these objects represent the literals x_j) to produce the object r_i (this object indicates that the clause number i evaluates true on the assignment associated with the membrane).
  3. $$\left.\begin{array}{l}[\ \overline{f}_j n_j \ \rightarrow\ f_j n_j\ ]_2 \\ [\ f_j\, \overline{x}_{i\,j} \ \rightarrow\ f_j\, r_i\ ]_2 \\ [\ f_j\, x_{i\,j} \ \rightarrow\ f_j\, \sharp\ ]_2\end{array}\right\} \text{ for } 1 \leq i \leq m,\ 1 \leq j \leq n.$$

The object n_j activates the catalyst f_j which "erases" the objects $x_{i,j}$ (these objects represent the literals x_j), but reacts with the objects $\overline{x}_{i,j}$ (these objects represent the literals $\neg x_j$) to produce the object r_i (this object indicates that the clause number i evaluates true on the assignment associated with the membrane).

4. $[\ \overline{s}_i r_i \ \rightarrow\ s_i r_i\ ]_2$, for $1 \leq i \leq m$,
$[\ s_i c_i \ \rightarrow\ s_i c_{i+1}\ ]_2$, for $1 \leq i \leq m-1$,
$[\ s_m\, c_m \ \rightarrow\ s_m\, yes\ ]_2$.

   The objects c_i are counters which represent the number of clauses that evaluate true on the assignment associated with the internal membrane. So the object c_i, for $1 \leq i \leq m-1$, reacts with the catalyst s_i, which is activated by the object r_i, to produce the object c_{i+1}, and the object c_m reacts with the object r_m to produce the object yes in order to show that every clause of the formula evaluates true on the assignment associated with the internal membrane.

5. $[\ yes\ ]_2 \ \rightarrow\ yes\ [\ ]_2$,
$[\ ans\, yes \ \rightarrow\ \overline{ans} YES\ ]_1$,
$[\ YES\ ]_1 \ \rightarrow\ YES\ [\ ]_1$.

   These rules produce and send the object YES to the environment.

6. $[\ no_i \ \rightarrow\ no_{i+1}\ ]_1$, for $1 \leq i \leq n+2m+3$,
$[\ ans\, no_{n+2m+4} \ \rightarrow\ \overline{ans} NO\ ]_1$,
$[\ NO\ ]_1 \ \rightarrow\ NO\ [\ ]_1$.

   These rules produce and send out the object NO to the environment. Note that the object NO appears one step later than the object YES and that the catalyst ans get barred in the output stage in order to make sure that the system sends out the right answer.

5 An Overview of the Computation

First of all we must define a suitable pair (g, h) of polynomial-time computable functions (see Definition 6) associated with the SAT problem. Given a formula $\varphi = C_1 \wedge \ldots C_m$ in CNF such that $Var(\varphi) = \{x_1, \ldots, x_n\}$, we define $h(\varphi) = \langle n, m \rangle$ (recall the bijection mentioned in the previous section) and $g(\varphi) = \{x_{ij} \mid x_j \in C_i\} \cup \{\overline{x}_j \mid \neg x_j \in C_i\}$

Next we will informally describe how the recognizer bi-stable catalytic P system $\Pi(h(\varphi))$ with input $g(\varphi)$ works.

The computation starts with the *generation and evaluation stages*. These two stages take place in parallel following the rules from group 1 to 3. The generation of membranes is controlled by the objects v_j, for $1 \leq j \leq n$. When an object v_j is present in an internal membrane the rule in 1 is applicable and so the system produces two new membranes. In one of these two new membranes the object p_j appears encoding that in the assignment associated with

the membrane we have $x_j = true$. In the other membrane the object n_j appears to show that in the assignment associated with this membrane we have $x_j = false$.

The *evaluation stage* takes place in a similar way in every internal membrane. The object p_j (respectively n_j) representing that $x_j = true$ (respectively $x_j = false$) in the assignment associated with the internal membrane, activates the bi-stable catalyst t_j (respectively f_j). The *active catalyst* t_j (respectively f_j) according to the rules in 2 (respectively 3) reacts with the objects x_{ij} and $\overline{x}_{ij}$ to produce the objects $\sharp$ and r_i. The objects r_i represent that the clause C_i evaluates true on the assignment associated with the membrane. These two stages take place in parallel and they take n steps of division, one step to activate the catalysts and m steps to evaluate each clause, that is, an overall of at most $n + m + 1$ steps.

The *checking stage* takes place according to the rules in 4. The object r_i activates the catalyst s_i which reacts with the object c_i for $1 \leq i \leq m-1$ to produce the object c_{i+1}. The object c_i represents that the clauses $C_1, \ldots, C_{i-1}$ for $1 \leq i \leq m$, evaluate true on the assignment associated with the internal membrane. So, the catalyst s_m reacts with the object c_m to produce the object *yes*, in order to show that the whole formula evaluates true on the assignment associated with the internal membrane. As it can be seen, the *checking stage* takes one step to activate the catalysts and m steps to check that every clause evaluates true; that is, an overall of at most $m + 1$ steps.

In the output stage the rules in 4 and 5 are applied to send the correct answer to the environment. The answer YES is sent out following the rules in 5; the object *yes* is sent to the skin by the first rule, in the second rule the bi-stable catalyst *ans* reacts with the object *yes* to produce the object YES and *ans* remains barred from now on, and finally the object YES is sent out to the environment. On the other hand, following the first rule in 5, the object no_k waits $n + 2m + 4$ and, if no object *yes* has been sent to the skin, the bi-stable catalyst *ans* and no_{n+2m+4} react to produce the object NO which is sent out to the environment. Note that the object no_{n+2m+4} appears a step later than the object *yes* in order to be sure that the system sends out the right answer. Thus, the output stage takes at most 4 steps.

6 Required Resources

The presented family of recognizer bi-stable catalytic P systems solving the SAT is polynomially uniform by Turing machines. Note that the definition of the family is done in a recursive manner starting from a given instance, in particular, from the constants n and m. Furthermore, the required resources to build the element $\Pi(\langle n, m\rangle)$ of the family are the following:

- Size of the alphabet: $2nm + 8n + 5m + 9 \in O((max\{n, m\})^2)$.
- Number of membranes: $2 \in \Theta(1)$.
- $|\mathcal{M}_1| + |\mathcal{M}_2| = 3n + m + 3 \in O(n + m)$.

- Sum of the rules lengths: $32nm + 27n + 32m + 60 \in O((max\{n, m\})^2)$.

The number of steps in each stage in the worst case are the following:

1. *Generation and evaluation stage:* $n + m + 1$ steps.
2. *Checking stage:* $m + 1$ steps.
3. *Output stage:* 4 steps.

Therefore, the overall number of steps is $n + 2m + 6 \in O(max\{n, m\})$.

From the above discussion we deduce the following results:

Theorem 2.

1. $SAT \in \mathbf{PMC}_{\mathcal{BAM}}$.
2. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{BAM}}$, *and* $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{BAM}}$.

Proof. In order to prove the theorem, it suffices to make the following remarks: the SAT problem is $\mathbf{NP}-$complete, $SAT \in \mathbf{PMC}_{\mathcal{BAM}}$ and the class $\mathbf{PMC}_{\mathcal{BAM}}$ is closed under polynomial-time reduction, and under complement. □

7 A CLIPS Session with $\varphi = (\, x_1 \vee \neg x_2 \,) \wedge (\, \neg x_1 \vee x_2 \,)$

In this section we illustrate how the designed family of recognizer bi-stable catalytic P systems works by presenting a simulation with CLIPS for the instance $\varphi = (\, x_1 \vee \neg x_2 \,) \wedge (\, \neg x_1 \vee x_2 \,)$.

```
Configuration number: 1

[environment [multiset ]]
[skin     [children 3 4]
          [label 1] [multiset ans , no 2]]
[membrane
    [number 4] [children ] [father 1]
    [label 2]  [multiset v 1 , n 2 , t- 1 , t- 2 , f- 1 , f- 2 ,
                         s- 1 , s- 2 , c 1 ,
                         x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]
[membrane
    [number 3] [children ] [father 1]
    [label 2] [multiset v 1 , p 2 , t- 1 , t- 2 , f- 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]
```

Here it can be seen how the generation stage takes place. In the presence of the object v_2 the system produces two new membranes. The membrane number 4, where the object n_2 appears, indicates that in the assignment associated with this membrane we have $x_2 = false$. The membrane number 3, where the object p_2 appears, indicates that in the assignment associated with this membrane we have $x_2 = true$.

```
Configuration number: 2

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset ans , no 3]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset n 1 , # , t- 1 , t- 2 , f- 1 , f 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset p 1 , # , t- 1 , t- 2 , f- 1 , f 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset n 1 , # , t- 1 , t 2 , f- 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset p 1 , # , t- 1 , t 2 , f- 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , x 2 2]]

Configuration number: 3

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset ans , no 4]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , # , -x 2 1 , x 2 2]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , r 1 , -x 2 1 , x 2 2]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 ,
                        x 1 1 , -x 1 2 , -x 2 1 , r 2]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s- 1 , s- 2 , c 1 ,
```

```
                        x 1 1 , r 1 , -x 2 1 , x 2 2]]

Configuration number: 4

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset ans , no 5]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 1 , # , r 1 , -x 2 1 , #]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s- 1 , s- 2 , c 1 , r 1 , # , -x 2 1 , r 2]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , -x 2 1 , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s- 2 , c 1 , x 1 1 , # , r 2 , #]]
```

At the end of the generation and evaluation stage it can be seen that the assignment associated with the internal membranes are: $\{x_1 = false, x_2 = false\}$ with the membrane number 8, $\{x_1 = false, x_2 = true\}$ with the membrane number 6, $\{x_1 = true, x_2 = false\}$ with the membrane number 7, and $\{x_1 = true, x_2 = true\}$ with the membrane number 5.

```
Configuration number: 5

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset ans , no 6]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s 1 , s 2 , c 1 , # , # , # , #]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 2 , # , r 1 , # , #]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , r 2 , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s 2 , c 2 , # , # , # , #]]
```

```
Configuration number: 6

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset ans , no 7]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 2 , # , r 1 , # , #]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , r 2 , #]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s 1 , s 2 , c 2 , # , # , # , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s 2 , yes , # , # , # , #]]
```

As a result of the *checking stage* the object YES is produced and sent out to the environment in the output stage.

```
Configuration number: 7

[environment [multiset ]]
[skin     [children 5 6 7 8]
          [label 1] [multiset yes , ans , no 8]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 2 , # , r 1 , # , #]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , r 2 , #]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s 1 , s 2 , yes , # , # , # , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s 2 , # , # , # , #]]

Configuration number: 8

[environment [multiset ]]
```

```
[skin     [children 5 6 7 8]
          [label 1] [multiset yes , YES , ans- , no 9]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 2 , # , r 1 , # , #]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , r 2 , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s 2 , # , # , # , #]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s 1 , s 2 , # , # , # , #]]

Configuration number: 9

[environment [multiset YES]]
[skin     [children 5 6 7 8]
          [label 1] [multiset yes , ans- , no 10]]
[membrane
    [number 7] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t- 2 , f- 1 , f 2 ,
                        s 1 , s- 2 , c 2 , # , r 1 , # , #]]
[membrane
    [number 6] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t 2 , f 1 , f- 2 ,
                        s- 1 , s 2 , c 1 , # , # , r 2 , #]]
[membrane
    [number 8] [children ] [father 1]
    [label 2] [multiset # , # , t- 1 , t- 2 , f 1 , f 2 ,
                        s 1 , s 2 , # , # , # , #]]
[membrane
    [number 5] [children ] [father 1]
    [label 2] [multiset # , # , t 1 , t 2 , f- 1 , f- 2 ,
                        s 1 , s 2 , # , # , # , #]]

The system has reached a halting configuration in the step number
9 and the element YES has been released into the environment.
```

8 Conclusions

In this paper we have presented a variant of P systems with active membrane in which we have traded polarization for bi-stable catalysts. We have proven

that this variant is computationally complete and able to solve efficiently NP-complete problems like SAT.

Future projects are to design families of recognizer bi-stable catalytic P systems to solve numerical NP-complete problems like Knapsack and Tripartite Matching and to study the computational power and efficiency of P systems with active membranes without polarizations.

CLIPS has been shown to be a convenient programming language for simulating P systems and it was helpful to debug the design and to understand how the P systems from the family **Π** work.

Acknowledgement

This work is supported by the Ministerio de Ciencia y Tecnología of Spain, by the *Plan Nacional de I+D+I (2000–2003)* (TIC2002-04220-C03-01), cofinanced by FEDER funds, and by a FPI fellowship (of the second author) from the University of Seville.

References

1. R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: Two catalysts suffice. *Theoretical Computer Sci.*, to appear.
2. Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, 2002.
3. Gh. Păun, Computing with membranes. *Journal of computer and Systems Sciences*, 61(1), 2000, 108–143.
4. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, P systems with tables of rules. In: Gh. Păun, A. RiscosNúñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds., *Proceedings of the Second Brainstorming Week on Membrane Computing*, Report RGNC 01/04, 2004, 366–380.
5. Gh. Păun, S. Yu, On synchronization in P systems. *Fundamenta Informaticae*, 38, 4 (1999), 397–410.
6. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, *Teoría de la Complejidad en modelos de computacion celular con membranas*, Ed. Kronos, Sevilla, 2002.
7. M.J. Pérez-Jiménez, A. Riscos-Núñez, Solving the Subset-Sum problem by active membranes. *New Generation Computing*, in press.
8. M.J. Pérez-Jiménez, A. Riscos-Núñez, A linear-time solution for the Knapsack problem using active membranes. *Lecture Notes in Computer Science*, 2933 (2004) 140–152.
9. M.J. Pérez-Jiménez, F.J. Romero-Campero, A CLIPS simulator for recognizer P systems with active membranes. In: Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds., *Proceedings of the Second Brainstorming Week on Membrane Computing*, Report RGNC 01/04, 2004, 387–413.
10. 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, Gy. Vaszyl, eds., *Proceedings of the Fifth International Workshop on Descriptional Complexity of Formal Systems*, 2003, 284–294.